第十一章:硬件断点与条件断点

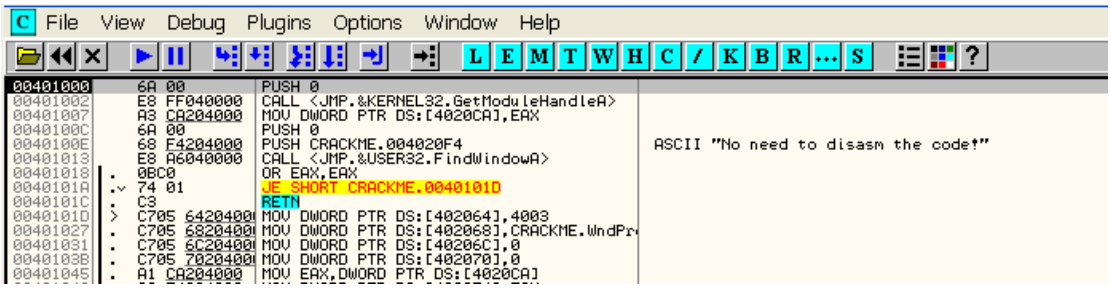
下面我们将把剩下的类型的断点介绍完,本章先介绍硬件断点和条件断点。

硬件断点

硬件断点(简称:HBP)是处理器的特性之一,它的工作原理我不是很了解,但是我们会用就行了,我们可以设置硬件断点使程序中断下来。

在 OD 中我们最多可以设置 4 个硬件断点,如果想设置第 5 个的话,你需要删除已经设置了的 4 个中的其中一个。

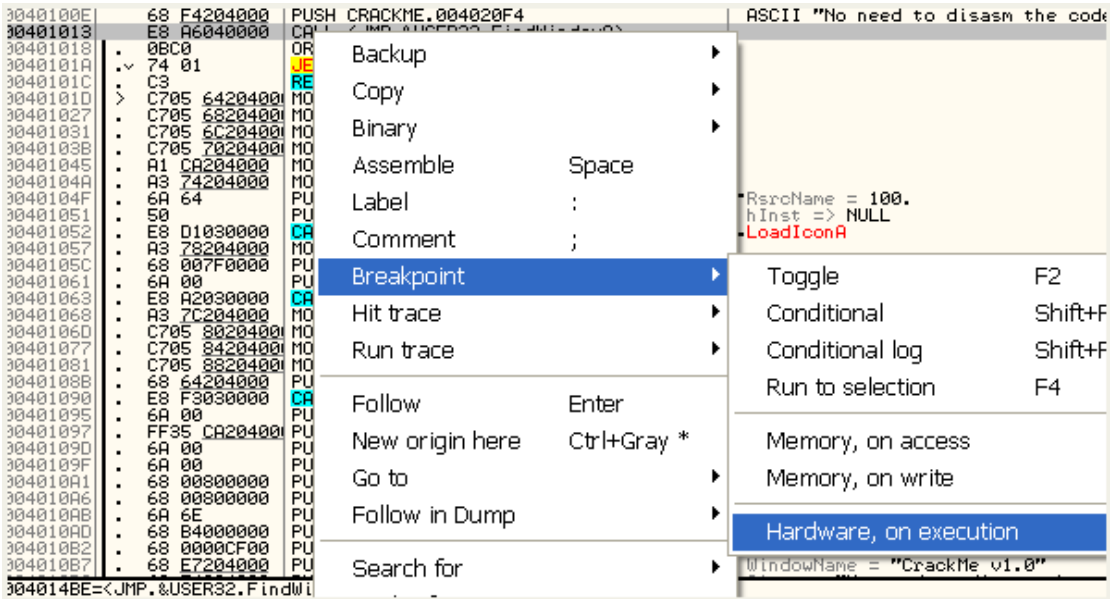
跟之前一样,我们还是拿 CrueHead'a 的 CrackMe 来做实验。



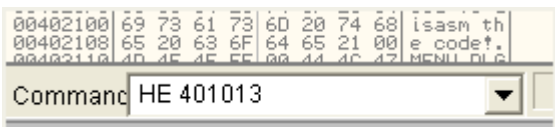
硬件断点分为:硬件执行断点(ON EXECUTION),硬件写入断点(ON WRITE),硬件访问断点(ON ACCESS)3 种。

硬件执行断点与普通的 CC 断点作用一样,但硬件执行断点并不会将指令首字节修改为 CC,所以更难检测。但是有些程序会使用一些技巧来清除硬件断点,应对方法我们会在后面的章节介绍。

如果你想在 401013 处设置硬件执行断点的话,请在 401013 这一行单击鼠标右键选择-Breakpoint-Hardware,on execution。

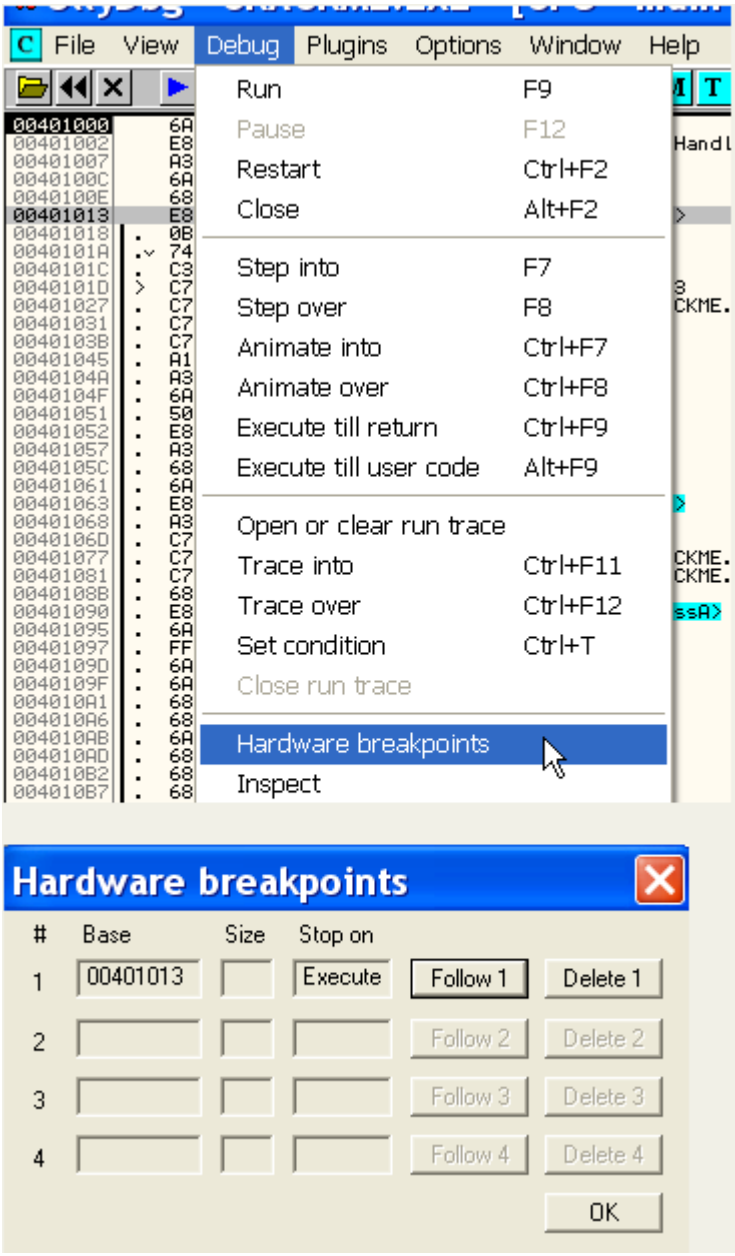


也可以在命令栏中输入:



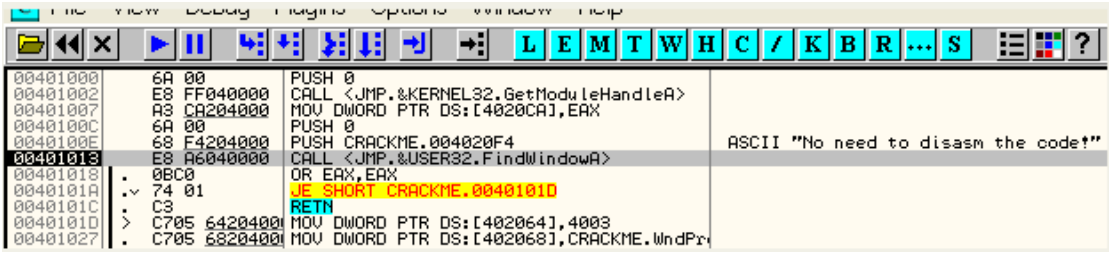
这样可以设置硬件执行断点。

OD 中有个特殊的窗口,通过它我们可以查看和管理硬件断点。我们选择菜单栏中的 Debug-Hardware breakpoints 就可以打开这个窗口。



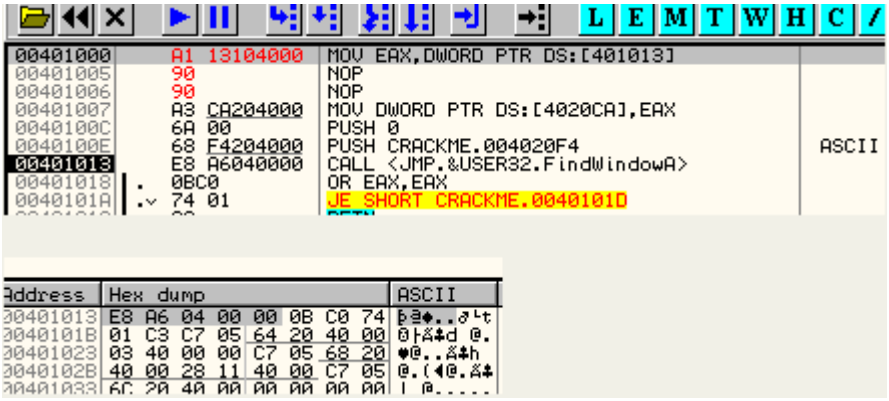
在硬件断点窗口中,如果我们单击 Follow 按钮,反汇编窗口中该硬件断点所对应的那一行指令就会灰色高亮显示。如果我们单击 Delete 按钮,那么相应的硬件断点就会被清除。

现在按 F9 键运行程序。

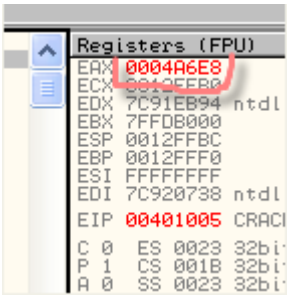


中断在 401013 处。

正如你所看到的,起到的效果跟普通的 CC 断点有点像,如果你像对 CC 断点做的测试-MOV EAX,DWORD PTR DS:[401013]一样的话,你会发现机器码并没有改变。

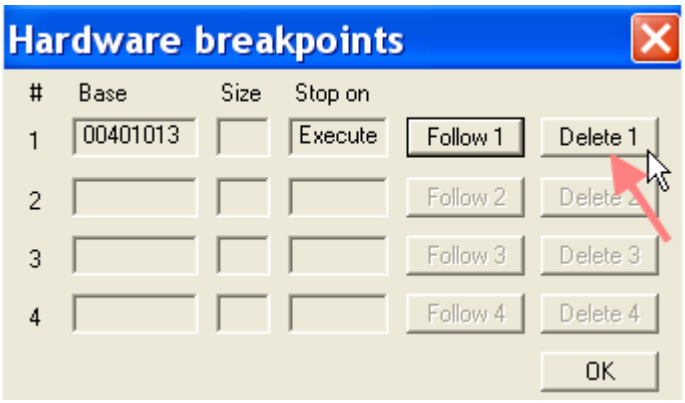


在 401000 这一行单击鼠标右键选择-New origin here 将 EIP 修改为 401000,接着按 F7 键单步。

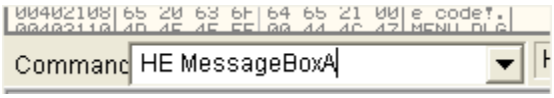


可以看到 EAX 的值为 0004A6E8,内存的形式为 E8 A60400,机器码并没有发生变化。

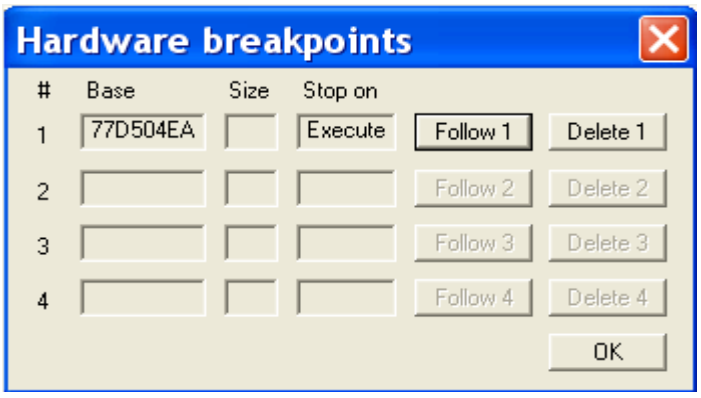
如果我们重启 OD,可以看到硬件断点依然存在。



单击 Delete 按钮将其删除,然后给 MessageBoxA API 函数设置一个硬件执行断点。



我们来看看硬件断点窗口。

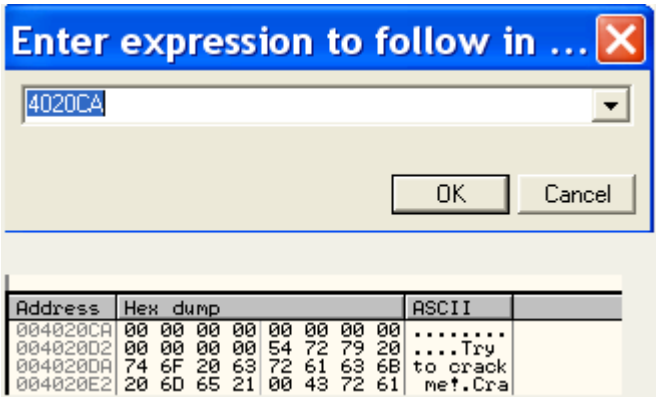


我们知道,如果按 F9 键将程序运行起来,然后输入用户名和序列号,跟之前的普通 CC 断点一样程序将断在 MessageBoxA 的第一条指令处,这里就不再赘述了。

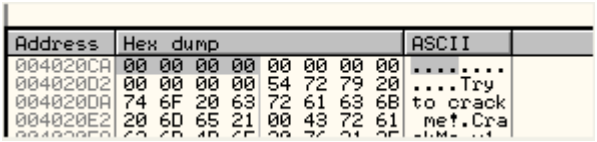
单个硬件写入断点或者硬件访问断点可以设置 1,2 或者 4 个字节的长度,不论我们选择的数据范围有多大,只有前 4 个字节会起作用。

现在我们清除所有设置的硬件断点,然后再 4020CA 处设置一个硬件访问断点。

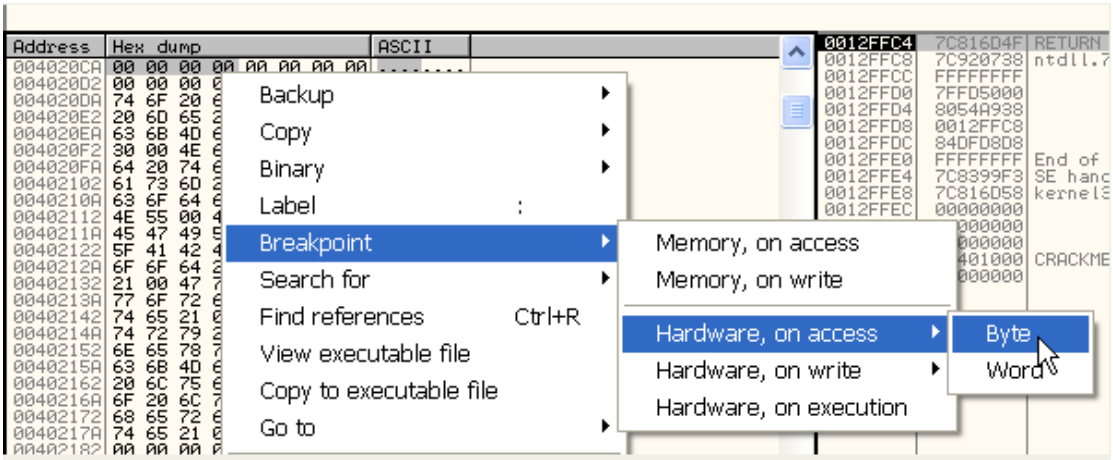
我们在数据窗口中转到 4020CA 这个地址处。



注意前 4 个字节。

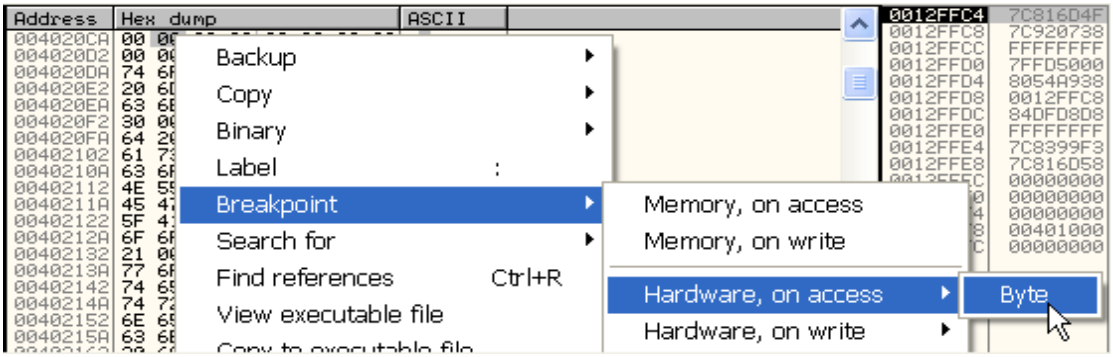


单击鼠标右键。

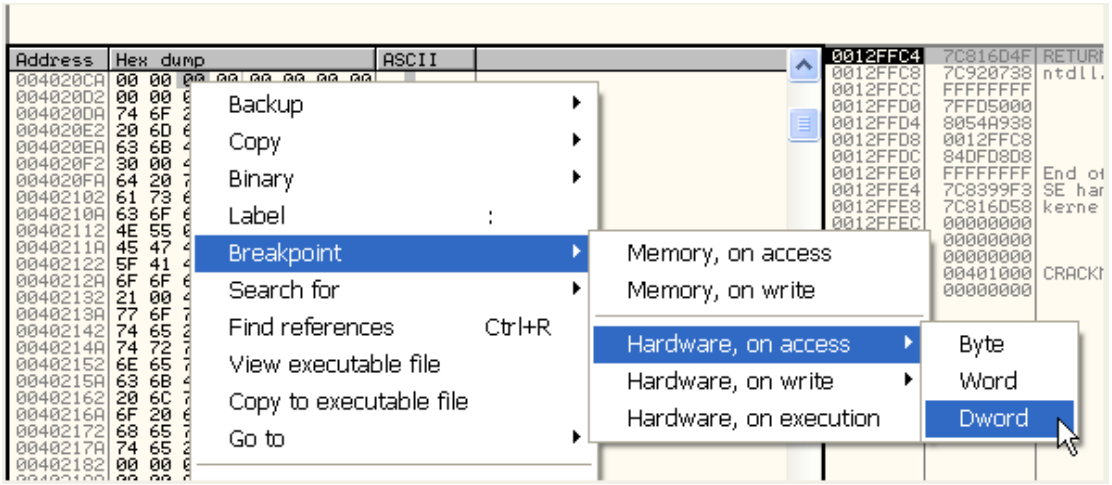


可以看到,对于我们标记的区域,硬件访问断点可供我们选择的长度只有 1 字节和 2 字节,由于 4 字节的只针对 4 的倍数的地址,当前地址并不是 4 的倍数,所以没有该菜单项。

接下来我们选择后面的 1 个字节,单击鼠标右键选择-Hardware,on access,可以看到只有 1 个字节的选项。



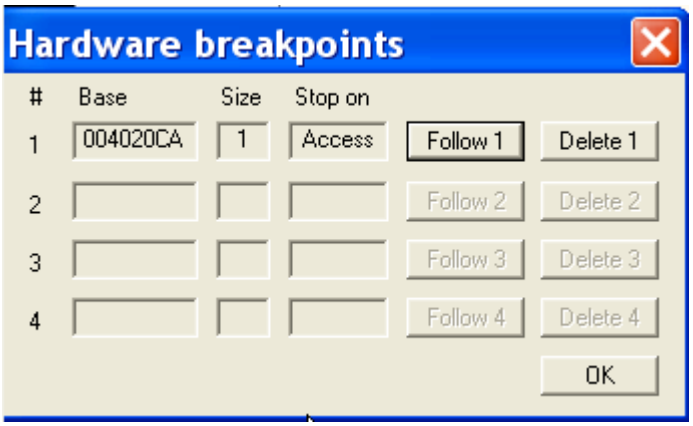
我们接着选中后面一个字节,这个时候出现了 4 个字节(Dword)的选项。



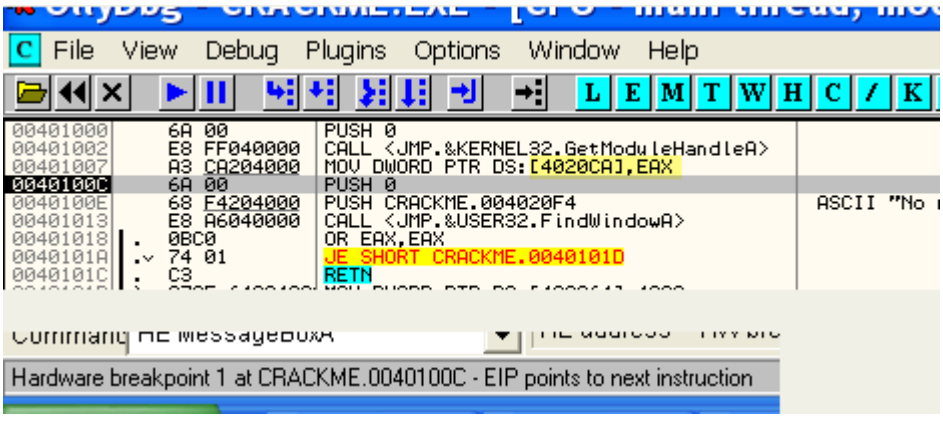
如果我们想在读取或者写入 4020CA 地址处的内容的时候断下来的话,我们给该地址设置 1 个字节的硬件访问断点即可。

返回到 4020CA 处,单击鼠标右键选择-Hardware,on access-Byte。

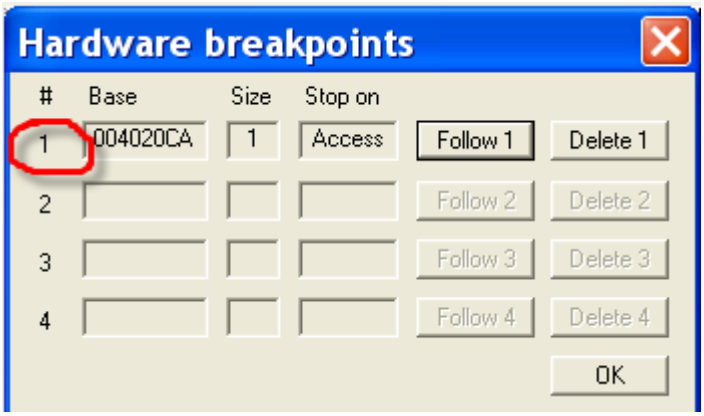
我们可以看到硬件断点窗口中显示的硬件访问断点长度为 1 个字节。



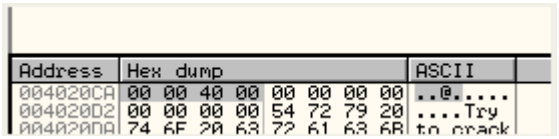
F9 键运行。



OD 提示命中了 1 号硬件断点。



我们可以看到程序断在了触发硬件访问断点或者硬件写入断点的下一条指令处,所以请记住硬件访问/写入断点是断在触发硬件断点的下一条指令处。



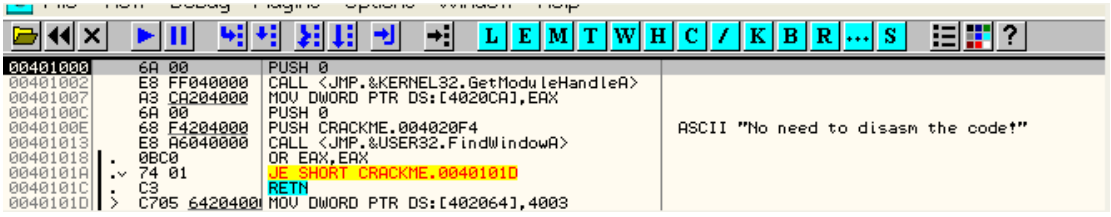
正如你所看到的,跟内存断点不一样,内存断点会断在触发断点指令处,也就是 401007 处。

而硬件访问/断点断在了触发硬件断点的下一条指令 40100C 处。

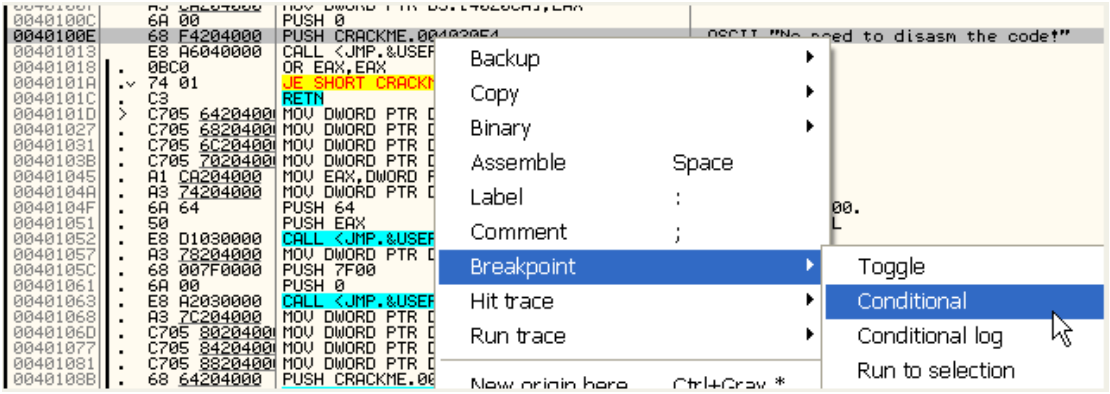
条件断点

条件断点实际上就是普通的 CC 断点,只不过该断点的触发需要满足设置的条件,如果满足设置的条件,那么程序就会中断下来,如果不满足条件的话,就和没有设置 CC 断点差不多。

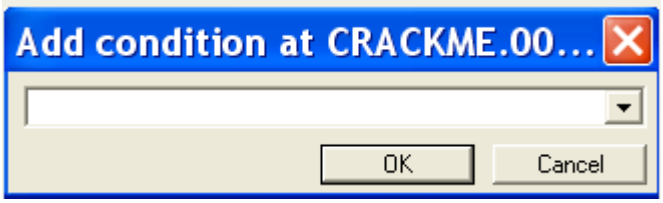
我们来看一个例子:



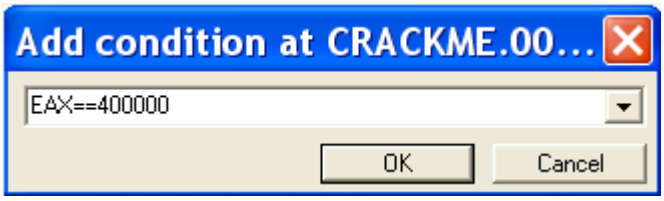
重启 OD 并清除我们之前设置的所有硬件断点,我们在 40100E 处设置一个条件断点,在 40100E 处单击鼠标右键选择-Breakpoint-Conditional。



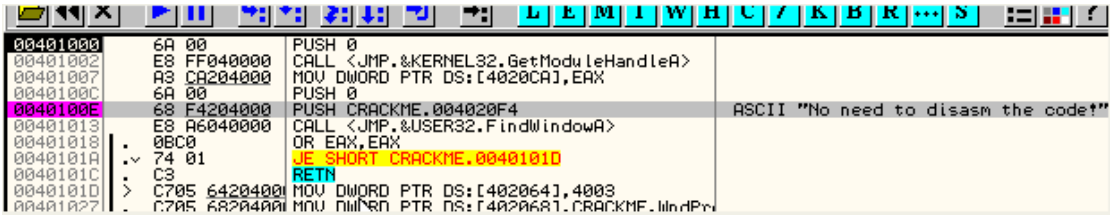
就会弹出设置条件的窗口。



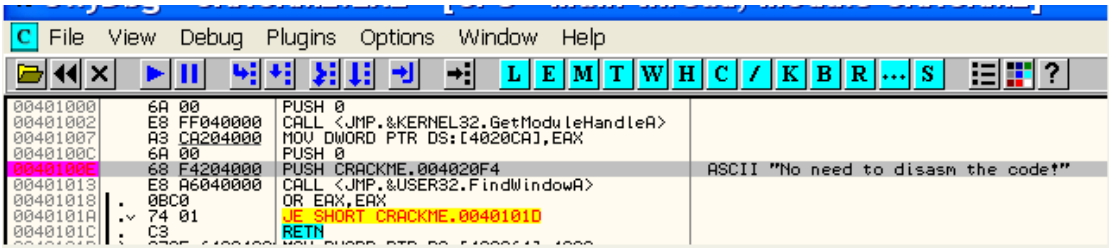
举个例子,如果你想当前 EAX 等于 400000 的时候,程序中断下来,那么条件应该写成:“EAX == 400000”。



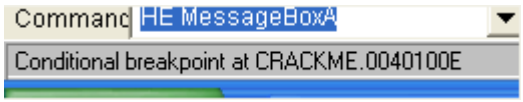
OD 的帮助文档中介绍了条件断点的设置可以使用的符号以及条件怎么书写。



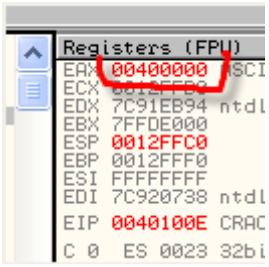
我们可以看到设置了条件断点的语句地址显示的是粉红色。按 F9 键运行。



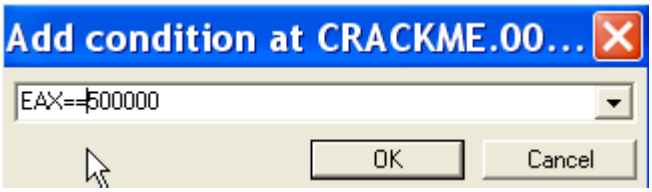
我们可以看到,OD 状态栏显示断下来了。



我们可以看到 EAX 等于 400000,满足条件,所以断了下来。



再次重启 OD,清除设置的条件断点,然后设置一个新的条件断点,条件为:EAX==500000。

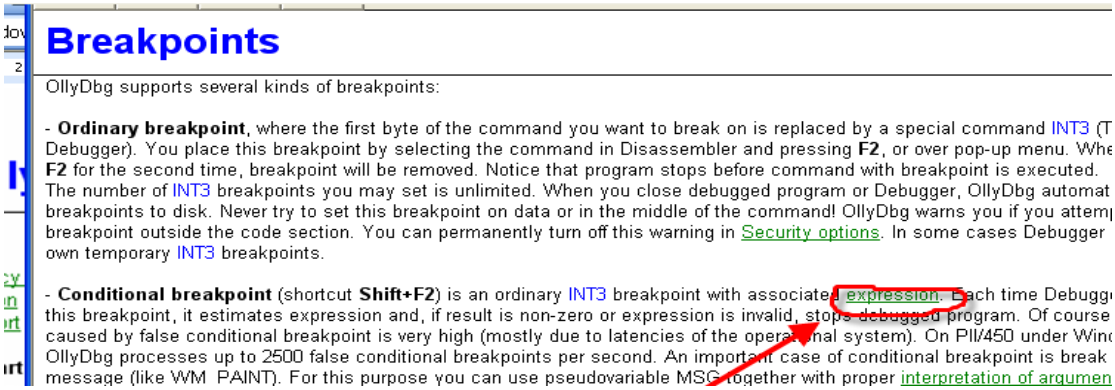


F9 键运行。

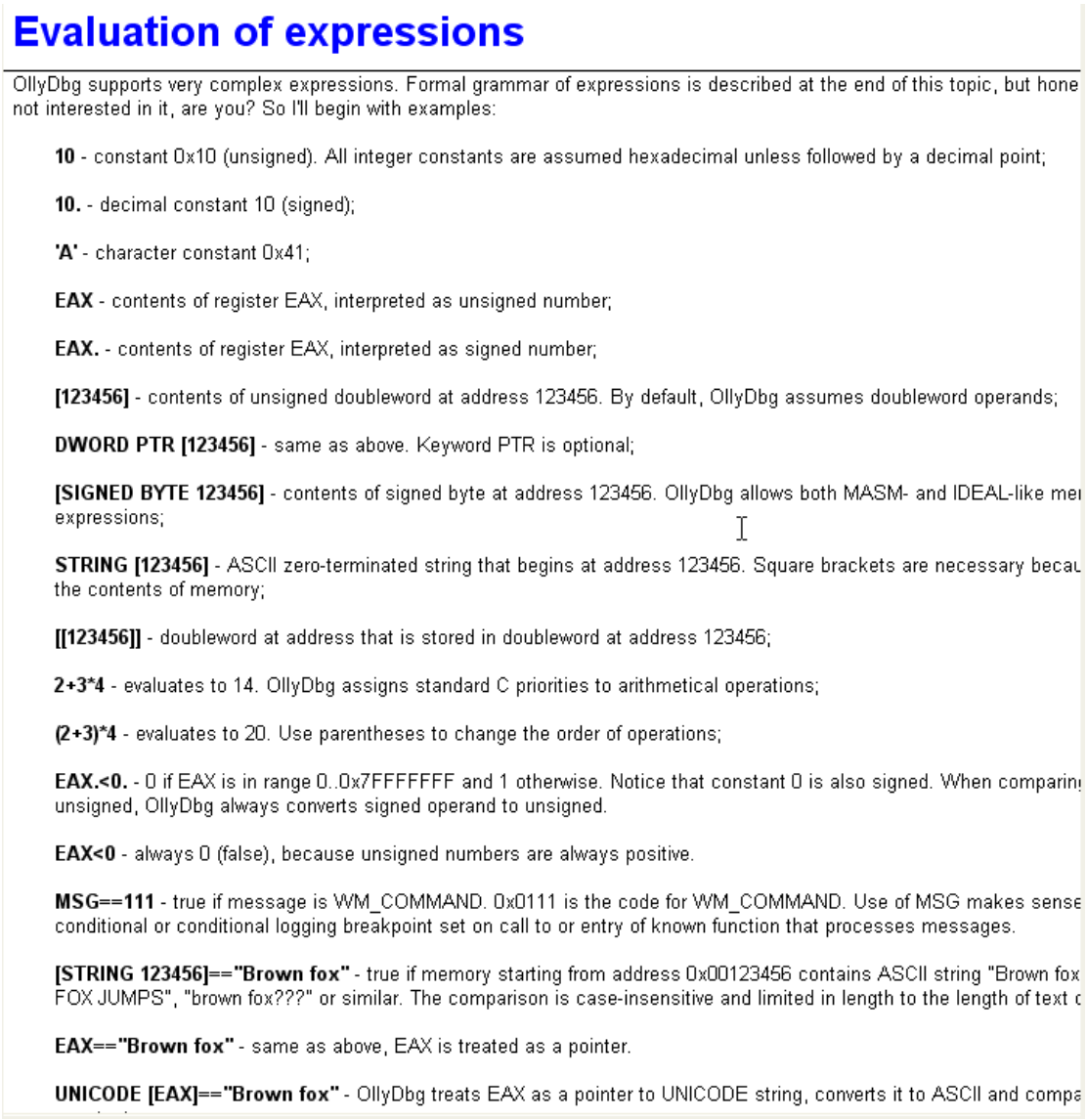
可以看到 CrackMe 运行起来了,并没有中断下来,因为 EAX 等于 400000,条件不满足。

我们在主菜单项中选择-Help-Contents。

选择超链接 Breakpoints,紧接着选择超链接 expression。



可以看到 OD 帮助文档中显示了很多条件表达式以及相应的例子。



Evaluation of expressions

OllyDbg supports very complex expressions. Formal grammar of expressions is described at the end of this topic, but hone not interested in it, are you? So I'll begin with examples:

10 - constant 0x10 (unsigned). All integer constants are assumed hexadecimal unless followed by a decimal point;

10. - decimal constant 10 (signed);

'A' - character constant 0x41;

EAX - contents of register EAX, interpreted as unsigned number;

EAX. - contents of register EAX, interpreted as signed number;

[123456] - contents of unsigned doubleword at address 123456. By default, OllyDbg assumes doubleword operands;

DWORD PTR [123456] - same as above. Keyword PTR is optional;

[SIGNED BYTE 123456] - contents of signed byte at address 123456. OllyDbg allows both MASM- and IDEAL-like me expressions;

STRING [123456] - ASCII zero-terminated string that begins at address 123456. Square brackets are necessary becau the contents of memory;

[[123456]] - doubleword at address that is stored in doubleword at address 123456;

2+3*4 - evaluates to 14. OllyDbg assigns standard C priorities to arithmetical operations;

(2+3)*4 - evaluates to 20. Use parentheses to change the order of operations;

EAX.<0. - 0 if EAX is in range 0..0x7FFFFFFF and 1 otherwise. Notice that constant 0 is also signed. When comparin unsigned, OllyDbg always converts signed operand to unsigned.

EAX<0 - always 0 (false), because unsigned numbers are always positive.

MSG==111 - true if message is WM_COMMAND. 0x0111 is the code for WM_COMMAND. Use of MSG makes sense conditional or conditional logging breakpoint set on call to or entry of known function that processes messages.

[STRING 123456]=="Brown fox" - true if memory starting from address 0x00123456 contains ASCII string "Brown fox FOX JUMPS", "brown fox???" or similar. The comparison is case-insensitive and limited in length to the length of text c

EAX=="Brown fox" - same as above, EAX is treated as a pointer.

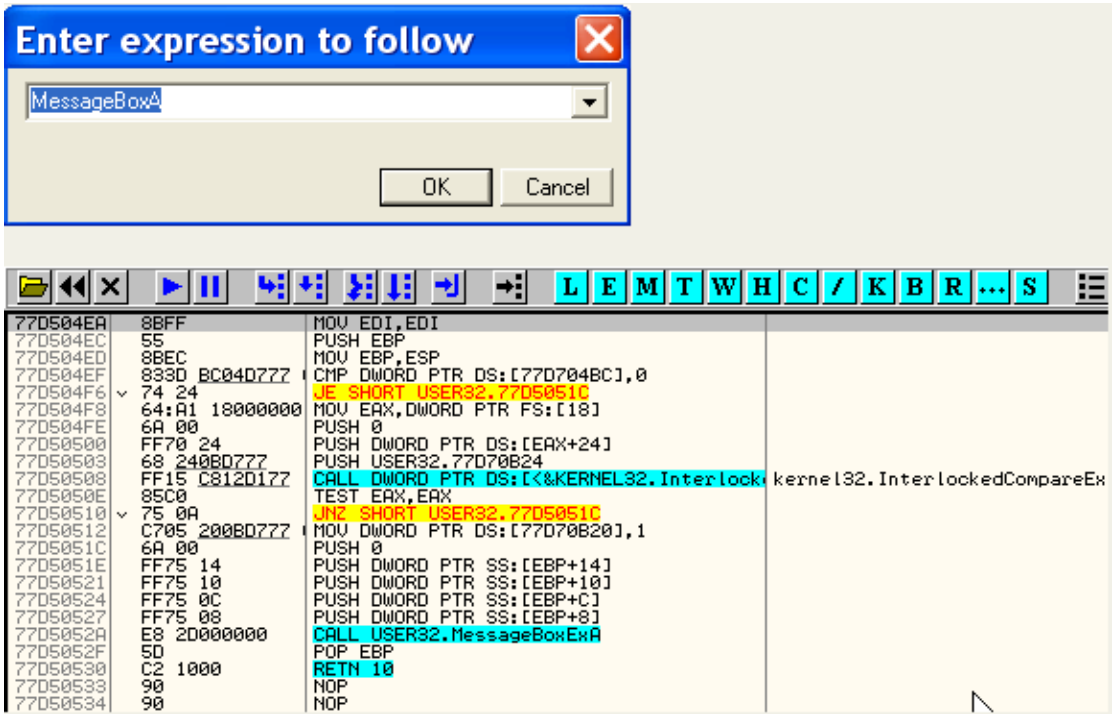
UNICODE [EAX]=="Brown fox" - OllyDbg treats EAX as a pointer to UNICODE string, converts it to ASCII and compa

条件记录断点

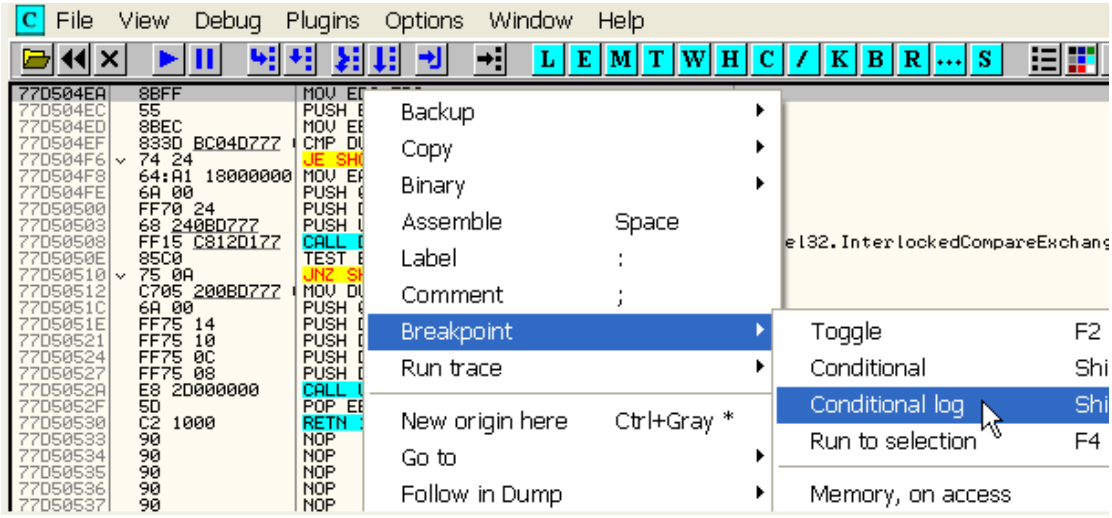
条件记录断点跟刚刚介绍的条件断点差不多,区别在于,我们可以通过设置该断点来记录下设置的条件的精确值。我们举个例子,我们给一个 API 函数设置条件记录断点,程序中有很多地方调用了这个 API 函数,通过该条件记录断点我们可以精确的记录程序中每处调用该 API 函数传递给它的内容。

我们删除之前设置的条件断点,并重启 OD。

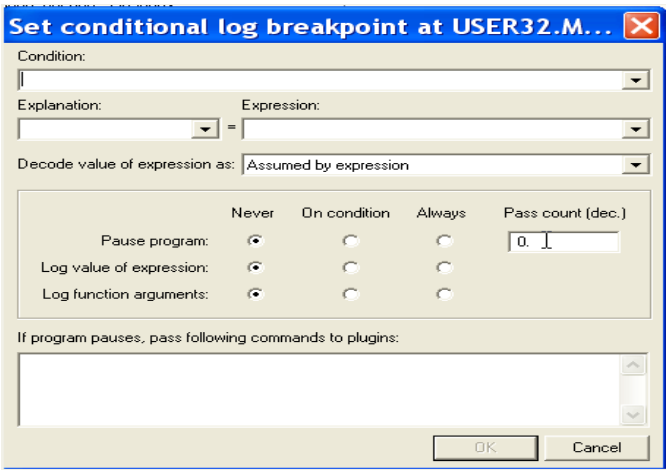
在反汇编窗口中单击鼠标右键选择-Goto-Expression 转到 MessageBoxA 函数的入口处。



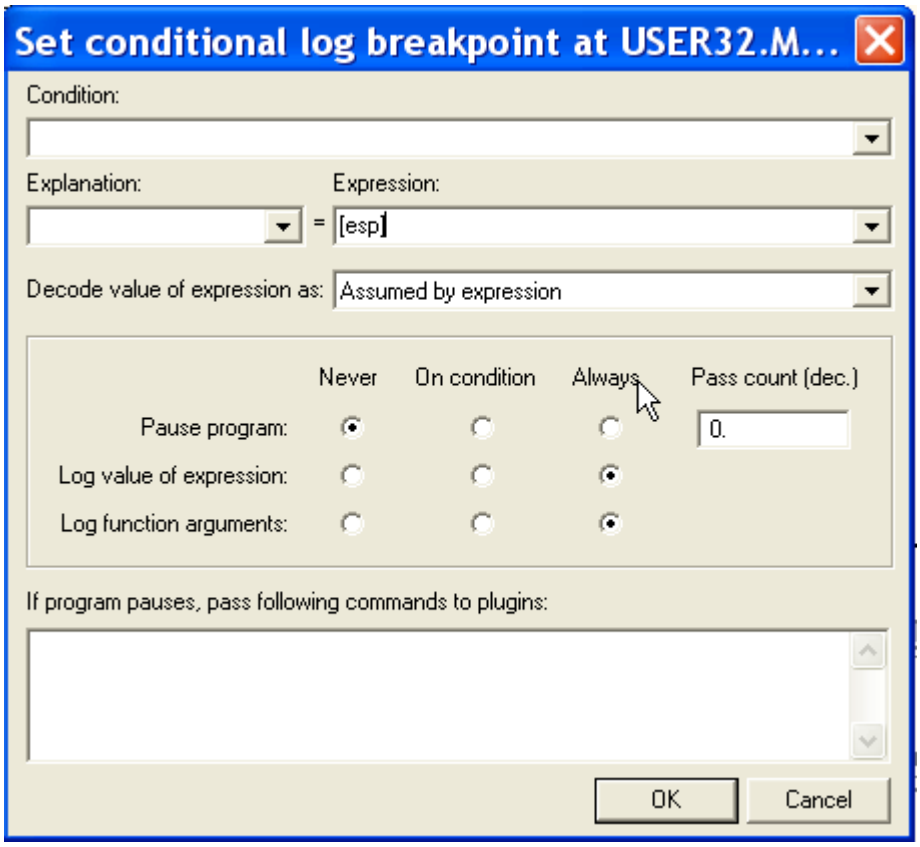
单击鼠标右键选择-Breakpoint-Conditional log。



弹出一个窗口,有很多选项。

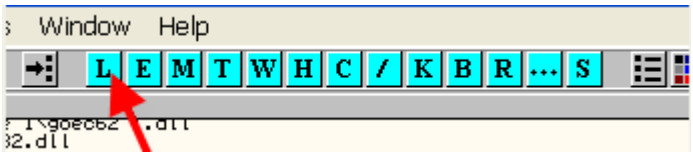


这里我们只是记录下我们关心的数据,并不让程序中断下来。

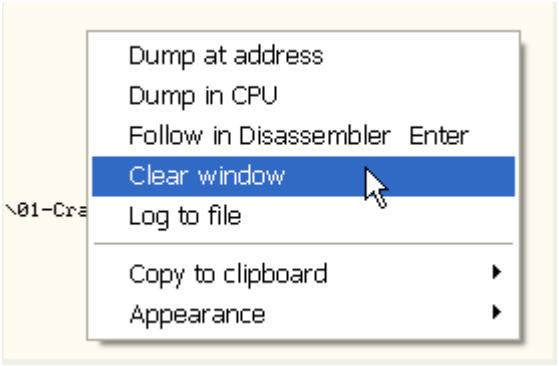


既然我们不想程序中断下来,那么 Condition(条件)编辑框这一栏我们就不填,Pause program(中断程序)这个单选按钮选着 Never(不中断)即可。Expression(表达式)这个编辑框我们填写[ESP],我们知道在 API 函数的入口处,栈顶存放的是函数的返回地址。接着,Log value of expression(记录表达式的值)这个单选按钮我们选择 Always(总是),即总是记录表达式的值,也就是[ESP]的值。Log function arguments(记录函数参数)这个单选按钮我们也选择 Always(总是),即记录函数的参数个数/参数内容。

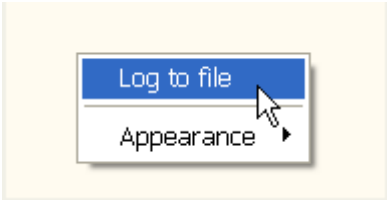
我们单击工具栏中【L】按钮打开日志窗口。



我们单击鼠标右键选择-Clear window 清空日志窗口。



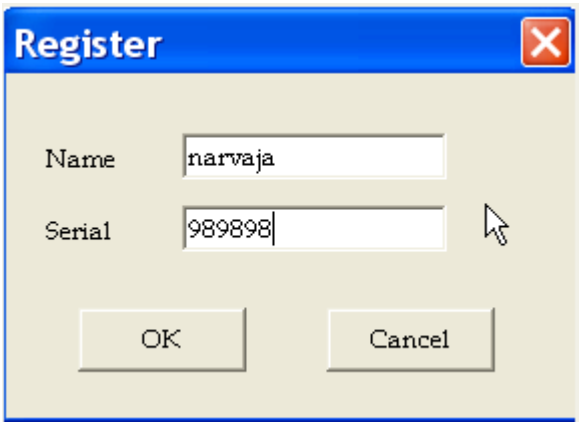
如果需要想将日志保存到文件的话,我们选择 Log to file 即可。



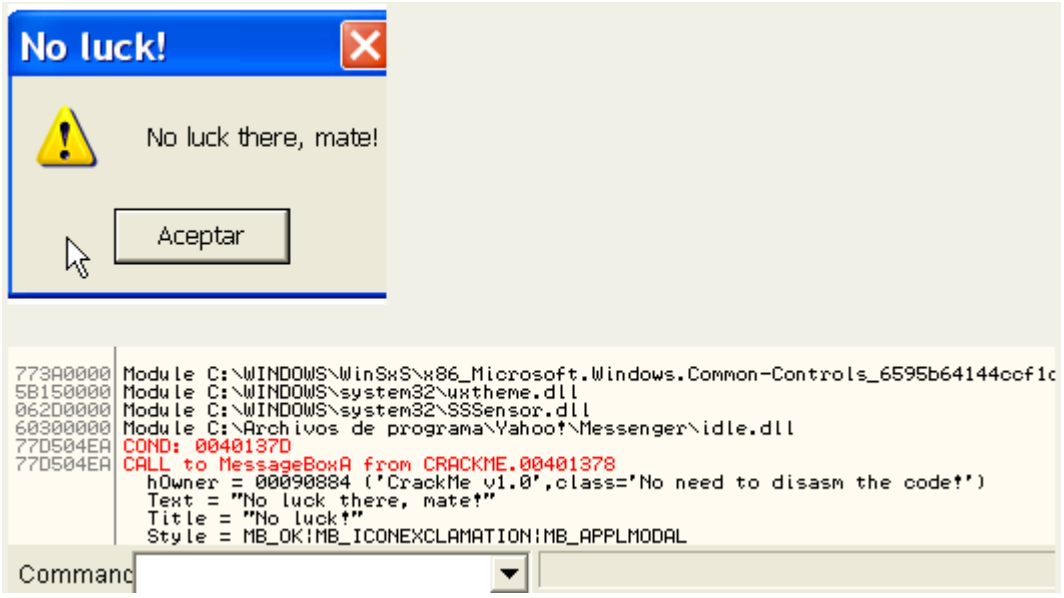
我们按 F9 键运行程序。

我们注意到日志窗口什么也没有记录,因为 MessageBoxA 并没有被调用。

我们打开注册窗口,输入指定的用户名和序列号。



单击 OK。

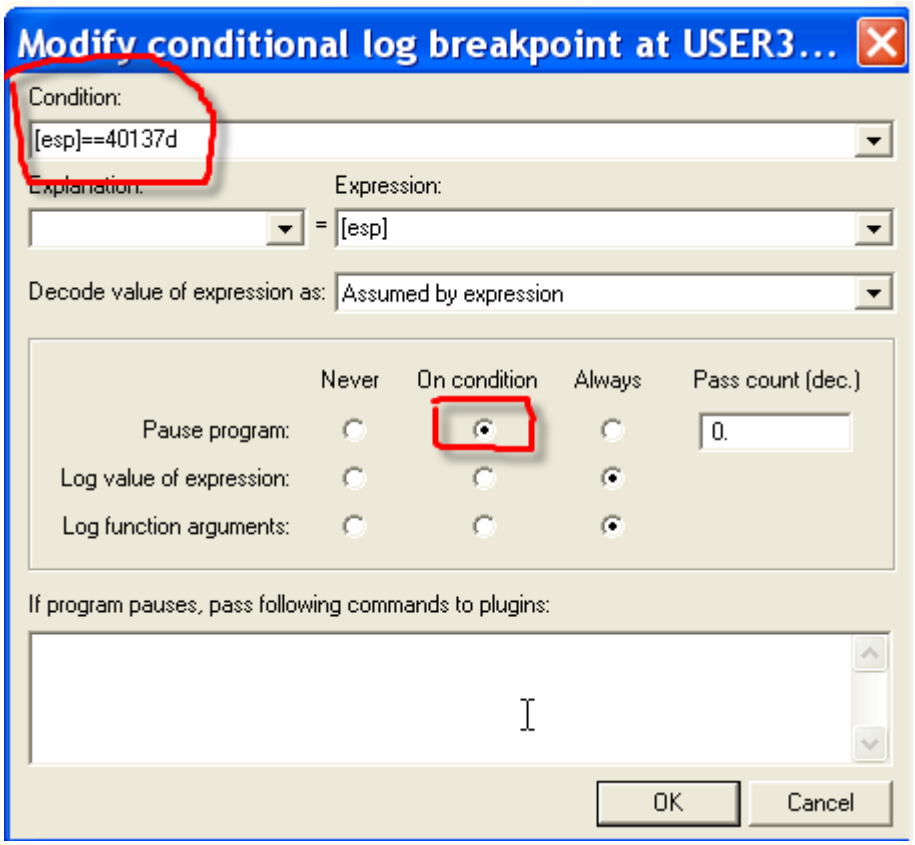


我们可以看到日志窗口显示有关 MessageBoxA 函数的信息,栈顶内容为 40137D 即函数的返回地址,接下来是 MessageBoxA 的参数,依次是 hOwner(所属窗口句柄),Text(文本),Title(标题),(Sytle)显示风格。

这里,只显示了一个函数调用的信息,如果某个 API 被调用了 100 次的话,你可以将日志保存到文本文件中。如果我们只需要记录某些函数调用的信息的话,我们可以设置相应的条件。

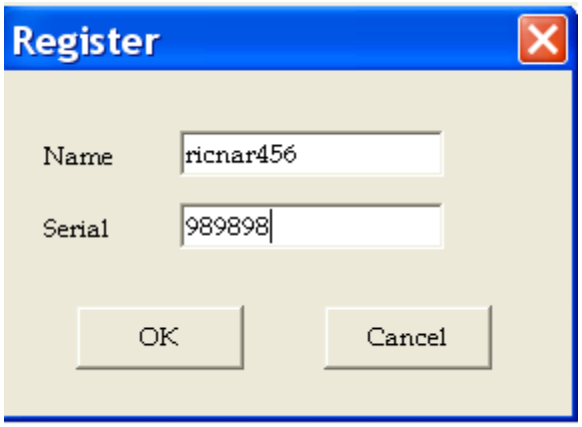
如果程序中有大约 100 处调用了该 API 函数,你可以指定相应的条件,比如说返回地址。设置条件为[ESP] == 40137D,那么这 100 个函数调用中只有返回地址为 40137D 的会被记录下来。

我们重启 CrackMe,定位到 MessageBoxA,并设置条件记录断点如下:

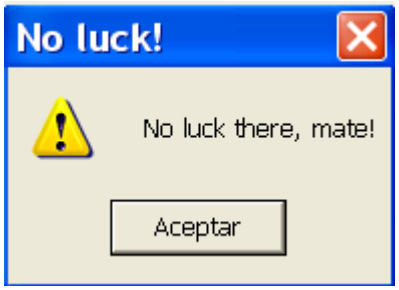


Pause program 这个单选按钮我们选择 On condition(满足条件时中断下来),即当[ESP] == 40137D 时中断下来。

运行程序,打开注册窗口输入用户名 ricnar456 以及序列号 989898。

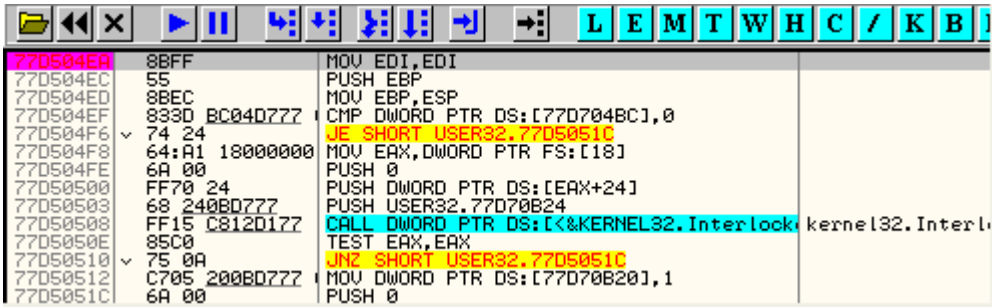


当我们输入包含数字的用户名的时候,会弹出两个错误消息框。现在单击 OK。



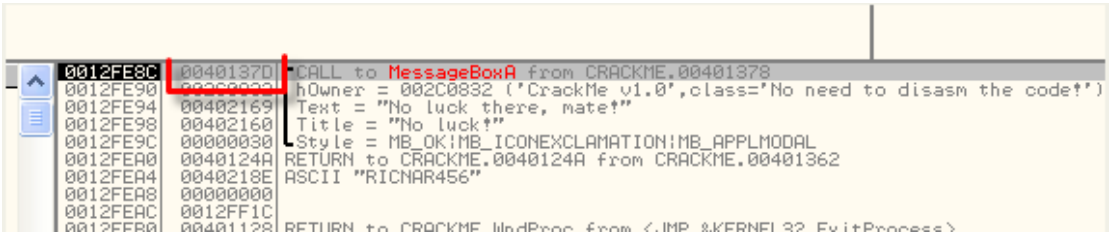
可以看到第一个 MesasgeBoxA 并没有中断下来。日志中显示的返回地址为 4013C1,并不满足设置的条件,所以不会中断下来。

我们按下 OK 按钮。

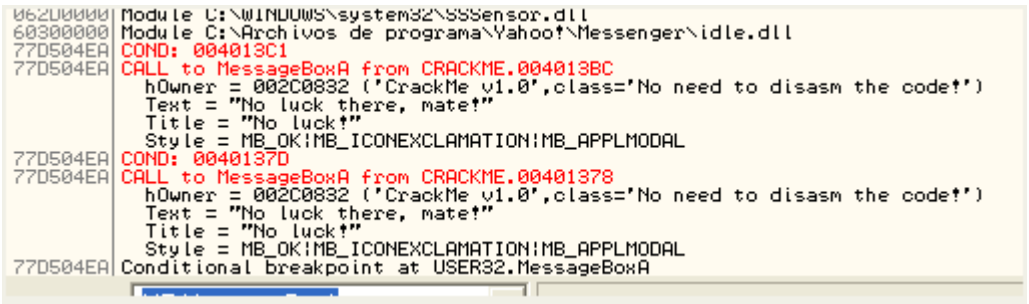


这是第二次调用 MessageBoxA,这里返回地址满足条件,所以中断下来了。

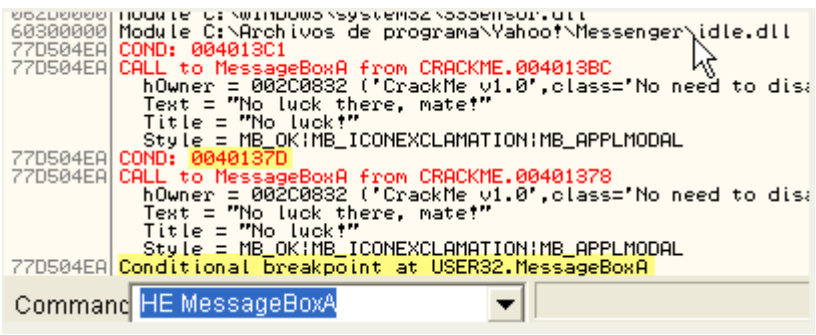
我们看到栈顶的内容是 40137D,满足条件记录断点设定的条件,所以中断下来了。



两次调用 MessageBoxA 的信息,OD 都记录下来了。



只有第二次返回地址为 40137D,断了下来。



好了,以上的内容够大家练习一段时间了。下一章将介绍消息断点。