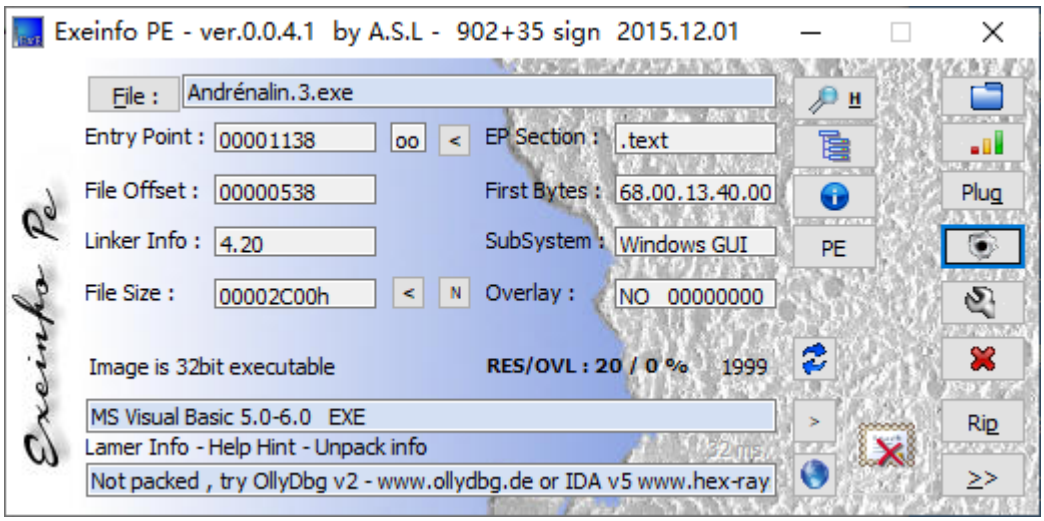
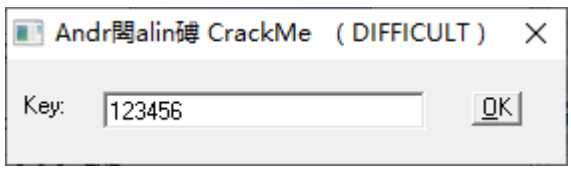
查壳



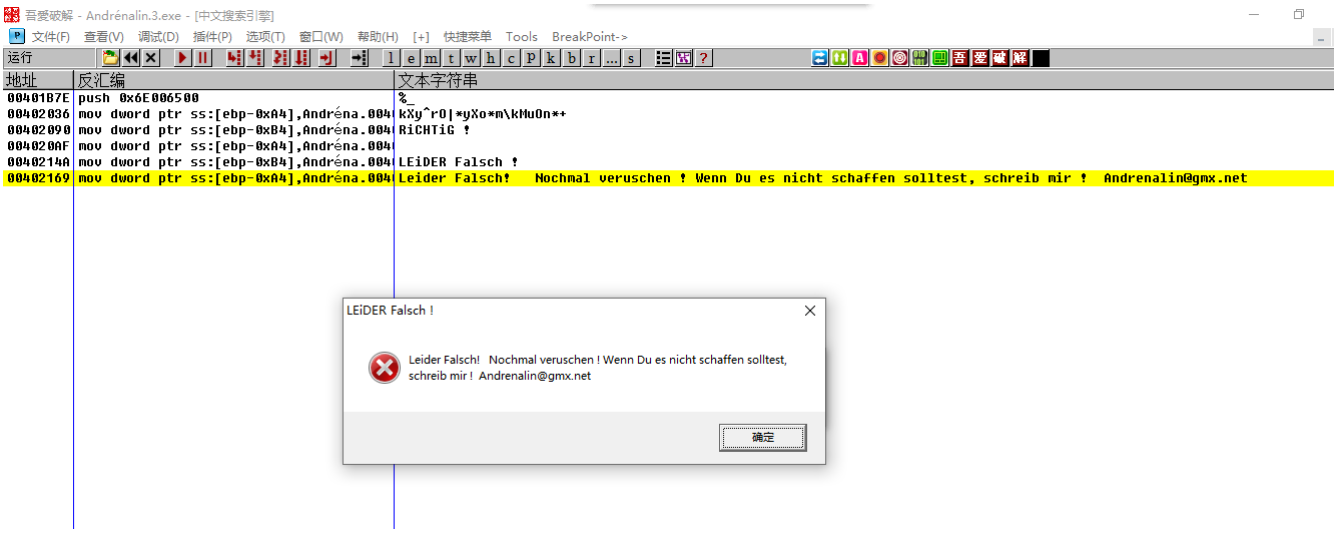
跟008和009这两个crackme一样是同一个作者，还是用VB写的。

分析程序

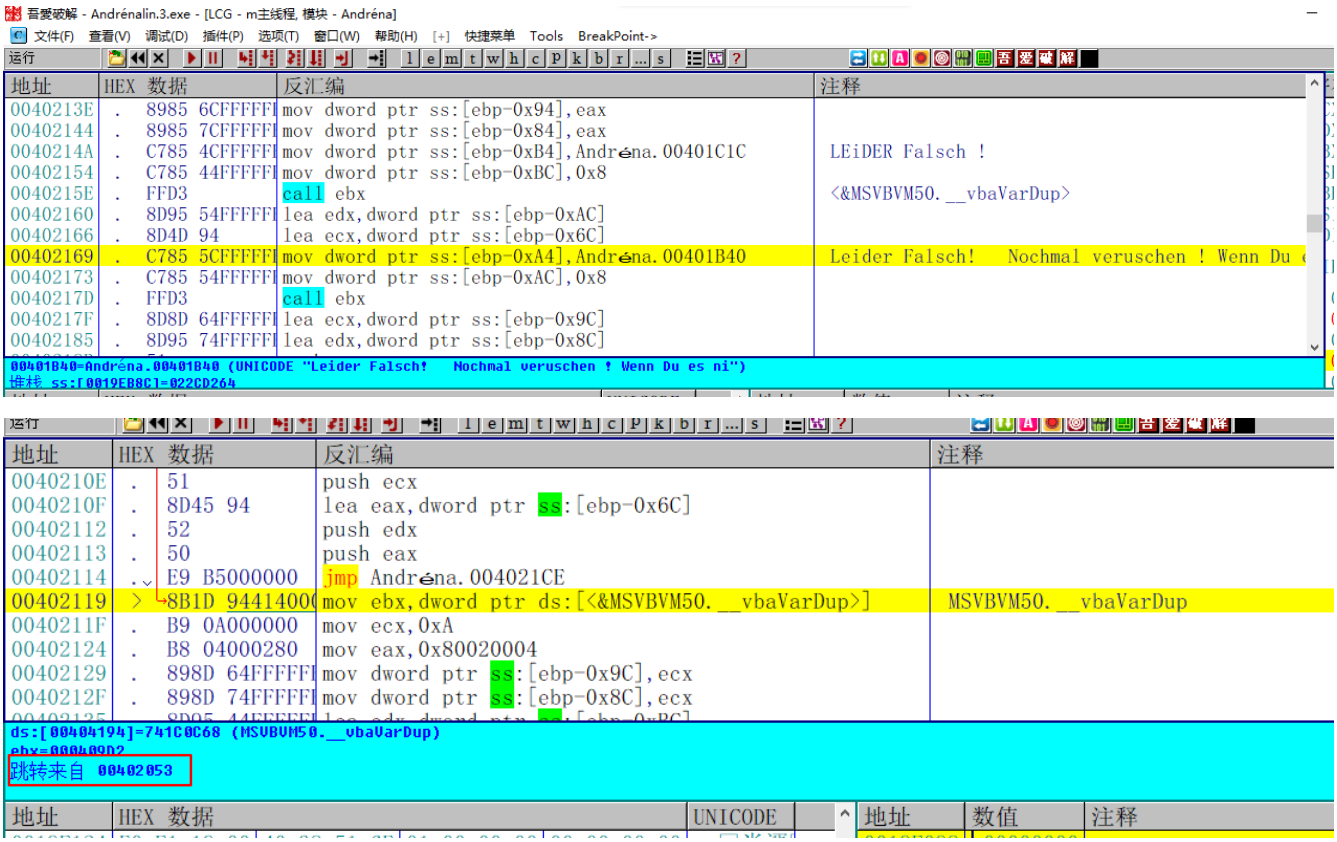
这个程序的保护方式也很简单，只有一个序列号



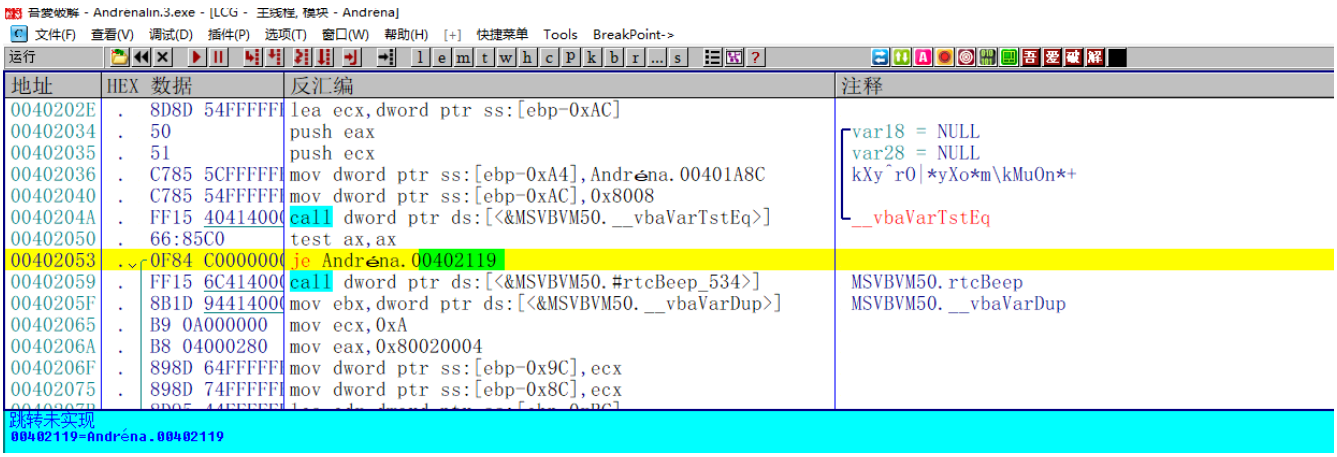
直接搜索字符串，



根据错误提示来到关键代码处，接着一直往上找，



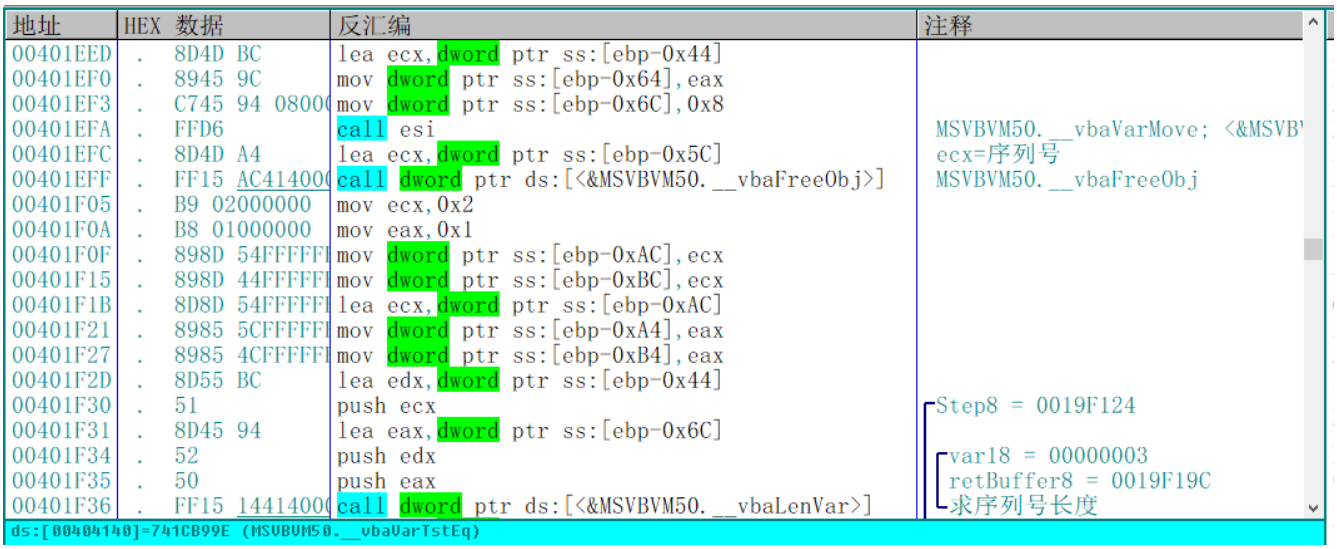
接着就看到了这个地方的跳转来自0x402053这个位置，跟过去看看



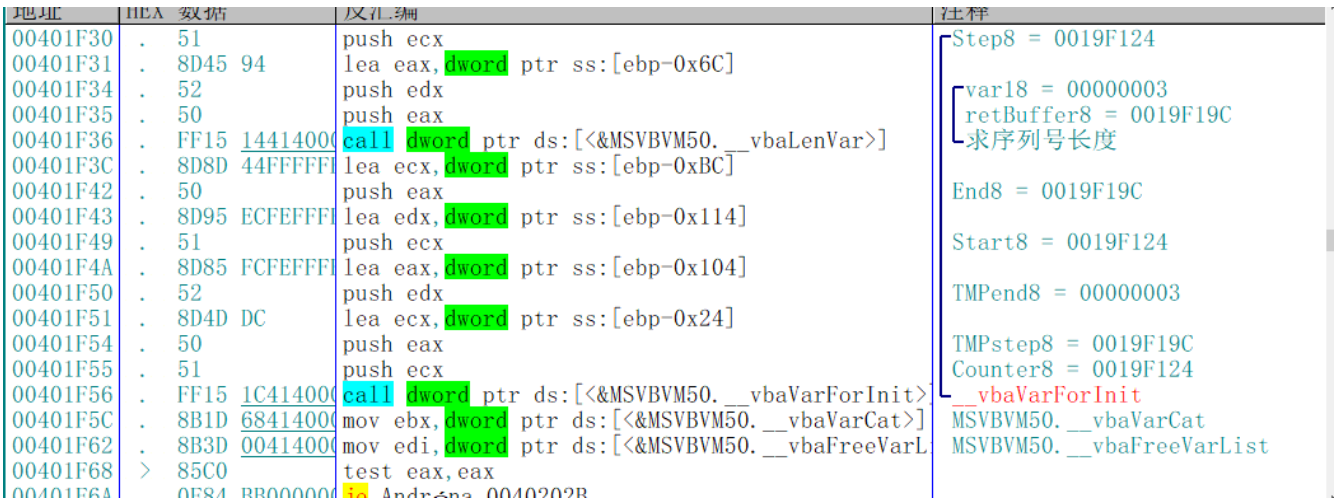
这里有一个关键的比较，根据ax的值来提示是否正确。 `kXy^rO|*yXo*m\kMuOn*+`我一开始以为上面这个字符串就是序列号，结果发现我想多了，看来是有算法，没办法，单步跟吧。

分析算法

大致的校验过程如下



首先获取到序列号的长度



然后将序列号的长度作为循环的次数

```
地址      HEX 数据      反汇编                                                  注释
00401F94  .   52        push edx                                                RetBUFFER = 00000003
00401F95  .   FF15 3441400( call dword ptr ds:[<&MSVBVM50.#rtcMidCharVar_6     rtcMidCharVar
00401F9B  .   8D45 84     lea eax,dword ptr ss:[ebp-0x7C]                         把序列号转成ASCII
00401F9E  .   8D4D A8     lea ecx,dword ptr ss:[ebp-0x58]
00401FA1  .   50          push eax                                                String8 = 0019F19C
00401FA2  .   51          push ecx                                                ARG2 = 0019F124
00401FA3  .   FF15 6441400( call dword ptr ds:[<&MSVBVM50.__vbaStrVarVal>]        取出序列号的ASCII每一位
00401FA9  .   50          push eax                                                String = "□"
00401FAA  .   FF15 0841400( call dword ptr ds:[<&MSVBVM50.#rtcAnsiValueBst       取出序列号的ASCII值
00401FB0  .   66:05 0A00  add ax,0xA                                              序列号的ASCII值+0xA
00401FB4  .v  0F80 B002000( jo Andréna.0040226A                                   溢出则报错
00401FBA  .   0FBFD0      movsx edx,ax                                            edx=序列号的ASCII值+0xA
00401FBD  .   52          push edx
00401FBE  .   FF15 7041400( call dword ptr ds:[<&MSVBVM50.#rtcBstrFromAnsi       把相加后的数值转为字符串
00401FC4  .   8985 7CFFFFF mov dword ptr ss:[ebp-0x84],eax                        把字符串保存到[ebp-0x84]
00401FCA  .   8D45 CC     lea eax,dword ptr ss:[ebp-0x34]
00401FCD  .   8D8D 74FFFFF lea ecx,dword ptr ss:[ebp-0x8C]                        ecx=序列号ASCII+0xA的值
00401FD3  .   50          push eax
00401FD4  .   8D95 64FFFFF lea edx,dword ptr ss:[ebp-0x9C]
ds:[00404164]=7411A825 (MSVBVM50.__vbaStrVarVal)
```

接着会取出序列号ASCII的每一位，然后将序列号的ASCII值+0xA之后，再将这个值转为字符串。

```
00401FC4  .   8985 7CFFFFF mov dword ptr ss:[ebp-0x84],eax                        把字符串保存到[ebp-0x84]
00401FCA  .   8D45 CC     lea eax,dword ptr ss:[ebp-0x34]
00401FCD  .   8D8D 74FFFFF lea ecx,dword ptr ss:[ebp-0x8C]                        ecx=序列号ASCII+0xA的值
00401FD3  .   50          push eax
00401FD4  .   8D95 64FFFFF lea edx,dword ptr ss:[ebp-0x9C]
00401FDA  .   51          push ecx
00401FDB  .   52          push edx
00401FDC  .   C785 74FFFFF mov dword ptr ss:[ebp-0x8C],0x8
00401FE6  .   FFD3        call ebx                                                拼接转换后的字符串
00401FE8  .   8BD0        mov edx,eax
00401FEA  .   8D4D CC     lea ecx,dword ptr ss:[ebp-0x34]
00401FED  .   FFD6        call esi                                                把结果移动到[ebp-0x34]
00401FEF  .   8D4D A8     lea ecx,dword ptr ss:[ebp-0x58]
00401FF2  .   FF15 B041400( call dword ptr ds:[<&MSVBVM50.__vbaFreeStr>]          MSVBVM50.__vbaFreeStr
00401FF8  .   8D85 74FFFFF lea eax,dword ptr ss:[ebp-0x8C]
00401FFE  .   8D4D 84     lea ecx,dword ptr ss:[ebp-0x7C]
00402001  .   50          push eax
00402002  .   8D55 94     lea edx,dword ptr ss:[ebp-0x6C]
00402005  .   51          push ecx
```

然后又是国际惯例了，会通过一个函数将结果保存到[ebp-0x34]这个位置。(VB的程序每次都是这样，这不是给逆向人员开绿色通道吗？)那么这个循环我们只要跟了一次，然后直接看[ebp-0x34]的结果就可以了。

```
00402020  .   FF15 A441400( call dword ptr ds:[<&MSVBVM50.__vbaVarForNext>]     __vbaVarForNext                  ECX 0019F124
00402026  .^  E9 3DFFFFFF  jmp Andréna.00401F68                                                               EDX 00000003
0040202B  >   8D45 CC      lea eax,dword ptr ss:[ebp-0x34]                                                    EBX 740C4B24 MSVBVM5(
0040202E  .   8D8D 54FFFFF lea ecx,dword ptr ss:[ebp-0xAC]                                                    ESP 0019F098
00402034  .   50           push eax                                        var18 = 0019F19C                   EBP 0019F1D0
00402035  .   51           push ecx                                        var28 = 0019F124                   ESI 741C0E23 MSVBVM5(
00402036  .   C785 5CFFFFF mov dword ptr ss:[ebp-0xA4],Andréna.00401A8C    kXy^rO|*yXo*m\kMuOn*+              EDI 741C182B MSVBVM5(
00402040  .   C785 54FFFFF mov dword ptr ss:[ebp-0xAC],0x8008                                                  EIP 0040204A Andréna.
0040204A  .   FF15 4041400( call dword ptr ds:[<&MSVBVM50.__vbaVarTstEq>]     __vbaVarTstEq                    C 0  ES 002B 32位 0(F
00402050  .   66:85C0      test ax,ax                                                                         P 1  CS 0023 32位 0(F
00402053  .v  0F84 C000000( je Andréna.00402119                                                               A 0  SS 002B 32位 0(F
00402059  .   FF15 6C41400( call dword ptr ds:[<&MSVBVM50.#rtcBeep_534>]        MSVBVM50.rtcBeep                 Z 1  DS 002B 32位 0(F
0040205F      8B1D 9441400( mov ebx,dword ptr ds:[<&MSVBVM50.__vbaVarDup>]     MSVBVM50.__vbaVarDup             S 0  FS 0053 32位 285
ds:[00404140]=741CB99E (MSVBVM50.__vbaVarTstEq)
```

```
地址      HEX 数据                                        UNICODE
00682974  3B 00 3C 00 3D 00 3E 00 3F 00 40 00 41 00 42 00 ;<=>?@AB
00682984  43 00 00 00 00 00 59 2A 59 5C 77 8D E7 9E 62 C5 C..×厠起
00682994  00 0B 00 88 12 00 00 00 31 00 32 00 33 00 34 00 □蝻□.12
006829A4  35 00 36 00 37 00 38 00 39 00 00 00 79 00 00 00 56789.y.
006829B4  B4 F6 19 00 E2 9E 1F C5 00 0C 00 8C 3C 13 14 6E   □黢炎
006829C4  00 00 00 00 02 00 00 00 80 6F 66 00 00 00 00 00 .. .澀f.
006829D4  00 00 00 00 03 00 00 00 70 00 00 00 E9 9E 18 C5 .. □.p.黢
006829E4  00 0D 00 88 60 1C AC 76 00 10 AC 76 01 00 00 00 □蝻o嚯ヅ砪
006829F4  30 B5 64 00 5A DD 13 17 E7 68 5B 4A 9A F6 59 2A 叱d □栈靴
00682A04  59 5C 77 8D 94 9E 15 C5 00 0E 00 88 B4 DE 4B 6E 厠起癘□
```

```
地址      数值      注释
0019F098  0019F124  var28 = 0019F124
0019F09C  0019F19C  var18 = 0019F19C
0019F0A0  0019F1DC
0019F0A4  0019F2B8  ASCII " 茜"
0019F0A8  022CD194
0019F0AC  0004B604  UNICODE "-security-logon-11-1-1"
0019F0B0  005B0FC0
0019F0B4  00000000
0019F0B8  FFB6171D
0019F0BC  00000003
```

循环结束之后，会把最后的结果跟硬编码的一个字符串作比较，根据比较的结果提示正确或者错误。那么我们很容易就能写出这个crackme的注册机了

写出注册机

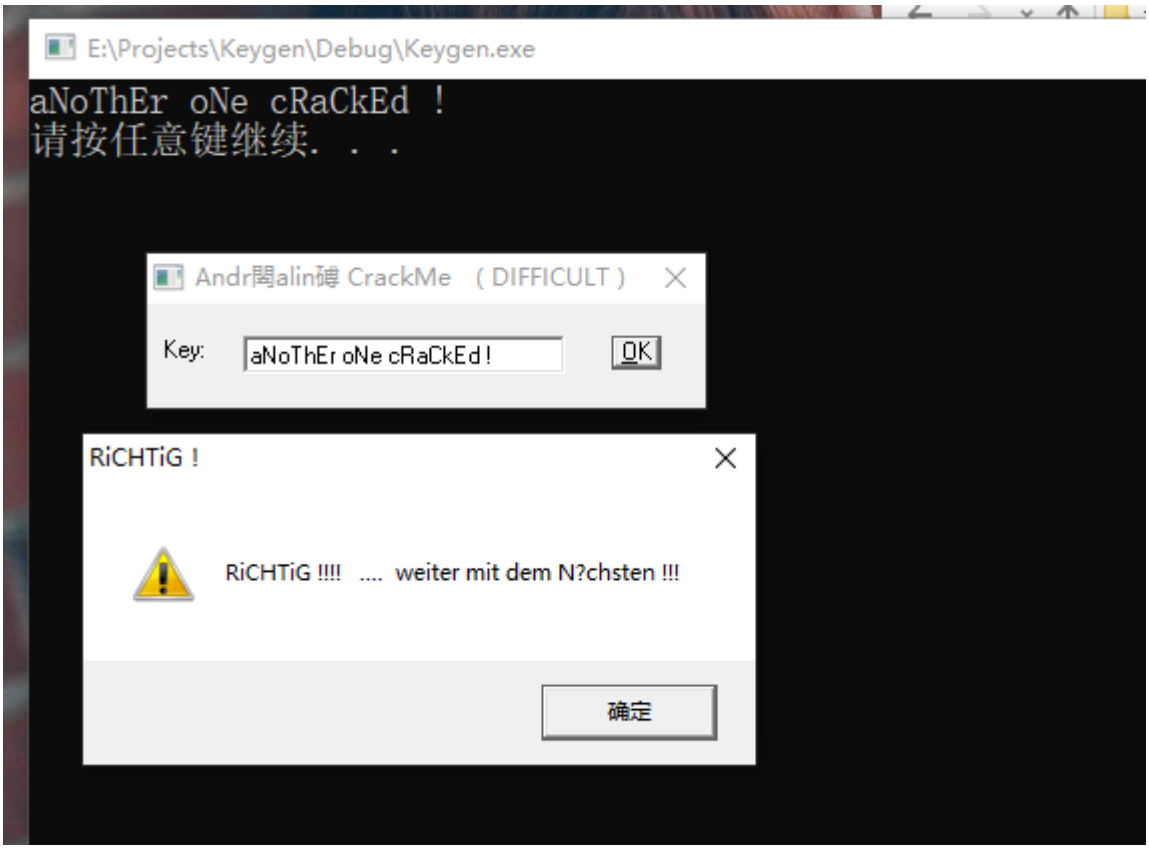
```
int CalcKey()
{
    char result[MAX_PATH] = { 0 };
    char key[MAX_PATH] = { "kXy^rO|*yXo*m\\kMuOn*+" };
    int keyLen = strlen(key);

    for (int i = 0; i < keyLen; i++)
    {
        result[i] = key[i] - 0xA;
    }

    printf("%s\n", result);

    return 0;
}
```

验证结果



把注册机的结果全部复制下来，显示正确，那么这个crackme就完成了

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