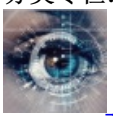
看雪转载——OllyDBG 入门系列（五）一消息断点及 RUN 跟踪

转载

BetaBin 于 2012-05-04 11:39:41 发布 1633 收藏
分类专栏： 安全 文章标签： windows 扩展 工具 算法 button 汇编


安全 专栏收录该内容
34 篇文章 0 订阅
订阅专栏

/*

消息断点偶尔总是不如意，留档学习学习。

*/

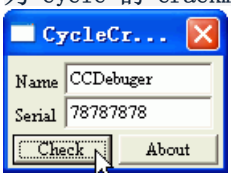
标 题：【原创】OllyDBG 入门系列（五）一消息断点及 RUN 跟踪

作 者：CCDebuger
时 间：2006-02-19,16:02:46
链 接：http://bbs.pediy.com/showthread.php?t=21532

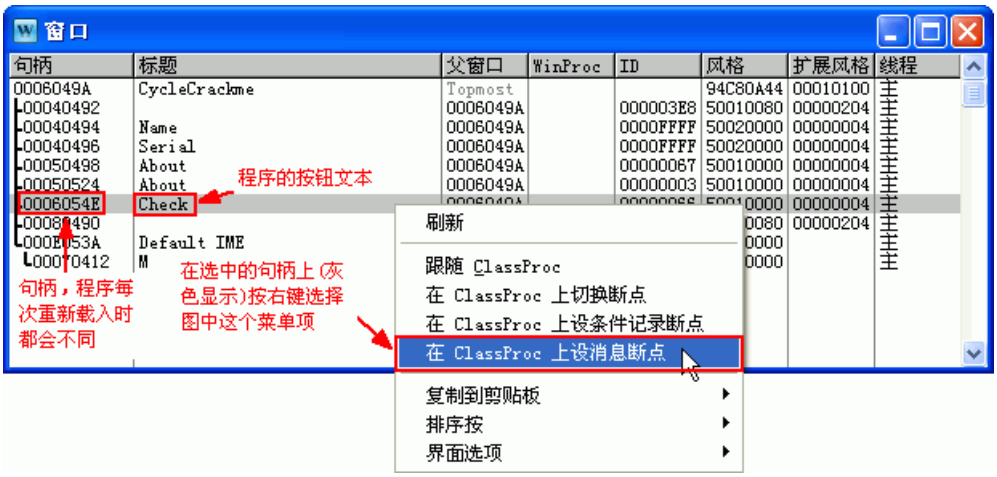
OllyDBG 入门系列（五）一消息断点及 RUN 跟踪

作者：CCDebuger

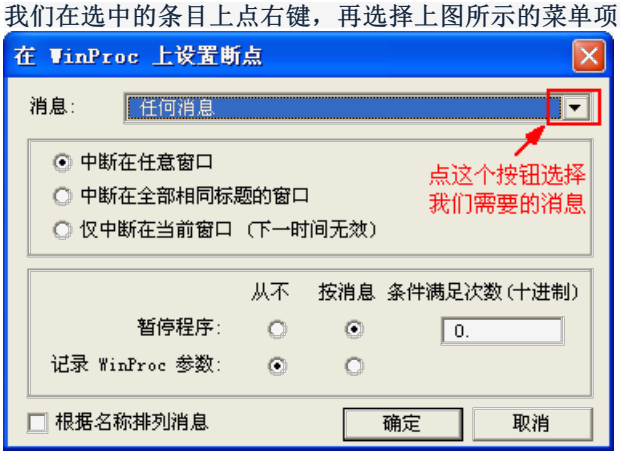
找了几十个不同语言编写的 crackme，发现只用消息断点的话有很多并不能真正到达我们要找的关键位置，想想还是把消息断点和 RUN 跟踪结合在一起讲，更有效一点。关于消息断点的更多内容大家可以参考 jingulong 兄的那篇《几种典型程序 Button处理代码的定位》的文章，堪称经典之作。今天仍然选择 crackmes.cjb.net 镜像打包中的一个名称为 cycle 的 crackme。按照惯例，我们先运行一下这个程序看看：



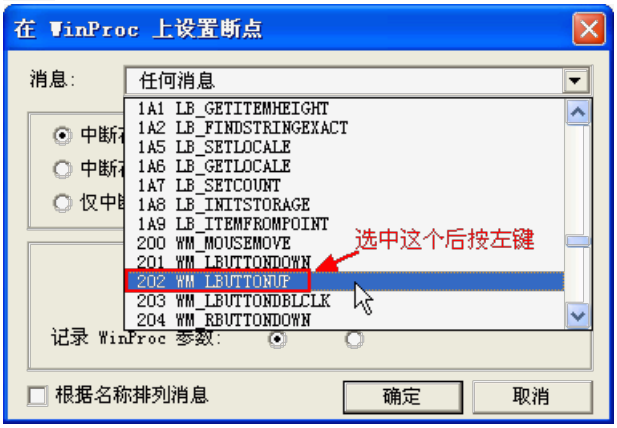
我们输入用户名 CCDebuger，序列号 78787878，点上面那个“Check”按钮，呵， 没反应！看来是要注册码正确才有动静。现在关掉这个 crackme，用 PEiD 查一下壳，原来是 MASM32 / TASM32 [Overlay]。启动 OllyDBG 载入这个程序，F9让它运行。这个程序按我们前面讲的采用字串参考或函数参考的方法都很容易断下来。但我们今天主要学习的是消息断点及 RUN 跟踪，就先用消息断点来断这个程序吧。在设消息断点前，有两个内容我们要简单了解一下：首先我们要了解的是消息。Windows 的中文翻译就是“窗口”，而 Windows 上面的应用程序也都是通过窗口来与用户交互的。现在就有一个问题，应用程序是如何知道用户作了什么样的操作的？这里就要用到消息了。Windows 是个基于消息的系统，它在应用程序开始执行后，为该程序创建一个“消息队列”，用来存放该程序可能创建的各种不同窗口的信息。比如你创建窗口、点击按钮、移动鼠标等等，都是通过消息来完成的。通俗的说，Windows 就像一个中间人，你要干什么事是先通知它，然后它才通过传递消息的方式通知应用程序作出相应的操作。说到这，又有个问题了，在 Windows 下有多个程序都在运行，那我点了某个按钮，或把某个窗口最大化，Windows 知道我是点的哪个吗？这里就要说到另一个内容：句柄（handle）了。句柄一般是个 32 位的数，表示一个对象。Windows 通过使用句柄来标识它代表的对象。比如你点击某个按钮，Windows 就是通过句柄来判断你是点击了那一个按钮，然后发送相应的消息通知程序。说完这些我们再回到我们调试的程序上来，你应该已经用 OllyDBG 把这个 crackme 载入并按 F9 键运行了吧？现在我们输入用户名“CCDebuger”，序列号“78787878”，先不要点那个“Check”按钮，我们来到 OllyDBG 中，点击菜单 查看->窗口（或者点击工具栏上那个“W”的图标），我们会看到以下内容：



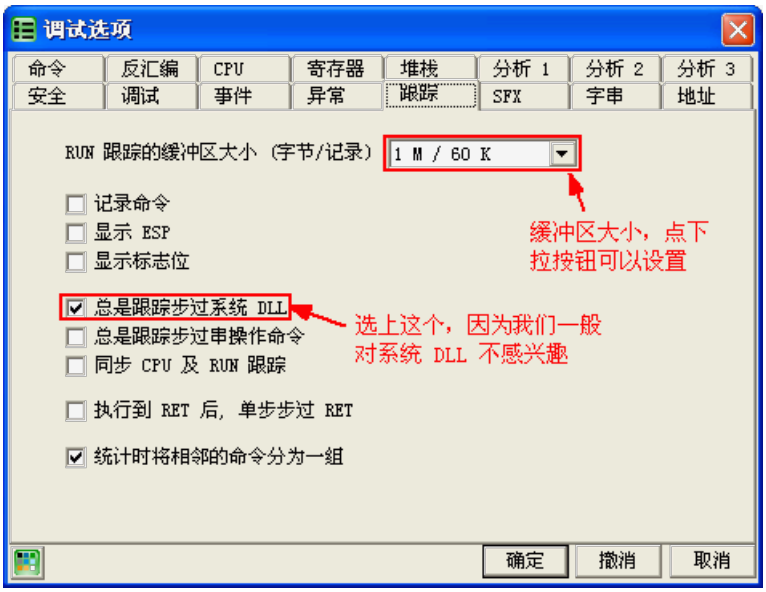
我们在选中的条目上点右键，再选择上图所示的菜单项，会来到下面这个窗口：



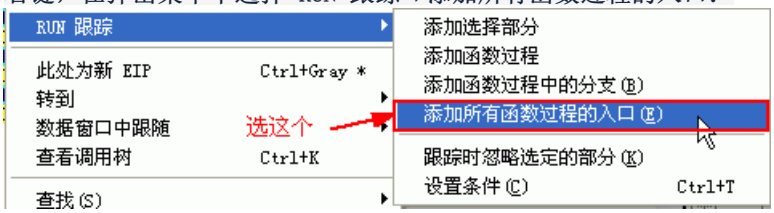
现在我们点击图上的那个下拉菜单，呵，原来里面的消息真不少。这么多消息我们选哪个呢？注册是个按钮，我们就在按下按钮再松开时让程序中断。查一下 MSDN，我们知道这个消息应该是 WM_LBUTTON_UP，看字面意思也可以知道是左键松开时的消息：



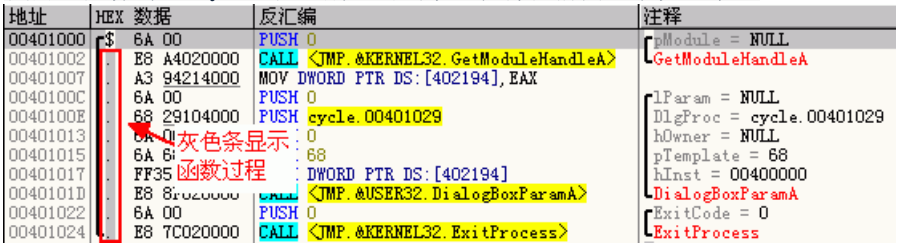
从下拉菜单中选中那个 202 WM_LBUTTON_UP，再按确定按钮，我们的消息断点就设好了。现在我们还要做一件事，就是把 RUN 跟踪打开。有人可能要问，这个 RUN 跟踪是干什么的？简单的说，RUN 跟踪就是把被调试程序执行过的指令保存下来，让你可以查看被调试程序运行期间干了哪些事。RUN 跟踪会把地址、寄存器的内容、消息以及已知的操作数记录到 RUN 跟踪缓冲区中，你可以通过查看 RUN 跟踪的记录来了解程序执行了那些指令。在这还要注意一个缓冲区大小的问题，如果执行的指令太多，缓冲区满了的话，就会自动丢弃前面老的记录。我们可以在调试选项->跟踪中设置：



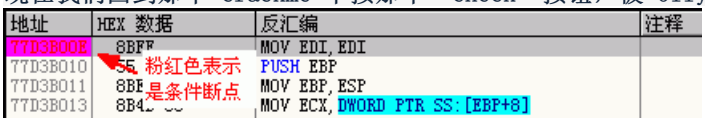
现在我们回到 OllyDBG 中，点击菜单调试->打开或清除 RUN 跟踪（第一次点这个菜单是打开 RUN 跟踪，在打开的情况下点击就是清除 RUN 跟踪的记录，对 RUN 跟踪熟悉时还可以设置条件），保证当前在我们调试的程序领空，在反汇编窗口中点击右键，在弹出菜单中选择 RUN 跟踪->添加所有函数过程的入口：



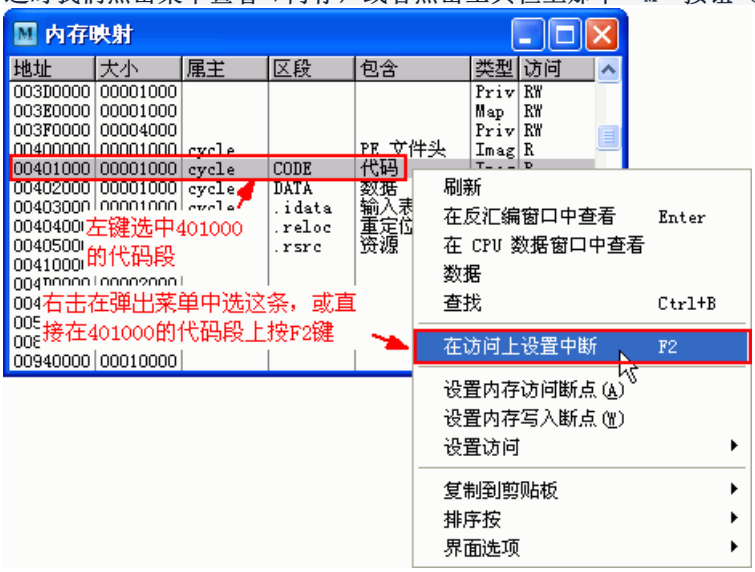
我们可以看到 OllyDBG 把识别出的函数过程都在前面加了灰色条：



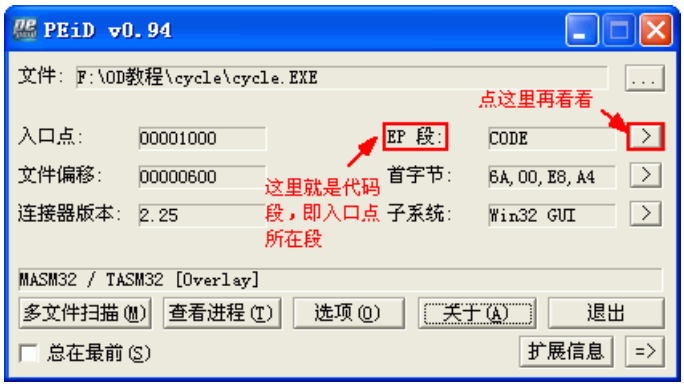
现在我们回到那个 crackme 中按那个“Check”按钮，被 OllyDBG 断下了：



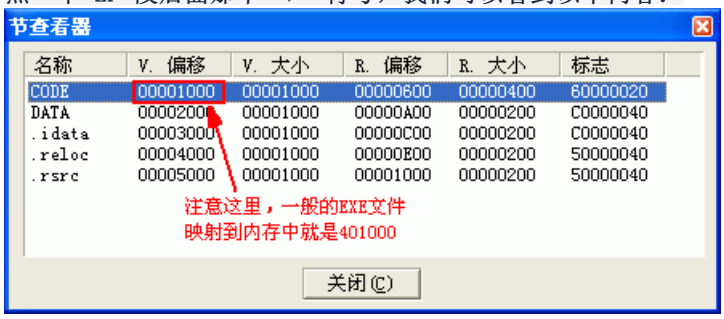
这时我们点击菜单查看->内存，或者点击工具栏上那个“M”按钮（也可以按组合键 ALT+M），来到内存映射窗口：



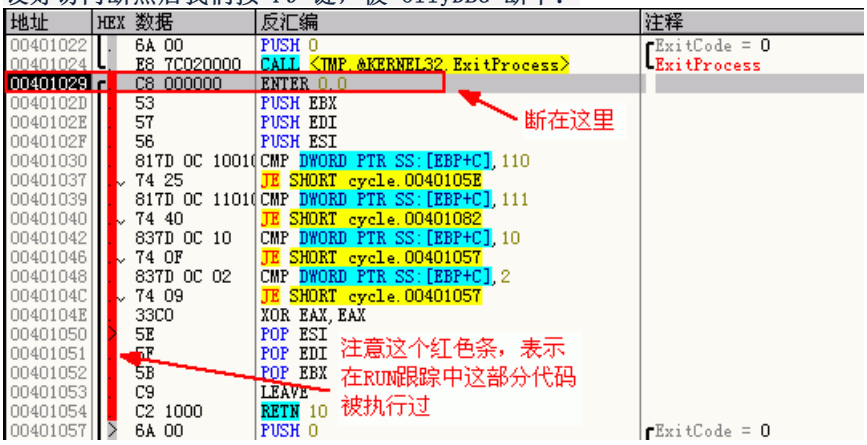
为什么在这里设访问断点，我也说一下。我们可以看一下常见的 PE 文件，没加过壳的用 PEiD 检测是这样：



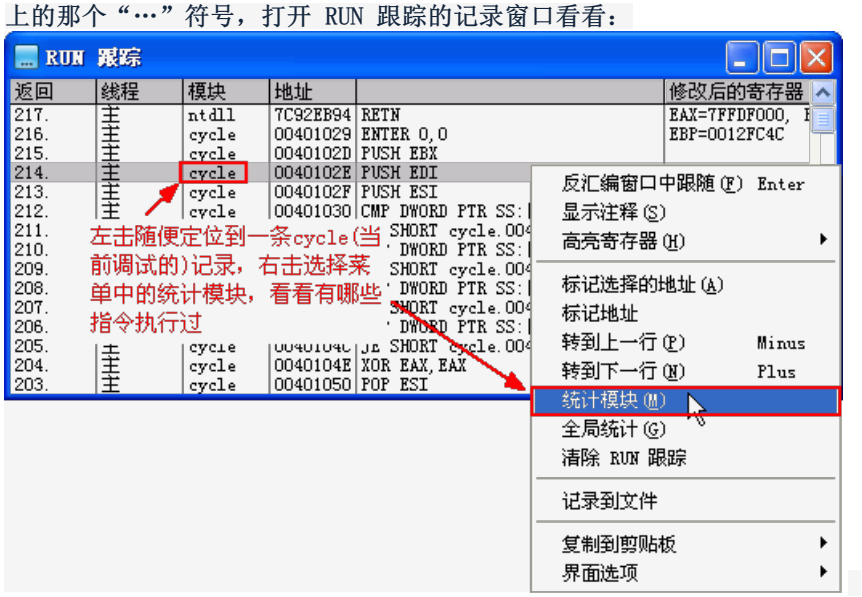
点一下 EP 段后面那个“>”符号，我们可以看到以下内容：



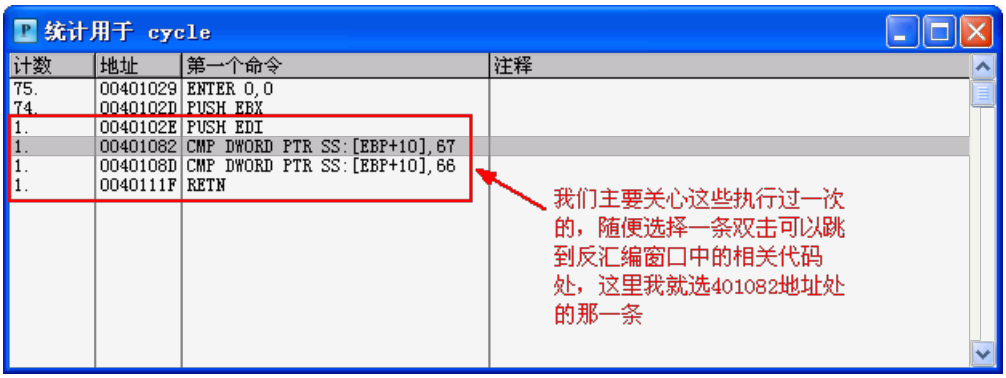
看完上面的图我们应该了解为什么在 401000 处的代码段下访问断点了，我们这里的意思就是在消息断点断下后，只要按 F9 键运行时执行到程序代码段的指令我们就中断，这样就可以回到程序领空了（当然在 401000 处所在的段不是绝对的，我们主要是要看程序的代码段在什么位置，其实在上面图中 OllyDBG 内存窗口的“包含”栏中我们就可以看得很清楚了）。设好访问断点后我们按 F9 键，被 OllyDBG 断下：



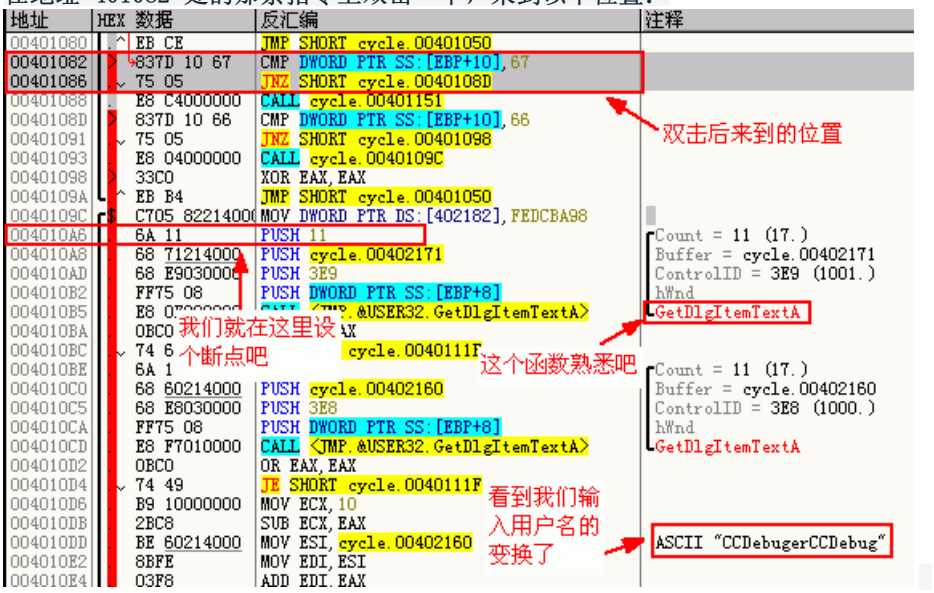
现在我们先不管，按 F9 键（或者按 CTR+F12 组合键跟踪步过）让程序运行，再点击菜单查看->RUN 跟踪，或者点击工具栏上的那个“…”符号，打开 RUN 跟踪的记录窗口看看：



我们现在再来看看统计的情况：



在地址 401082 处的那条指令上双击一下，来到以下位置：



现在我们在地址 4010A6 处的那条指令上按 F2，删除所有其它的断点，点菜单调试->关闭 RUN 跟踪，现在我们就可以开始分析了：

```
004010E2 |. 8BFE              MOV EDI,ESI                              ; 用户名送 EDI
004010E4 |. 03F8              ADD EDI,EAX
004010E6 |. FC                CLD
004010E7 |. F3:A4             REP MOVS BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]
004010E9 |. 33C9              XOR ECX,ECX                              ; 清零，设循环计数器
004010EB |. BE 71214000       MOV ESI,cycle.00402171                   ; 注册码送ESI
004010F0 |> 41                INC ECX
004010F1 |. AC                LODS BYTE PTR DS:[ESI]                   ; 取注册码的每个字符
004010F2 |. 0AC0              OR AL,AL                                 ; 判断是否为空
004010F4 |. 74 0A             JE SHORT cycle.00401100                  ; 没有则跳走
004010F6 |. 3C 7E             CMP AL,7E                                ; 判断字符是否为非ASCII字符
004010F8 |. 7F 06             JG SHORT cycle.00401100                  ; 非ASCII字符跳走
004010FA |. 3C 30             CMP AL,30                                ; 看是否小于30H，主要是判断是不是数字或字母等
004010FC |. 72 02             JB SHORT cycle.00401100                  ; 小于跳走
004010FE |.^ EB F0             JMP SHORT cycle.004010F0
00401100 |> 83F9 11           CMP ECX,11                               ; 比较注册码位数，必须为十进制17位
00401103 |. 75 1A             JNZ SHORT cycle.0040111F
00401105 |. E8 E7000000       CALL cycle.004011F1                      ; 关键，F7跟进去
0040110A |. B9 01FF0000       MOV ECX,0FF01
0040110F |. 51                PUSH ECX
00401110 |. E8 7B000000       CALL cycle.00401190                      ; 关键，跟进去
00401115 |. 83F9 01           CMP ECX,1
00401118 |. 74 06             JE SHORT cycle.00401120
0040111A |> E8 47000000       CALL cycle.00401166                      ; 注册失败对话框
0040111F |> C3                RETN
00401120 |> A1 68214000       MOV EAX,DWORD PTR DS:[402168]
00401125 |. 8B1D 6C214000     MOV EBX,DWORD PTR DS:[40216C]
0040112B |. 33C3              XOR EAX,EBX
0040112D |. 3305 82214000     XOR EAX,DWORD PTR DS:[402182]
00401133 |. 0D 40404040       OR EAX,40404040
00401138 |. 25 77777777       AND EAX,77777777
```

```
0040113D |. 3305 79214000     XOR EAX,DWORD PTR DS:[402179]
00401143 |. 3305 7D214000     XOR EAX,DWORD PTR DS:[40217D]
00401149 |.^ 75 CF            JNZ SHORT cycle.0040111A                    ; 这里跳走就完蛋
0040114B |. E8 2B000000       CALL cycle.0040117B                         ; 注册成功对话框
```

写到这准备跟踪算法时，才发现这个 crackme 还是挺复杂的，具体算法我就不写了，实在没那么多时间详细跟踪。有兴趣的可以跟一下，注册码是17位，用户名采用复制的方式扩展到 16 位，如我输入“CCDebuger”，扩展后就是“CCDebugerCCDebug”。大致是先取扩展后用户名的前 8 位和注册码的前 8 位，把用户名的前四位和后四位分别与注册码的前四位和后四位进行运算，算完后再把扩展后用户名的后 8 位和注册码的后 8 位分两部分，再与前面用户名和注册码的前 8 位计算后的值进行异或计算，最后结果等于 0 就成功。注册码的第 17 位我尚未发现有何用处。对于新手来说，可能这个 crackme 的难度大了一点。没关系，我们主要是学习 OllyDBG 的使用，方法掌握就可以了。

最后说明一下：
1、这个程序在设置了消息断点后可以省略在代码段上设访问断点那一步，直接打开 RUN 跟踪，消息断点断下后按 CTR+F12 组合键让程序执行，RUN 跟踪记录中就可以找到关键地方。
2、对于这个程序，你可以不设消息断点，在输入用户名和注册码后先不按那个“Check”按钮，直接打开 RUN 跟踪，添加“所有函数过程的入口”后再回到程序中点“Check”按钮，这时在 OllyDBG 中打开 RUN 跟踪记录同样可以找到关键位置。