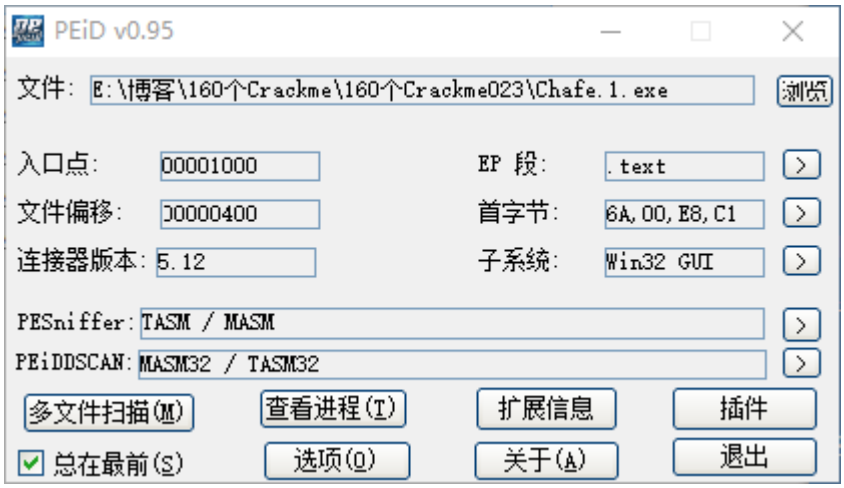
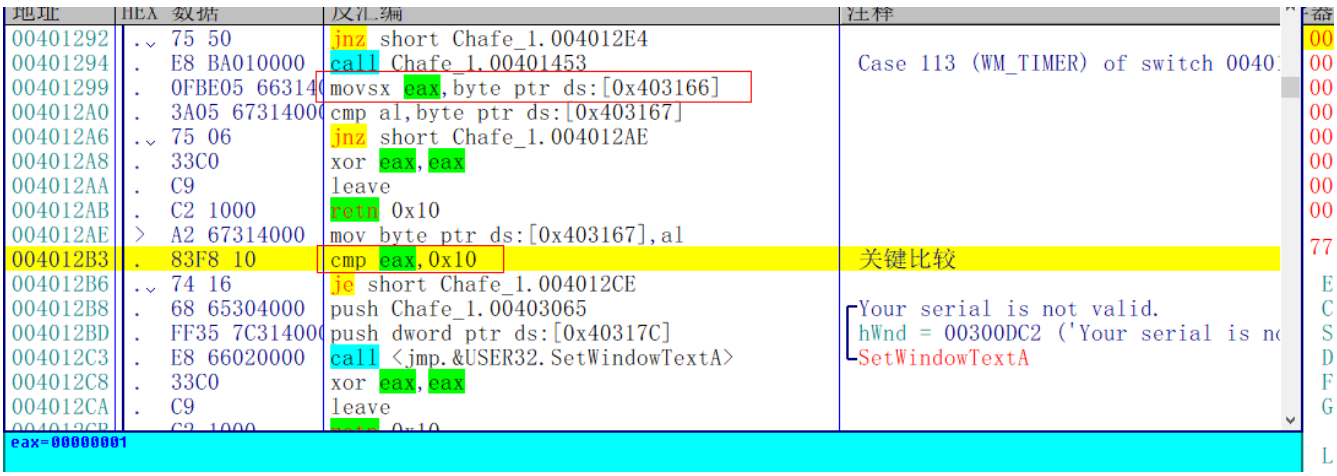
查壳



目标程序是一个用汇编写的带图形界面的程序，没有壳

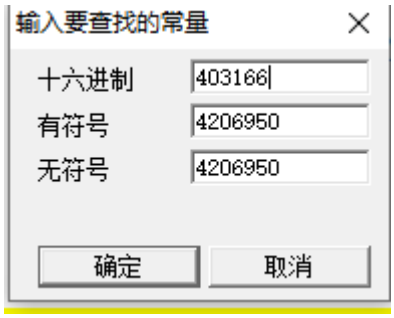
分析程序

直接查找错误的字符串，来到错误的提示处，可以看到这个cmp就是关键比较了



这里会比较eax是否等于0x10，而eax来自于0x403166，所以必须让0x这个地址的值为0x10才能注册成功

直接右键->查找所有常量




```
地址      反汇编
00401093  add byte ptr ds:[0x403166],0x4
004010A3  mov byte ptr ds:[0x403166],ah
00401299  movsx eax,byte ptr ds:[0x403166]
004012AE  mov byte ptr ds:[0x403167],al
00401398  add byte ptr ds:[0x403166],0x4
00401465  movsx eax,byte ptr ds:[0x403166]
00401493  add byte ptr ds:[0x403166],0x4
004014AA  add byte ptr ds:[0x403166],0x4
004014B3  mov byte ptr ds:[0x403166],0x0
```

这里有四个地址分别对0x403166进行了+4的操作，只要同时满足四个条件，就能注册成功了

分析条件一

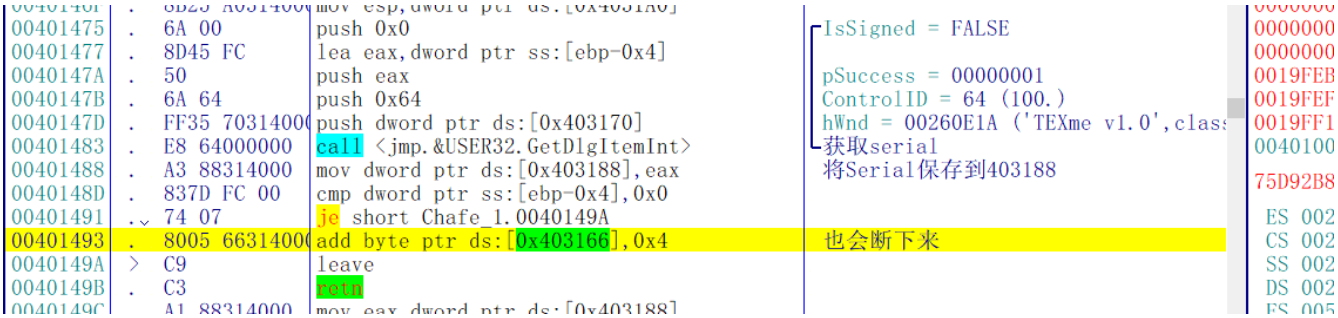
首先来分析第一个地址的401093处的代码

```
00401069  .  6A 14          push 0x14                          Count = 14 (20.)
0040106B  .  68 8C314000    push Chafe_1.0040318C              GuiShou
00401070  .  FF35 74314000  push dword ptr ds:[0x403174]       hWnd = 000D0E14 (class='Edit',parer
00401076  .  E8 7D040000    call <jmp.&USER32.GetWindowTextA>  获取用户名
0040107B  .  B9 14000000    mov ecx,0x14
00401080  .  2BC8           sub ecx,eax                        ecx=14-用户名长度
00401082  .  8DB8 8C314000  lea edi,dword ptr ds:[eax+0x40318C] GuiShou
00401088  >  C607 00        mov byte ptr ds:[edi],0x0
0040108B  .  47             inc edi                            Chafe_1.<ModuleEntryPoint>
0040108C  .  49             dec ecx
0040108D  .^ 75 F9          jnz short Chafe_1.00401088
0040108F  .  85C0           test eax,eax                       判断用户名长度是否为零
00401091  .v 74 10          je short Chafe_1.004010A3
00401093  .  8005 66314000  add byte ptr ds:[0x403166],0x4     一定会断下来
0040109A  .  C605 68314000  mov byte ptr ds:[0x403168],0x0
004010A1  .v EB 06          jmp short Chafe_1.004010A9
004010A3  >  8825 66314000  mov byte ptr ds:[0x403166],ah
004010A9  >  C9             leave
004010AA  .  C3             retn
004010AB  >  68 007F0000    push 0x7F00                        RsrcName = IDC_ARROW
004010B0  .  6A 00          push 0x0                           hInst = NULL
004010B2  .  E8 53040000    call <jmp.&USER32.LoadCursorA>     LoadCursorA
```

这里首先获取用户名，然后判断用户名长度是否为零，不为零则对0x403166这个地址执行+4操作，所以这个地方一定会断下来

分析条件三

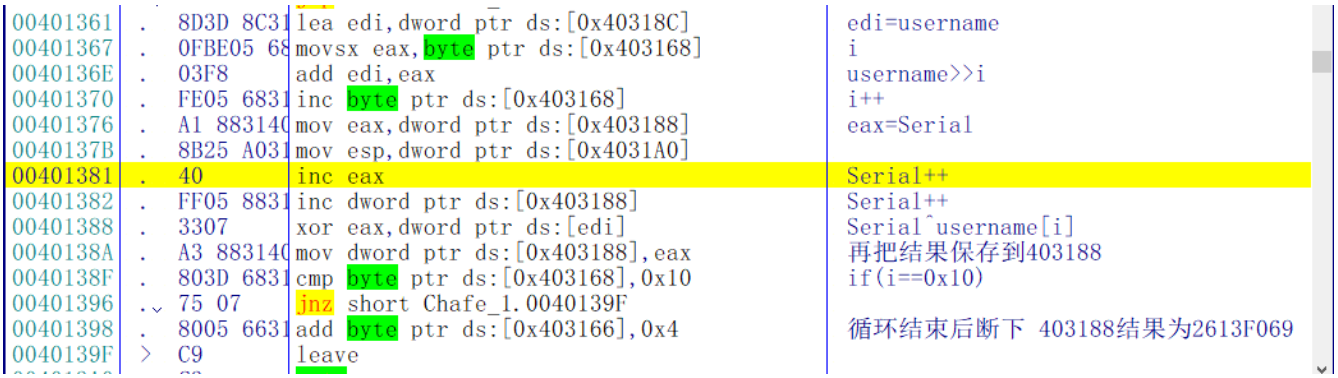
为什么先看条件三，因为我之前已经分析完了，这四个位置有一个先后顺序的问题，按照顺序再来到00401493这个地址



这里会获取输入的序列号，然后将序列号保存到0x403188这个地址，这个地址很重要，这个地址也是一定会断下来的

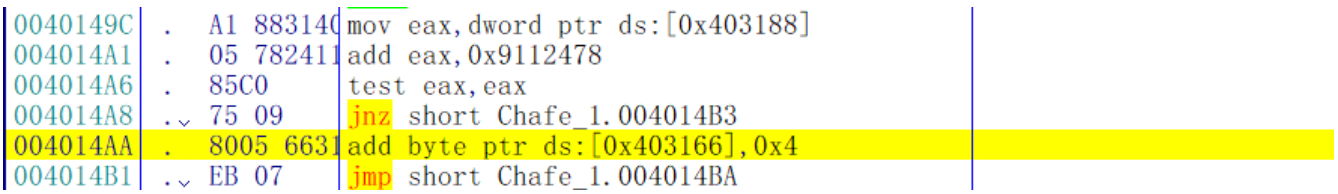
分析条件二

再来看条件二，这里就是这个程序校验的算法了，校验过程如下



1. 获取用户名
2. 设置循环次数，初始值i=0
3. 用户名右移i位
4. i++
5. Serial++
6. Serial和username[i]进行异或(username[i])指的是用户名左移i位后前四个字母的ASCII值)
7. 保存结果到403188处
8. 循环0x10次

分析条件四



这里首先会取出0x403188的结果，然后加上0x9112478，接着比较eax是否为零，为零则403166这个位置加上4，从这里可以得出条件三的算法结果必须为0-0x9112478=0xf6eedb88

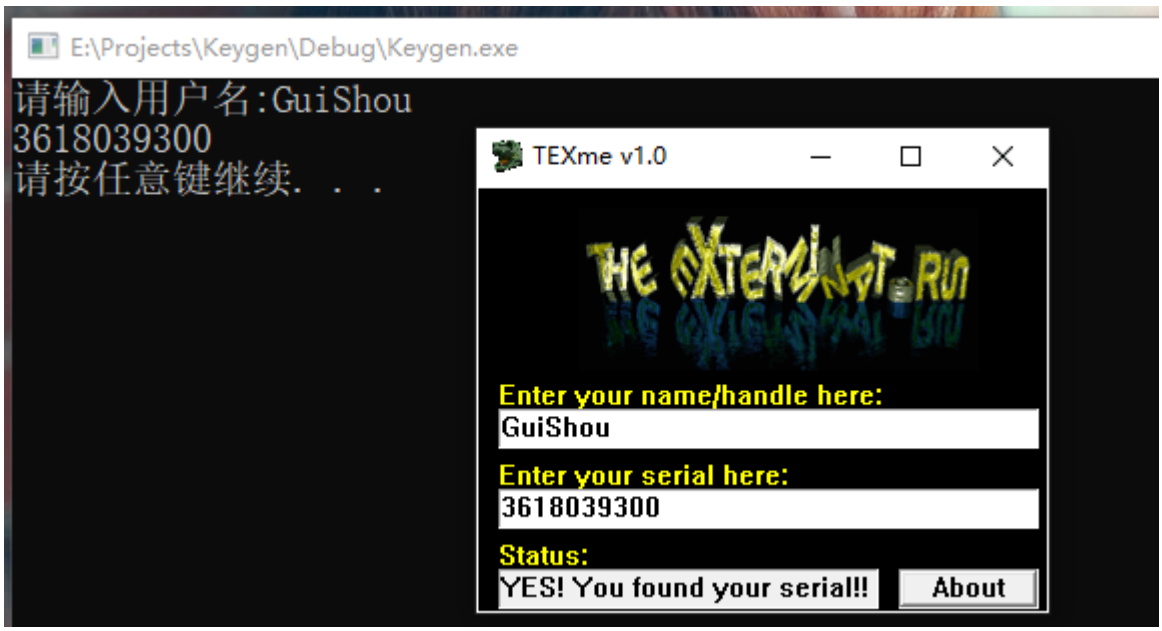
写出注册机

根据条件三的结果，我们可以直接逆推出注册机

```
int CalcKey()
{
    char* name;
    unsigned long serial = 0xF6EEDB88;
    unsigned long *p;
    name = new char[20]{0};

    cout << "请输入用户名:";
    gets_s(name, strlen(name) - 1);
    for (int i= 0x10 - 1; i >= 0; i--)
    {
        p = (unsigned long *)&name[i];
        serial ^= *p;
        serial--;
    }
    cout <<serial << endl;
    return 0;
}
```

输入用户名和计算的序列号，注册成功，破解完成



需要相关文件的可以到我的Github下载：https://github.com/TonyChen56/160-Crackme