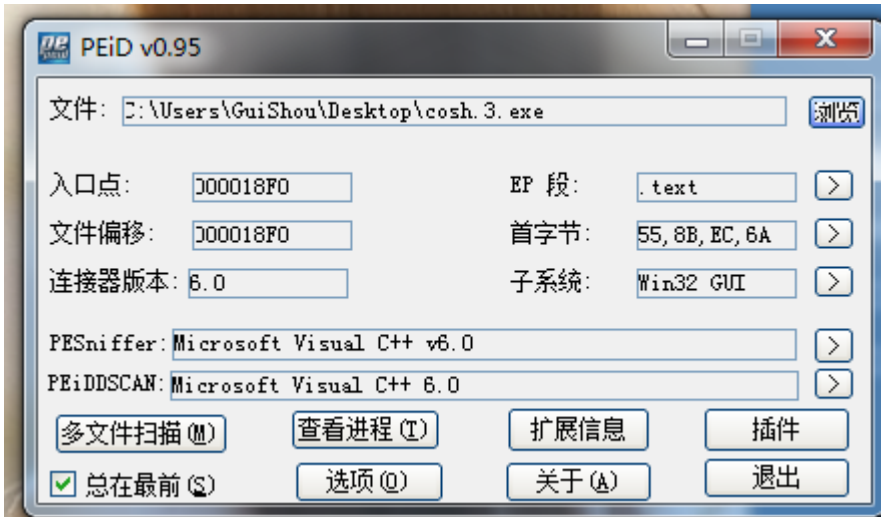


- 查壳
- 分析程序
- 分析算法
- 写出注册机
- 校验结果

查壳



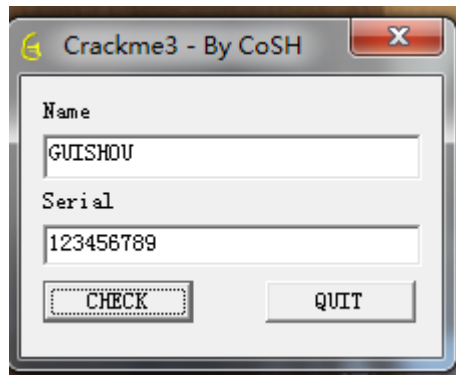
这个Crackme跟027和028是同一个作者，VC6写的，难度为一颗星

分析程序

| 地址 | HEX 数据 | 反汇编 | 注释 |
|----------|-----------------|---|--|
| 004015C3 | > 6A 00 | push 0x0 | |
| 004015C5 | . 68 6C304000 | push cosh_3.0040306C | ERROR |
| 004015CA | . 68 40304000 | push cosh_3.00403040 | One of the Details you entered was wrong |
| 004015CF | . 8B4D E0 | mov ecx,[local.8] | |
| 004015D2 | . E8 BB020000 | call <jmp.&MFC42.#CWnd::MessageBoxA_4224> | |
| 004015D7 | < EB 14 | jmp short cosh_3.004015ED | |
| 004015D9 | > 6A 00 | push 0x0 | |
| 004015DB | . 68 34304000 | push cosh_3.00403034 | YOU DID IT |
| 004015E0 | . 68 20304000 | push cosh_3.00403020 | Well done, Cracker |
| 004015E5 | . 8B4D E0 | mov ecx,[local.8] | |
| 004015E8 | . E8 A5020000 | call <jmp.&MFC42.#CWnd::MessageBoxA_4224> | |
| 004015ED | > 6A 64 | push 0x64 | Timeout = 100. ms |
| 004015EF | . FF15 00204000 | call dword ptr ds:[<&KERNEL32.Sleep>] | Sleep |
| 004015F5 | . C645 FC 00 | mov byte ptr ss:[ebp-0x4],0x0 | |
| 004015F9 | . 8D4D F0 | lea ecx,[local.4] | |
| 004015FC | . E8 65010000 | call <jmp.&MFC42.#CString::~CString_800> | |
| 00401601 | . C745 FC FFFF | mov [local.1],-0x1 | |
| 00401608 | . 8D4D E4 | lea ecx,[local.7] | |
| 0040160B | . E8 56010000 | call <jmp.&MFC42.#CString::~CString_800> | |
| 00401610 | . 8B4D F4 | mov ecx,[local.3] | mfc42.722340A5 |
| 00401613 | . 64:890D 0000 | mov dword ptr fs:[0],ecx | |
| 0040161A | . 5F | pop edi | mfc42.722340A5 |
| 0040161B | . 5E | pop esi | mfc42.722340A5 |

同样，根据字符串的错误提示，来到函数头的位置，配合IDA的伪代码分析整个算法，

分析算法



随便输入一个用户名和序列号，算法的校验过程如下。这个Crackme跟028一样有花指令的干扰，如果想在IDA中看到F5的伪代码，需要手动去除花指令，去除的方法请参考我的Crackme027的分析

```

15
16 this_1 = this;
17 CString::CString((CString *)&username_1);
18 v13 = 0;
19 CString::CString((CString *)&Serial);
20 LOBYTE(v13) = 1;
21 UsernameLength = CWnd::GetWindowTextLengthA((CWnd *)((char *)this_1 + 160)); // 获取用户名长度
22 if ( UsernameLength > 5 ) // 用户名长度必须大于5
23 {
24     v10 = CWnd::GetWindowTextLengthA((CWnd *)((char *)this_1 + 96)); // 获取序列号长度
25     if ( v10 > 5 )
26     {
27         CWnd::GetWindowTextA((CWnd *)((char *)this_1 + 160), (CWnd *)((char *)this_1 + 224)); // 获取用户名
28         CWnd::GetWindowTextA((CWnd *)((char *)this_1 + 96), (CWnd *)((char *)this_1 + 228)); // 获取序列号
29         CString::operator+=(username_1, (char *)this_1 + 224);
30         CString::operator+=(Serial, (char *)this_1 + 228);
31         index = 1;
32         username = username_1;
33         do
34             *username++ ^= index++; // 用户名第i位的ASCII值和i进行异或
35         while ( *username );
36         i = 10;
37         Serial_1 = Serial;
38         do
39             *Serial_1++ ^= i++; // 序列号第i位的ASCII值和i进行异或
40         while ( *Serial_1 );
41         CalcUsername = username_1;
42         CalcSerial = Serial;
43         while ( *CalcUsername == *CalcSerial ) // 比较用户名和序列号的每一位
44         {
45             ++CalcUsername;
46             ++CalcSerial;
47             if ( !*CalcUsername )
48             {
49                 CWnd::MessageBoxA(this_1, aWellDoneCracke, aYouDidIt, 0); // 比较通过则提示正确
50                 goto LABEL_12;
51             }
52         }
53     }
54 }
55 CWnd::MessageBoxA(this_1, aOneOfTheDetail, aError, 0); // 比较不通过则提示错误
56 LABEL_12:
57 Sleep(0x64u);
58 LOBYTE(v13) = 0;
59 CString::~CString((CString *)&Serial);
60 v13 = -1;
61 return CString::~CString((CString *)&username_1);
62 }

```

1. 获取用户名长度 比较是否大于5

| 地址 | HEX 数据 | 反汇编 | 注释 | 寄存器 (FPU) |
|----------|-----------------|--|------------|--------------------------------------|
| 004014CC | . 56 | push esi | | EAX 00000007 |
| 004014CD | . 57 | push edi | | ECX 00760A70 |
| 004014CE | . 894D E0 | mov [local.8],ecx | | EDX 00000030 |
| 004014D1 | . 8D4D E4 | lea ecx,[local.7] | | EBX 00000111 |
| 004014D4 | . E8 83030000 | call <jmp.&MFC42.#CString::CString_540> | | ESP 0018F720 |
| 004014D9 | . C745 FC 0000 | mov [local.1],0x0 | | EBP 0018F74C |
| 004014E0 | . 8D4D F0 | lea ecx,[local.4] | | ESI 00000001 |
| 004014E3 | . E8 74030000 | call <jmp.&MFC42.#CString::CString_540> | | EDI 00000000 |
| 004014E8 | . C645 FC 01 | mov byte ptr ss:[ebp-0x4],0x1 | | EIP 004014FA cosh_3.(|
| 004014EC | . 8B4D E0 | mov ecx,[local.8] | | C 0 ES 002B 32位 0(F |
| 004014EF | . 81C1 A0000000 | add ecx,0xA0 | | P 1 CS 0023 32位 0(F |
| 004014F5 | . E8 AA030000 | call <jmp.&MFC42.#CWnd::GetWindowTextLengthA_3876> | 获取用户名长度 | A 0 SS 002B 32位 0(F |
| 004014FA | . 8945 EC | mov [local.5],eax | | Z 1 DS 002B 32位 0(F |
| 004014FD | . 837D EC 05 | cmp [local.5],0x5 | 用户名长度必须大于5 | S 0 FS 0053 32位 7E |
| 00401501 | . 7F 05 | jg short cosh_3.00401508 | | T 0 GS 002B 32位 0(F |
| 00401503 | . E9 BB000000 | jmp cosh_3.00401503 | | D 0 |
| 00401508 | . 8B4D E0 | mov ecx,[local.8] | | O 0 LastErr ERROR_SUCCESS (00000000) |
| 0040150B | . 83C1 60 | add ecx,0x60 | | EFL 00000246 (NO,NB,E,BE,NS,PE,GE,L |
| 0040150E | . E8 91030000 | call <jmp.&MFC42.#CWnd::GetWindowTextLengthA_3876> | 获取用户名长度 | ST0 empty 0.0 |
| 00401513 | . 8945 E8 | mov [local.6],eax | | ST1 empty 0.0 |
| 00401516 | . 837D E8 05 | cmp [local.6],0x5 | 序列号长度必须大于5 | ST2 empty 0.0 |
| 0040151A | . 7F 05 | jg short cosh_3.00401521 | | ST3 empty 0.0 |
| 0040151C | . E9 A2000000 | jmp cosh_3.004015C3 | | |
| 00401521 | . 8B45 E0 | mov eax,[local.8] | | |
| 00401524 | . 05 E0000000 | add eax,0xE0 | | |
| 00401529 | . 50 | push eax | | |
| 0040152A | . 8B4D E0 | mov ecx,[local.8] | | |
| 0040152D | . 81C1 A0000000 | add ecx,0xA0 | | |
| 00401533 | . E8 66030000 | call <jmp.&MFC42.#CWnd::GetWindowTextA_3874> | 获取用户名 | |

2. 获取序列号长度 比较是否大于5

| 地址 | HEX 数据 | 反汇编 | 注释 | 寄存器 (FPU) |
|----------|-----------------|--|------------|--------------------------------------|
| 004014E0 | . 8D4D F0 | lea ecx,[local.4] | | EAX 00000009 |
| 004014E3 | . E8 74030000 | call <jmp.&MFC42.#CString::CString_540> | | ECX 00760A70 |
| 004014E8 | . C645 FC 01 | mov byte ptr ss:[ebp-0x4],0x1 | | EDX 00000030 |
| 004014EC | . 8B4D E0 | mov ecx,[local.8] | | EBX 00000111 |
| 004014EF | . 81C1 A0000000 | add ecx,0xA0 | | ESP 0018F720 |
| 004014F5 | . E8 AA030000 | call <jmp.&MFC42.#CWnd::GetWindowTextLengthA_3876> | 获取用户名长度 | EBP 0018F74C |
| 004014FA | . 8945 EC | mov [local.5],eax | | ESI 00000001 |
| 004014FD | . 837D EC 05 | cmp [local.5],0x5 | 用户名长度必须大于5 | EDI 00000000 |
| 00401501 | . 7F 05 | jg short cosh_3.00401508 | | EIP 00401516 cosh_3.00401516 |
| 00401503 | . E9 BB000000 | jmp cosh_3.00401503 | | C 0 ES 002B 32位 0(F |
| 00401508 | . 8B4D E0 | mov ecx,[local.8] | | P 1 CS 0023 32位 0(F |
| 0040150B | . 83C1 60 | add ecx,0x60 | | A 0 SS 002B 32位 0(F |
| 0040150E | . E8 91030000 | call <jmp.&MFC42.#CWnd::GetWindowTextLengthA_3876> | 获取序列号长度 | Z 1 DS 002B 32位 0(F |
| 00401513 | . 8945 E8 | mov [local.6],eax | | S 0 FS 0053 32位 7E |
| 00401516 | . 837D E8 05 | cmp [local.6],0x5 | 序列号长度必须大于5 | T 0 GS 002B 32位 0(F |
| 0040151A | . 7F 05 | jg short cosh_3.00401521 | | D 0 |
| 0040151C | . E9 A2000000 | jmp cosh_3.004015C3 | | O 0 LastErr ERROR_SUCCESS (00000000) |
| 00401521 | . 8B45 E0 | mov eax,[local.8] | | EFL 00000246 (NO,NB,E,BE,NS,PE,GE,L |
| 00401524 | . 05 E0000000 | add eax,0xE0 | | ST0 empty 0.0 |
| 00401529 | . 50 | push eax | | ST1 empty 0.0 |
| 0040152A | . 8B4D E0 | mov ecx,[local.8] | | ST2 empty 0.0 |
| 0040152D | . 81C1 A0000000 | add ecx,0xA0 | | ST3 empty 0.0 |
| 00401533 | . E8 66030000 | call <jmp.&MFC42.#CWnd::GetWindowTextA_3874> | 获取用户名 | |

3. 根据用户名计算，得出结果，算法转换为C++代码如下：

| 地址 | HEX 数据 | 反汇编 | 注释 | 寄存器 (FPU) |
|----------|---------------|---|---------------|---------------------------------------|
| 00401557 | . 8D4D E4 | lea ecx,[local.7] | | EAX 00372648 ASCII "GUISHOU" |
| 0040155A | . E8 39030000 | call <jmp.&MFC42.#CString::operator=_858> | | ECX 00000001 |
| 0040155F | . 8B45 E0 | mov eax,[local.8] | | EDX 00000000 |
| 00401562 | . 05 E4000000 | add eax,0xE4 | | EBX 00000000 |
| 00401567 | . 50 | push eax | | ESP 0018F720 |
| 00401568 | . 8D4D F0 | lea ecx,[local.4] | | EBP 0018F74C |
| 0040156B | . E8 28030000 | call <jmp.&MFC42.#CString::operator=_858> | | ESI 00000001 |
| 00401570 | . 33C0 | xor eax,eax | | EDI 00000000 |
| 00401572 | . 33DB | xor ebx,ebx | | EIP 00401580 cosh_3.00401580 |
| 00401574 | . 33C9 | xor ecx,ecx | | C 0 ES 002B 32位 0(F |
| 00401576 | . B9 01000000 | mov ecx,0x1 | ecx=1 | P 1 CS 0023 32位 0(F |
| 0040157B | . 33D2 | xor edx,edx | | A 0 SS 002B 32位 0(F |
| 0040157D | . 8B45 E4 | mov eax,[local.7] | | Z 1 DS 002B 32位 0(F |
| 00401580 | . 8A18 | mov bl,byte ptr ds:[eax] | 取出用户名的ASCII值 | S 0 FS 0053 32位 7E |
| 00401582 | . 32D9 | xor bl,cl | | T 0 GS 002B 32位 0(F |
| 00401584 | . 8818 | mov byte ptr ds:[eax],bl | username[i]`i | D 0 |
| 00401586 | . 41 | inc ecx | i++ | O 0 LastErr ERROR_SUCCESS (00000000) |
| 00401587 | . 40 | inc eax | username[i]++ | EFL 00000246 (NO,NB,E,BE,NS,PE,GE,LE) |
| 00401588 | . 8038 00 | cmp byte ptr ds:[eax],0x0 | | ST0 empty 0.0 |
| 0040158B | . 75 F3 | jnz short cosh_3.00401580 | | ST1 empty 0.0 |
| 0040158D | . 33C0 | xor eax,eax | | ST2 empty 0.0 |
| 0040158F | . 33DB | xor ebx,ebx | | ST3 empty 0.0 |
| 00401591 | . 33C9 | xor ecx,ecx | | |

```
do
    *username++ ^= index++;
while ( *username );
```

4. 根据序列号计算，得出结果，算法转换为C++代码如下：

| 地址 | HEX 数据 | 反汇编 | 注释 | 寄存器 (FPU) |
|----------|---------------|---------------------------|---------------|---------------------------------------|
| 00401582 | . 32D9 | xor bl,cl | username[i]~i | EAX 00372698 ASCII "123456789" |
| 00401584 | . 8818 | mov byte ptr ds:[eax],bl | | ECX 0000000A |
| 00401586 | . 41 | inc ecx | i++ | EDX 00000000 |
| 00401587 | . 40 | inc eax | username[i]++ | EBX 00000000 |
| 00401588 | . 8038 00 | cmp byte ptr ds:[eax],0x0 | | ESP 0018F720 |
| 0040158B | . 75 F3 | jnz short cosh_3.00401580 | | EBP 0018F74C |
| 0040158C | . 33C0 | xor eax,eax | | ESI 00000001 |
| 0040158F | . 33DB | xor ebx,ebx | | EDI 00000000 |
| 00401591 | . 33C9 | xor ecx,ecx | | EIP 0040159D cosh_3.0040159D |
| 00401593 | . B9 0A000000 | mov ecx,0xA | | C 0 ES 002B 32位 0(FFFFFFFF) |
| 00401598 | . 33D2 | xor edx,edx | | P 1 CS 0023 32位 0(FFFFFFFF) |
| 0040159A | . 8B45 F0 | mov eax,[local.4] | eax=序列号 | A 0 SS 002B 32位 0(FFFFFFFF) |
| 0040159D | . 8A18 | mov bl,byte ptr ds:[eax] | | Z 1 DS 002B 32位 0(FFFFFFFF) |
| 0040159F | . 32D9 | xor bl,cl | serial[i]~i | S 0 FS 0053 32位 7EFD0000(FFF) |
| 004015A1 | . 8818 | mov byte ptr ds:[eax],bl | | T 0 GS 002B 32位 0(FFFFFFFF) |
| 004015A3 | . 41 | inc ecx | i-- | D 0 |
| 004015A4 | . 40 | inc eax | serial[i]-- | O 0 LastErr ERROR_SUCCESS (00000000) |
| 004015A5 | . 8038 00 | cmp byte ptr ds:[eax],0x0 | | EFL 00000246 (NO,NB,E,BE,NS,PE,GE,LE) |
| 004015A8 | . 75 F3 | jnz short cosh_3.0040159D | | ST0 empty 0.0 |
| 004015AA | . 8B45 E4 | mov eax,[local.7] | 计算后的用户名 | ST1 empty 0.0 |
| 004015AD | . 8B55 F0 | mov edx,[local.4] | 计算后的序列号 | ST2 empty 0.0 |
| 004015B0 | . 33C9 | xor ecx,ecx | | ST3 empty 0.0 |

```

i = 10;
do
    *Serial_1++ ^= i++;
while ( *Serial_1 );

```

对 没错 两个算法都是直接从IDA的伪代码中拷出来的 直接就能用

5. 循环比较用户名的计算结果和序列号的计算结果是否相等，根据比较的结果提示跳转与否

| 地址 | HEX 数据 | 反汇编 | 注释 | 寄存器 (FPU) |
|----------|---------------|---|-----------------------------|---------------------------------------|
| 004015A4 | . 40 | inc eax | serial[i]-- | EAX 00372648 ASCII "FWJWMIR" |
| 004015A5 | . 8038 00 | cmp byte ptr ds:[eax],0x0 | | ECX 0000003B |
| 004015A8 | . 75 F3 | jnz short cosh_3.0040159D | | EDX 00372698 ASCII "':9?9:'")+" |
| 004015AA | . 8B45 E4 | mov eax,[local.7] | 计算后的用户名 | EBX 00000046 |
| 004015AD | . 8B55 F0 | mov edx,[local.4] | 计算后的序列号 | ESP 0018F720 |
| 004015B0 | . 33C9 | xor ecx,ecx | | EBP 0018F74C |
| 004015B2 | . 8A18 | mov bl,byte ptr ds:[eax] | | ESI 00000001 |
| 004015B4 | . 8A0A | mov cl,byte ptr ds:[edx] | | EDI 00000000 |
| 004015B7 | . 3AD9 | cmp bl,cl | 比较用户名和序列号的每一位 | EIP 004015B6 cosh_3.004015B6 |
| 004015B8 | . 75 09 | jnz short cosh_3.004015C3 | | C 0 ES 002B 32位 0(FFFFFFFF) |
| 004015BA | . 40 | inc eax | | P 1 CS 0023 32位 0(FFFFFFFF) |
| 004015BB | . 42 | inc edx | | A 0 SS 002B 32位 0(FFFFFFFF) |
| 004015BC | . 8038 00 | cmp byte ptr ds:[eax],0x0 | | Z 1 DS 002B 32位 0(FFFFFFFF) |
| 004015BF | . 75 EF | jnz short cosh_3.004015B0 | | S 0 FS 0053 32位 7EFD0000(FFF) |
| 004015C1 | . EB 16 | jmp short cosh_3.004015D9 | | T 0 GS 002B 32位 0(FFFFFFFF) |
| 004015C3 | . 6A 00 | push 0x0 | | D 0 |
| 004015C5 | . 68 6C304000 | push cosh_3.0040306C | ERROR | O 0 LastErr ERROR_SUCCESS (00000000) |
| 004015CA | . 68 40304000 | push cosh_3.00403040 | One of the Details you ente | EFL 00000246 (NO,NB,E,BE,NS,PE,GE,LE) |
| 004015CF | . 8B4D E0 | mov ecx,[local.8] | | ST0 empty 0.0 |
| 004015D2 | . E8 BB020000 | call <jmp.&MFC42.#CWnd::MessageBoxA_4224> | | ST1 empty 0.0 |
| 004015D7 | . EB 14 | jmp short cosh_3.004015ED | | ST2 empty 0.0 |
| 004015D9 | . 6A 00 | push 0x0 | | ST3 empty 0.0 |
| 004015DB | . 68 34304000 | push cosh_3.00403034 | YOU DID IT | ST4 empty 0.0 |

也就是说用户名和序列号必须要满足程序中的等式才能注册成功

写出注册机

这个程序的注册机也比较好写，首先根据用户名计算出中间结果，然后再根据结果反向逆推出注册码，代码如下：

```

#include <iostream>
#include <windows.h>

using namespace std;

int main()
{
    char username[20] = { 0 };
    char result[20] = { 0 };
    char key[20] = { 0 };
    printf("请输入用户名 必须为全大写:");
}

```

```

scanf_s("%s", username, 20);
int usernameLength = strlen(username);
if (usernameLength <= 5)
{
    printf("用户名长度必须大于5");
}
//计算中间结果
for (int i=0;i<usernameLength;i++)
{
    username[i] ^= i+1;
    result[i] = username[i];
}

//根据结果逆推注册码

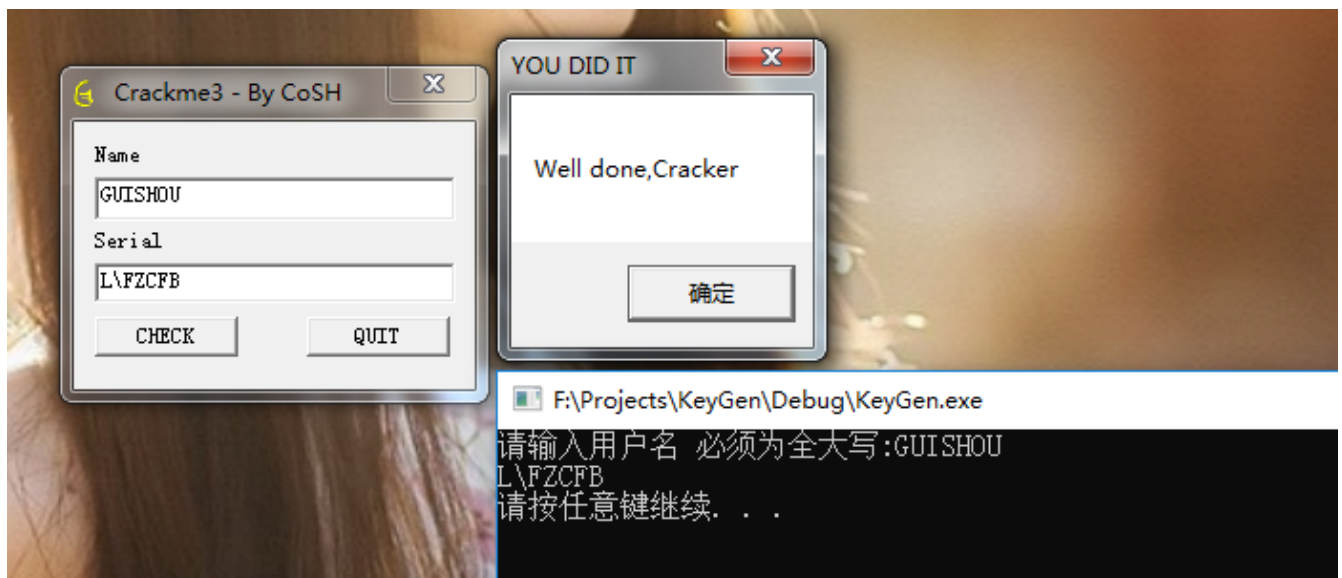
for (int i = 10; i < usernameLength+10; i++)
{
    result[i-10] ^= i;
    key[i-10] = result[i-10];
}

printf("%s\n", key);
system("pause");
return 0;
}

```

校验结果

输入用户名和计算出来的序列号



提示正确 破解成功

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