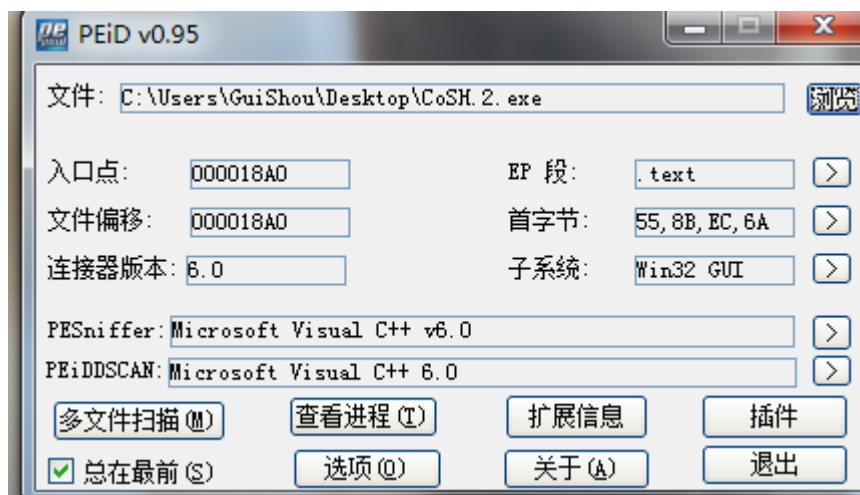


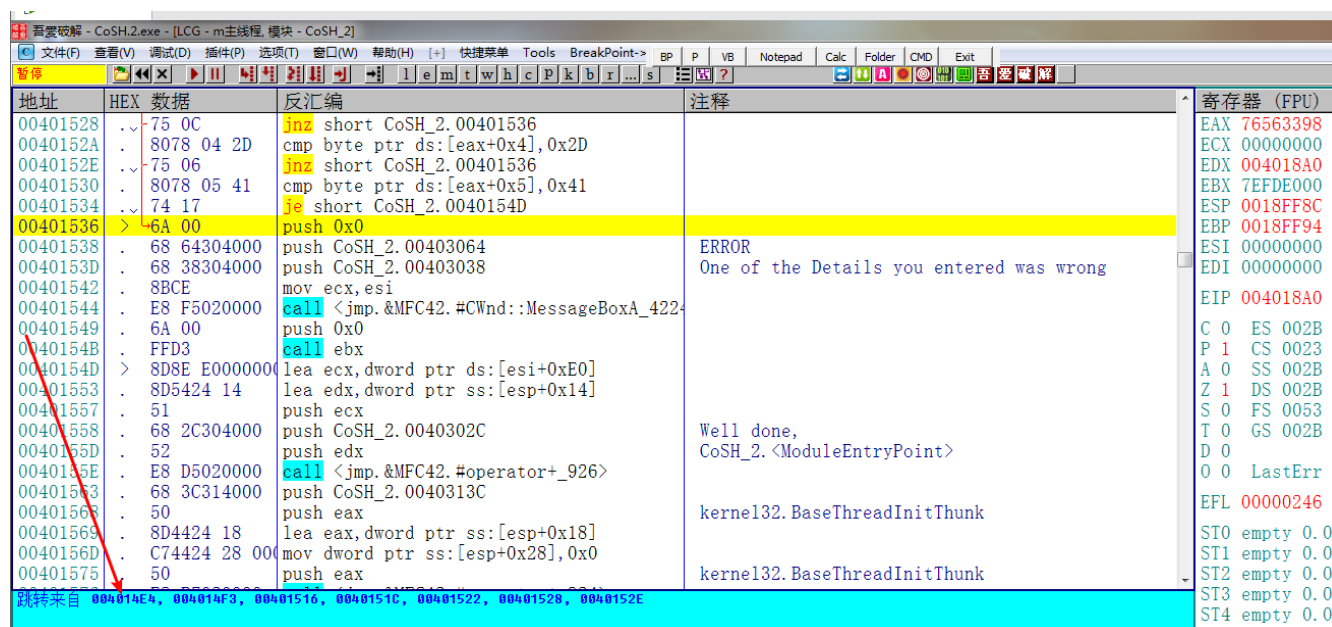
- 查壳
- 分析程序
- 分析算法
- 对抗花指令
- 校验过程
- 校验结果

查壳



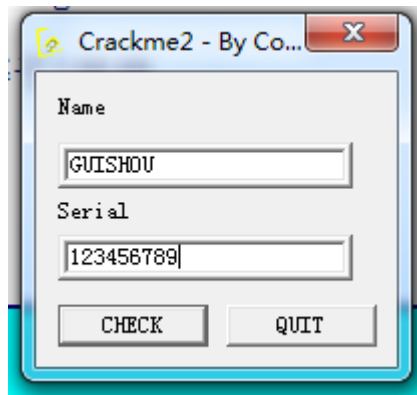
目标程序是用VC6写的，没有加壳，跟027是同一个作者，难度为一颗星

分析程序



首先根据错误提示可以看到跳转到这个地址的地方有很多，我们直接选择最前面那个地址，开始分析整个算法

先随便输入一个序列号



另外，这个程序如果有IDA的帮助会分析的更快

分析算法

地址	HEX 数据	反汇编	注释	寄存器
004014AF	90	nop		EAX 0018
004014B0	64:A1 000000	mov eax,dword ptr fs:[0]		ECX 0018
004014B6	6A FF	push -0x1		EDX 004C
004014B8	68 701B4000	push CoSH_2.00401B70	概%0	EBX 000C
004014BD	50	push eax		ESP 0018
004014BE	64:8925 0000	mov dword ptr fs:[0],esp		EBP 0018
004014C5	83EC 08	sub esp,0x8		ESI 0018
004014C8	53	push ebx		EDI 0018
004014C9	55	push ebp		EIP 004C
004014CA	56	push esi		C 0 ES
004014CB	8BE1	mov esi,ecx		P 1 CS
004014CD	57	push edi		A 1 SS
004014CE	8DBE A0000000	lea edi,dword ptr ds:[esi+0xA0]		Z 0 DS
004014D4	8BCF	mov ecx,edi		S 0 FS
004014D6	E8 6F030000	call <jmp.&MFC42.#CWnd::GetWindowTextLengthA_3876>	获取用户名长度	T 0 GS
004014DB	8B1D FC214000	mov ebx,dword ptr ds:[<&USER32.PostQuitMessage_3876>]	user32.PostQuitMessage	D 0
004014E1	83F8 05	cmp eax,0x5		O 0 Las
004014E4	7E 50	jle short CoSH_2.00401536	用户名长度小于或等于5报错	EFL 000C
004014E6	8D6E 60	lea ebp,dword ptr ds:[esi+0x60]		ST0 empt
004014E9	8BCD	mov ecx,ebp		ST1 empt
004014EB	E8 5A030000	call <jmp.&MFC42.#CWnd::GetWindowTextLengthA_3876>	获取密码长度	ST2 empt
004014F0	83F8 05	cmp eax,0x5		ST3 empt
004014F3	7E 41	jle short CoSH_2.00401536		ST4 emot
0040184A=<jmp.&MFC42.#CWnd::GetWindowTextLengthA_3876>				

IDA View-A Hex View-1 Structures Enums Imports

```

.text:004014CB mov esi, ecx
.text:004014CD push edi
.text:004014CE lea edi, [esi+0A0h]
.text:004014D4 mov ecx, edi
.text:004014D6 call ?GetWindowTextLengthA@CWnd@@QBEXZ ; CWnd::GetWindowTextLengthA(voi
.text:004014DB mov ebx, ds:PostQuitMessage
.text:004014E1 cmp eax, 5
.text:004014E4 jle short loc_401536
.text:004014E6 lea ebp, [esi+60h]
.text:004014E9 mov ecx, ebp
.text:004014EB call ?GetWindowTextLengthA@CWnd@@QBEXZ ; CWnd::GetWindowTextLengthA(voi
.text:004014F0 cmp eax, 5
.text:004014F3 jle short loc_401536
.text:004014F5 lea eax, [esi+0E0h]
.text:004014FB mov ecx, edi
.text:004014FD push eax
.text:004014FE call ?GetWindowTextA@CWnd@@QBEXAAVCString@@@Z ; CWnd::GetWindowTextA(CSt
.text:00401503 lea edi, [esi+0E4h]
.text:00401509 mov ecx, ebp
.text:0040150B push edi
.text:0040150C call ?GetWindowTextA@CWnd@@QBEXAAVCString@@@Z ; CWnd::GetWindowTextA(CSt
.text:00401511
.text:00401513
.text:00401516
.text:00401518
.text:0040151C
.text:0040151E
.text:00401522
.text:00401524 cmp byte ptr [eax+3], 37h
.text:00401528 jnz short loc_401536
.text:0040152A cmp byte ptr [eax+4], 2Dh
.text:0040152E jnz short loc_401536
.text:00401530 cmp byte ptr [eax+5], 41h
.text:00401534 jz short loc_40154D
.text:00401536
.text:00401536 loc_401536: ; CODE XREF: .text:004014E4↑j

```

Warning: Please position the cursor within a function. [X] Don't display this message again (for this session only) [OK]

我们找到算法开始处的地址，对应到IDA中，F5查看一下伪C代码，但是你会发现有个提示，让我们把光标选择一个函数，这表示IDA的反编译失败了!这是为什么呢?

对抗花指令

上一个Crackme作者犯了一个低级错误，写错了一个参数，这一次他玩了一个小把戏，用花指令干扰我们静态分析。现在来破解一个小把戏

OD直接拉到函数开头的部分，你会看到有一堆nop指令

```

004014A0 . 8B81 E8000000 mov eax, dword ptr ds:[ecx+0xE8]
004014A6 . C3          ret
004014A7 . 90          nop
004014A8 . 90          nop
004014A9 . 90          nop
004014AA . 90          nop
004014AB . 90          nop
004014AC . 90          nop
004014AD . 90          nop
004014AE . 90          nop
004014AF . 90          nop
004014B0 . 64:A1 00000000 mov eax, dword ptr fs:[0]
004014B6 . 6A FF       push -0x1
004014B8 . 68 701B4000 push CoSH_2.00401B70
004014BD . 50          push eax

```

Warning: Please position the cursor within a function. [X] Don't display this message again (for this session only) [OK]

在IDA同样地址处这里却不是nop，而是被解释成了align10，IDA的反汇编器在这里出现了错误，把数据解释成了代码

```

.text:004014A6      retn
.text:004014A6 ; -----
.text:004014A7      align 10h
.text:004014B0      mov     eax, large fs:0
.text:004014B6      push    0FFFFFFFh
.text:004014B8      push    offset loc_401B70
.text:004014BD      push    eax
.text:004014BE      mov     large fs:0, esp
.text:004014C5      sub     esp, 8
.text:004014C8      push    ebx
.text:004014C9      push    ebp
.text:004014CA      push    esi
.text:004014CB      mov     esi, ecx
.text:004014CD      push    edi
.text:004014CE      lea     edi, [esi+0A0h]
.text:004014D4      mov     ecx, edi
.text:004014D6      call    ?GetWindowTextLengthA@CWnd@@QBEXZ ; CWnd::GetWindowTextLengthA(void)
.text:004014DB      mov     ebx, ds:PostQuitMessage
.text:004014E1      cmp     eax, 5
.text:004014E4      jle     short loc_401536
.text:004014E6      lea     ebp, [esi+60h]
.text:004014E9      mov     ecx, ebp
.text:004014EB      call    ?GetWindowTextLengthA@CWnd@@QBEXZ ; CWnd::GetWindowTextLengthA(void)
.text:004014F0      cmp     eax, 5
.text:004014F3      jle     short loc_401536
.text:004014F5      lea     eax, [esi+0E0h]
.text:004014FB      mov     ecx, edi

```

那么解决办法就是在这行按字母D键，将代码强行解释为数据，修改后如下：

```

.text:004014A6      retn
.text:004014A6 ; -----
.text:004014A7      db 90h
.text:004014A8      db 90h
.text:004014A9      db 90h
.text:004014AA      db 90h
.text:004014AB      db 90h
.text:004014AC      db 90h
.text:004014AD      db 90h
.text:004014AE      db 90h
.text:004014AF      db 90h
.text:004014B0 ; -----
.text:004014B0      mov     eax, large fs:0
.text:004014B6      push    0FFFFFFFh
.text:004014B8      push    offset loc_401B70
.text:004014BD      push    eax
.text:004014BE      mov     large fs:0, esp
.text:004014C5      sub     esp, 8
.text:004014C8      push    ebx
.text:004014C9      push    ebp
.text:004014CA      push    esi
.text:004014CB      mov     esi, ecx
.text:004014CD      push    edi
.text:004014CE      lea     edi, [esi+0A0h]
.text:004014D4      mov     ecx, edi

```

然后选中整个函数，按P键组合成一个函数

```
1 void __thiscall sub_4014B0(CWnd *this)
2 {
3     CWnd *v1; // esi
4     CWnd *v2; // edi
5     _BYTE v3; // eax
6     int v4; // eax
7     const char *v5; // ST00_4
8     char v6; // [esp+10h] [ebp-14h]
9     char v7; // [esp+14h] [ebp-10h]
10    int v8; // [esp+20h] [ebp-4h]
11
12    v1 = this;
13    v2 = (CWnd *)((char *)this + 160);
14    if ( CWnd::GetWindowTextLengthA((CWnd *)((char *)this + 160)) <= 5// 用户名长度必须大于5
15        || CWnd::GetWindowTextLengthA((CWnd *)((char *)v1 + 96)) <= 5// 序列号长度必须大于5
16        || (CWnd::GetWindowTextA(
17            v2,
18            (CWnd *)((char *)v1 + 224)), // 获取用户名
19            CWnd::GetWindowTextA(
20                (CWnd *)((char *)v1 + 96),
21                (CWnd *)((char *)v1 + 228)), // 获取序列号
22            v3 = (_BYTE *)*((_DWORD *)v1 + 57),
23            **((_BYTE **))v1 + 57) != 54)
24        || v3[1] != 50
25        || v3[2] != 56
26        || v3[3] != 55
27        || v3[4] != 45
28        || v3[5] != 65 )
29    {
30        CWnd::MessageBoxA(v1, aOneOfTheDetail, aError, 0);
31        PostQuitMessage(0);
32    }
33    v4 = operator+(&v7, aWellDone, (char *)v1 + 224);
34    v8 = 0;
35    v5 = *(const char **)operator+(v6, v4, &unk_40313C);
36    LOBYTE(v8) = 1;
37    CWnd::MessageBoxA(v1, v5, aYouDidIt, 0);
38    LOBYTE(v8) = 0;
39    CString::~CString((CString *)&v6);
40    v8 = -1;
41    CString::~CString((CString *)&v7);
42    PostQuitMessage(1);
43 }
```

这个时候你再按F5，就能显示整个函数的伪代码了

校验过程

这个程序的校验过程相对来说比较简单，过程如下

1. 获取用户名长度，比较是否小于等于5

地址	HEX 数据	反汇编	注释	寄存器 (FPU)
004014B6	6A FF	push -0x1		EAX 00000007
004014B8	68 701B4000	push CoSH_2.00401B70	源%	ECX 007F0A70
004014BD	50	push eax		EDX 00000030
004014BE	64:8925 0000	mov dword ptr fs:[0],esp		EBX 00000111
004014C5	83EC 08	sub esp,0x8		ESP 0018F72C
004014C8	53	push ebx		EBP 0018F75C
004014C9	55	push ebp		ESI 0018FDD0
004014CA	56	push esi		EDI 0018FE70
004014CB	8BF1	mov esi,ecx		EIP 004014DB CoSH_2.004014DB
004014CE	57	push edi		C 0 ES 002B 32位 0 (FFFFFFFF)
004014CD	8DBE A0000000	lea edi,dword ptr ds:[esi+0xA0]		P 1 CS 0023 32位 0 (FFFFFFFF)
004014D4	8BCF	mov ecx,edi		A 0 SS 002B 32位 0 (FFFFFFFF)
004014D6	E8 6F030000	call <jmp.&MFC42.#CWnd::GetWindowTextLengthA_3876>	获取用户名长度	Z 1 DS 002B 32位 0 (FFFFFFFF)
004014DB	8B1D FC214000	mov ebx,dword ptr ds:[<&USER32.PostQuitMessage>]	user32.PostQuitMessage	S 0 FS 0053 32位 7EFDD000 (FFF)
004014E1	83F8 05	cmp eax,0x5		I 0 GS 002B 32位 0 (FFFFFFFF)
004014E4	7E 50	jle short CoSH_2.00401536	用户名长度小于或等于5报错	

2. 获取密码长度 比较是否小于等于5

地址	HEX	数据	反汇编	注释	寄存器 (FPU)
04014C5	83EC	08	sub esp,0x8		EAX 00000009
04014C8	53		push ebx	user32.PostQuitMessage	ECX 007F0A70
04014C9	55		push ebp		EDX 00000030
04014CA	56		push esi		EBX 75AC9ABB user32.Post
04014CB	8BF1		mov esi,ecx		ESP 0018F72C
04014CD	57		push edi		EBP 0018FE30
04014CE	8DBE	A0000000	lea edi,dword ptr ds:[esi+0xA0]		ESI 0018FDD0
04014D4	8BCF		mov ecx,edi		EDI 0018FE70
04014D6	E8	6F030000	call <jmp.&MFC42.#CWnd::GetWindowTextLengthA_3876>	获取用户名长度	EIP 004014F0 CoSH_2.00401
04014DB	8B1D	FC214000	mov ebx,dword ptr ds:[<&USER32.PostQuitMessage>]	user32.PostQuitMessage	C 0 ES 002B 32位 0(FFFFFF)
04014E1	83F8	05	cmp eax,0x5		P 1 CS 0023 32位 0(FFFFFF)
04014E4	7E	50	jle short CoSH_2.00401536	用户名长度小于或等于5报错	A 0 SS 002B 32位 0(FFFFFF)
04014E6	8D6E	60	lea ebp,dword ptr ds:[esi+0x60]		Z 1 DS 002B 32位 0(FFFFFF)
04014E9	8BCD		mov ecx,ebp		S 0 FS 0053 32位 7EFDD000
04014EB	E8	5A030000	call <jmp.&MFC42.#CWnd::GetWindowTextLengthA_3876>	获取密码长度	T 0 GS 002B 32位 0(FFFFFF)
04014F0	83F8	05	cmp eax,0x5		D 0
04014F3	7E	41	jle short CoSH_2.00401536		O 0 LastErr ERROR_SUCCESS
04014F5	8D86	E0000000	lea eax,dword ptr ds:[esi+0xE0]		EFL 00000246 (NO, NB, E, BE,
04014FB	8BCF		mov ecx,edi		

3. 获取序列号 开始校验序列号

地址	HEX	数据	反汇编	注释	寄存器 (FPU)
004014FE	E8	41030000	call <jmp.&MFC42.#CWnd::GetWindowTextA_3874>	获取用户名	EAX 002B2698 ASCII "123456789"
00401503	8DBE	E4000000	lea edi,dword ptr ds:[esi+0xE4]		ECX 0018FEB4
00401509	8BCD		mov ecx,ebp		EDX 002B2699 ASCII "23456789"
0040150B	57		push edi		EBX 75AC9ABB user32.PostQuitMessage
0040150C	E8	33030000	call <jmp.&MFC42.#CWnd::GetWindowTextA_3874>	获取序列号	ESP 0018F72C
00401511	8B07		mov eax,dword ptr ds:[edi]		EBP 0018FE30
00401513	8038	36	cmp byte ptr ds:[eax],0x36		ESI 0018FDD0
00401516	75	1E	jnz short CoSH_2.00401536		EDI 0018FEB4
00401518	8078	01 32	cmp byte ptr ds:[eax+0x1],0x32		EIP 00401513 CoSH_2.00401513
0040151C	75	18	jnz short CoSH_2.00401536		C 0 ES 002B 32位 0(FFFFFFFF)
0040151E	8078	02 38	cmp byte ptr ds:[eax+0x2],0x38		P 0 CS 0023 32位 0(FFFFFFFF)
00401522	75	12	jnz short CoSH_2.00401536		A 1 SS 002B 32位 0(FFFFFFFF)
00401524	8078	03 37	cmp byte ptr ds:[eax+0x3],0x37		Z 0 DS 002B 32位 0(FFFFFFFF)
00401528	75	0C	jnz short CoSH_2.00401536		S 0 FS 0053 32位 7EFDD000(FFF)
0040152A	8078	04 2D	cmp byte ptr ds:[eax+0x4],0x2D		T 0 GS 002B 32位 0(FFFFFFFF)
0040152E	75	06	jnz short CoSH_2.00401536		D 0
00401530	8078	05 41	cmp byte ptr ds:[eax+0x5],0x41		O 0 LastErr ERROR_SUCCESS (00000000)
00401534	74	17	jz short CoSH_2.0040154D		EFL 00000212 (NO, NB, NE, A, NS, PO, GE, G)
00401536	6A	00	push 0x0		ST0 empty 0.0
00401538	68	64304000	push CoSH_2.00403064	ERROR	ST1 empty 0.0
0040153D	68	38304000	push CoSH_2.00403038	One of the Details you entered wa	ST2 empty 0.0
00401542	8BCE		mov ecx,esi		ST3 empty 0.0
00401544	E8	F5020000	call <jmp.&MFC42.#CWnd::MessageBoxA_4224>		

可以看到校验的部分就是6个cmp和jcc指令了，直接看IDA

```

10  int v8; // [esp+20h] [ebp-4h]
11
12  v1 = this;
13  v2 = (CWnd *)((char *)this + 160);
14  if ( CWnd::GetWindowTextLengthA((CWnd *)((char *)this + 160)) <= 5 // 用户名长度必须大于5
15  || CWnd::GetWindowTextLengthA((CWnd *)((char *)v1 + 96)) <= 5 // 序列号长度必须大于5
16  || (CWnd::GetWindowTextA(
17      v2,
18      (CWnd *)((char *)v1 + 224)), // 获取用户名
19      CWnd::GetWindowTextA(
20      (CWnd *)((char *)v1 + 96),
21      (CWnd *)((char *)v1 + 228)), // 获取序列号
22      v3 = (_BYTE *)((_DWORD *)v1 + 57),
23      **((_BYTE **)v1 + 57) != 54)
24  || v3[1] != 50
25  || v3[2] != 56
26  || v3[3] != 55
27  || v3[4] != 45
28  || v3[5] != 65 )
29  {
30      CWnd::MessageBoxA(v1, aOneOfTheDetail, aError, 0);
31      PostQuitMessage(0);
32  }
33  v4 = operator+(&v7, dword ptr ds:[esi+0x41], (char *)v1 + 224);

```

这里显示的是一堆十进制的数字，这个对我们来说没有意义，按R键，可以把数字转为字符串，转换后如下

```

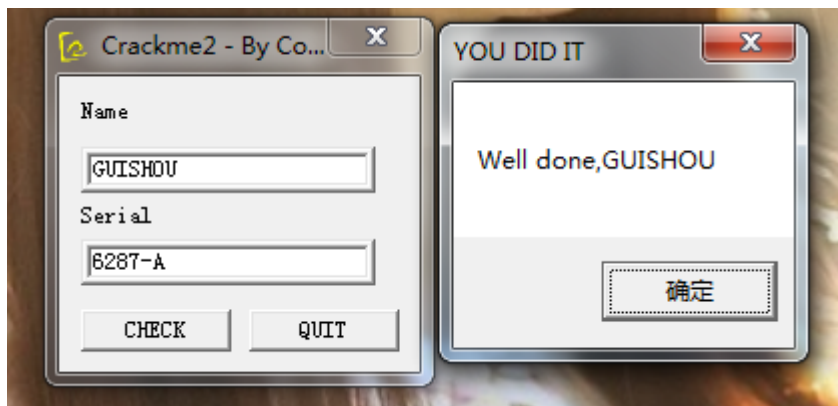
9 char v7; // [esp+14h] [ebp-10h]
10 int v8; // [esp+20h] [ebp-4h]
11
12 v1 = this;
13 v2 = (CWnd *)((char *)this + 160);
14 if ( CWnd::GetWindowTextLengthA((CWnd *)((char *)this + 160)) <= 5 // 用户名长度必须大于5
15 || CWnd::GetWindowTextLengthA((CWnd *)((char *)v1 + 96)) <= 5 // 序列号长度必须大于5
16 || CWnd::GetWindowTextA(
17     v2,
18     (CWnd *)((char *)v1 + 224)), // 获取用户名
19     CWnd::GetWindowTextA(
20     (CWnd *)((char *)v1 + 96),
21     (CWnd *)((char *)v1 + 228)), // 获取序列号
22     v3 = (_BYTE *)*((_DWORD *)v1 + 57),
23     **((_BYTE **)v1 + 57) != '6')
24 || v3[1] != '2'
25 || v3[2] != '8'
26 || v3[3] != '7'
27 || v3[4] != '-'
28 || v3[5] != 'A' )
29 {
30     CWnd::MessageBoxA(v1, aOneOfTheDetail, aError, 0);
31     PostQuitMessage(0);
32 }
33 v4 = operator+(&v7, aWellDone, (char *)v1 + 224);

```

可以很直观的看到序列号就是6287-A

校验结果

随便输入一个用户名，但是长度必须大于5，然后输入刚才的序号，提示正确 破解完成



最后，需要相关文件可以到我的Github下载:

<https://github.com/TonyChen56/160-Crackme>