目录

[1. EasyCrackMe 1](#_Toc485647751)

[2. EasyKeygen 2](#_Toc485647752)

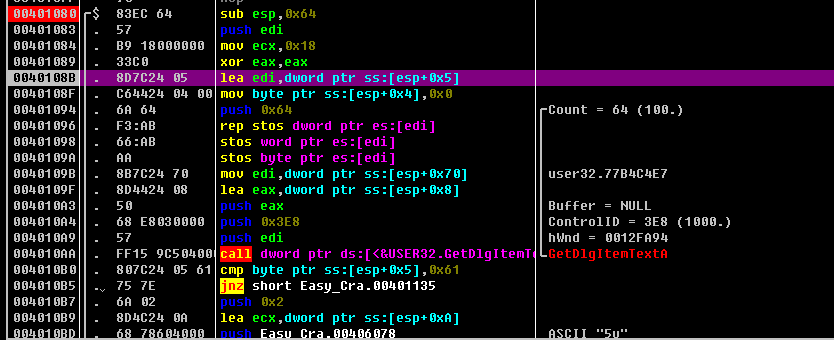
[3. Easy\_UnpackMe 3](#_Toc485647753)

[4. Easy\_ELF 3](#_Toc485647754)

[5. RePlace 3](#_Toc485647755)

[6. Direct3D FPS 5](#_Toc485647756)

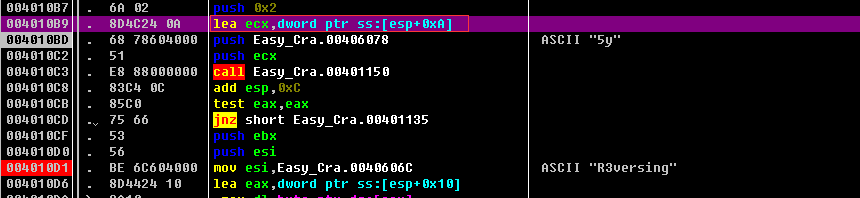
# EasyCrackMe



这段代码是读取对话框中输入的字符串，分配一个100字节的buffer。



判断第二个字符为“a”



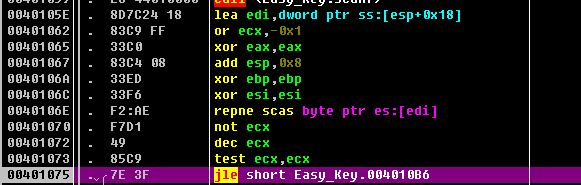
Lea 传递字符串地址。



比较“E”

Flag ==”Ea5yR3versing”

# EasyKeygen



即：repnz scasb（32位地址操作）。扫描es:edi指向的一系列字节数据，扫描长度由ecx指定，当遇到与al中的数据相等时停止扫描。

最经典的求字符串长度的代码，strlen()在VC优化编译模式是这段代码。

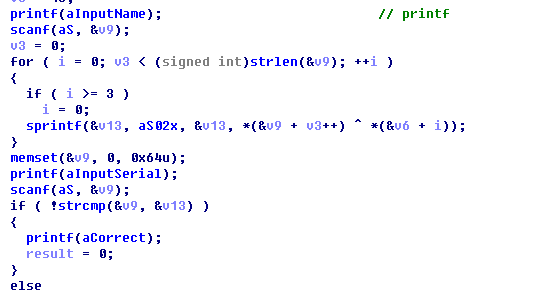
得到的字符串最后存放在ecx中。

总结：该段代码是strlen的汇编代码，返回值存放在ECX中。

JLE跳转是 <= 跳转。该段代码的功能是当strlen() <= 0 时，跳转代码。



注册机验证函数：

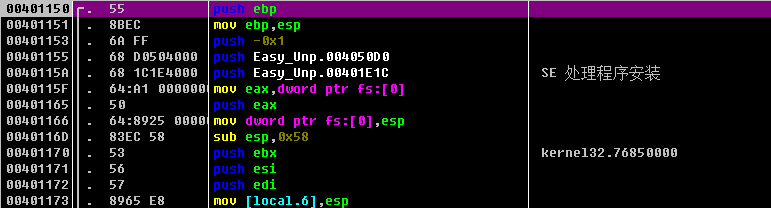


跑出的结果是K3yg3nm3

# Easy\_UnpackMe



上图为解密之后的jmp。跳转之后是常见函数入口。



# Easy\_ELF

# RePlace

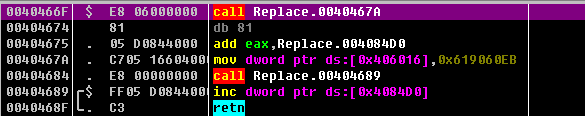


401000地址出的函数为主要函数。



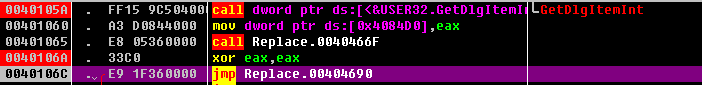
函数读取对话框中输入的数字。（按整数格式读取）。





EAX读取的值 + 0x6015cb，得到的数存放在4084d0地址处。





Jmp到404690处。





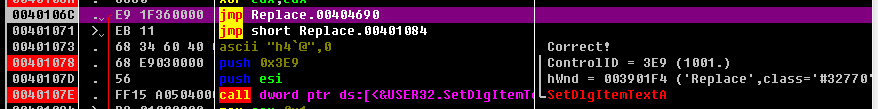
4084d0地址存放的数据自增1.



将40466f处的指令修改为c39000c6.，也就是将[EAX]=601605cc 修改成Nop



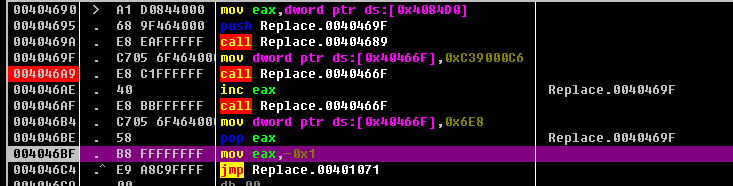
解题思路：



1．输入数据，然后程序运行到00401071，跳转到4046a9,eax=0x601605cb+输入的数据，这个值必须要等于0x100401071。然后这个eax值

（0040466F $ C600 90 mov byte ptr ds:[eax],0x90 ）也就是该中的eax值，程序会把eax也就是00401071中的指令修改为nop.

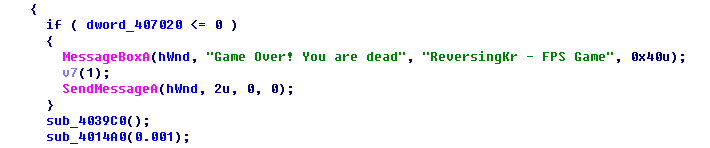
2,运行到004046AE。eax+1=00401072，在运行到004046af.程序会把eax也就是004010712中的指令修改为nop

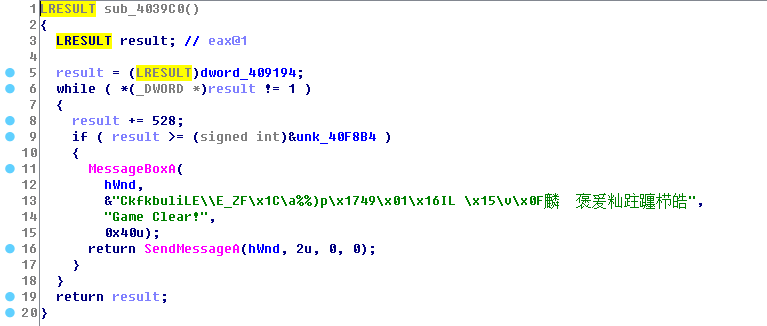




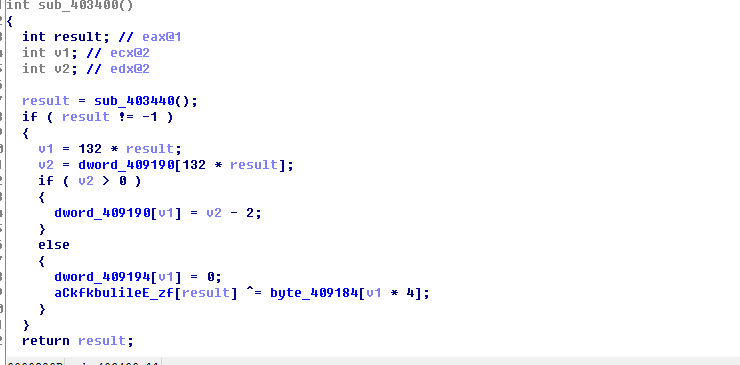
所以2687109798即为最后的结果。

# Direct3D FPS





Game Clear中的乱码就是flag。



查看交叉引用，发现乱码采用异或加密。

for i in range(0, len(s)):

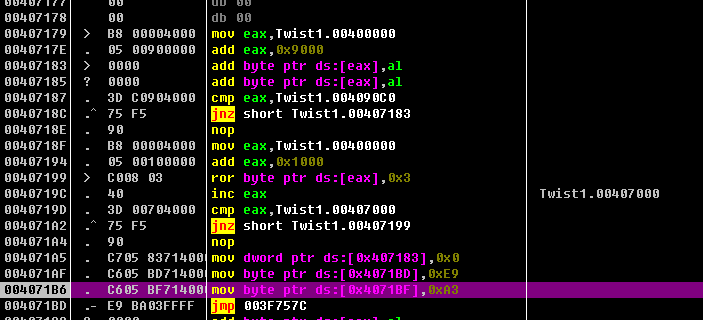
flag.append(s[i] ^ (i \* 4))

结果是Congratulation~ Game Clear! Password is Thr3EDPr0mÈ，因为乱码字符串最后一个字符为\0，所以flag是Thr3EDPr0m

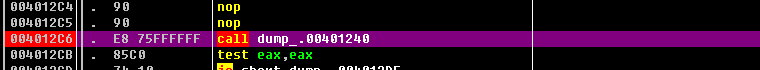
# Twist1

程序加壳，ctrl+f7自动调试，经过多次解密后跳转到如下界面。

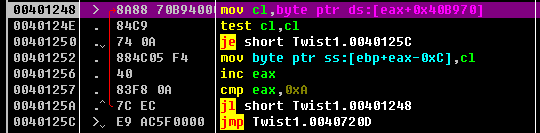
4071BD 处的long jmp，跳转到程序真正入口点。



如果看的不习惯可以用ollydump脱壳，修复IAT。输入Input之后，跳转到验证函数。



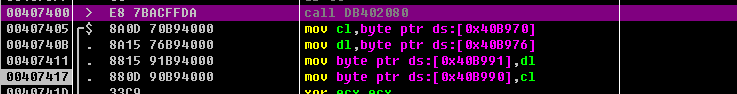
计算字符串长度。



然后跳转到抗调试和解密部分。



字符串第一个字母写入cl，第七个字符写入dl，然后写入40b991地址处。



哎呀我操抗调试代码。。。。



就是解密出来的字符串，然后调用，要绕过这里和下面的getthreadcontext，直接把call 编程nop。

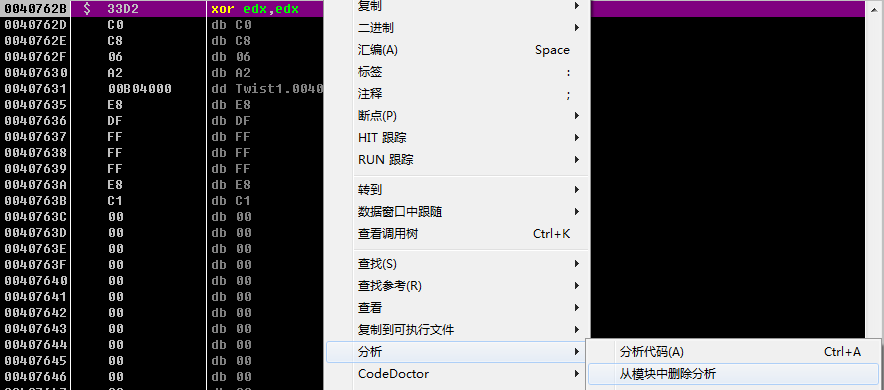
下面有个判断，将字符串第七个字符和0x36异或，然后异或结果和0x36比较。就是判断第七个字符是不是’\0’，也就是字符串长度是不是6



真正的解密函数



步入函数后，解决乱码的方法是：



第一位 ror 6 然后 xor 0x49,，结果是R



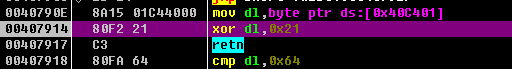
第二位 xor 0x20 然后cmp 0x69 结果是0x49 I



第三位 xor 0x77 然后比较0x35 结果是0x42 B



第四位 xor 0x21 cmp 0x64 结果是0x45 E



第五位 xor 0x46 cmp 0x8 N





第六位 取出字符，然后右移四位，并比较0x14, 结果是A







结果是RIBENA