

使用 OllyDbg 从零开始 Cracking

第一章

(翻译: BGCoder)

《使用 OllyDbg 从零开始 Cracking》教程的目的是为那些想精通 Cracking (译注 1) 艺术的人提供必要的基础知识。有了这些基础知识, 你就可以阅读和理解更高级的教程, 例如《INTRODUCCIÓN AL CRACKING》(译注-2), 这个系列教程现在仍在不断加入新的内容。

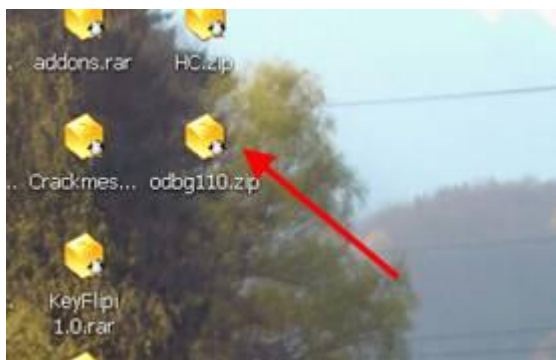
写这个教程想法的产生, 是由于意识到了《INTRODUCCIÓN AL CRACKING》中的很多教程对于新手来说太过于复杂。而新手们还未能达到课程所需的能力要求。多方面的原因导致这些新人难以继续学习。因此, 《使用 OllyDbg 从零开始 Cracking》的目标并不是取代来自《INTRODUCCIÓN AL CRACKING》中的教程(它里面的教程数量已逾 500), 而是为学完这个课程后的读者打下一个良好的基础, 以便去学习阅读日益复杂的资料。它的主要目的是减少课程数量, 提供必要的基础知识, 从而使新的 Cracker 对更复杂的文章资料能够深入的理解。

为什么使用 OllyDbg?

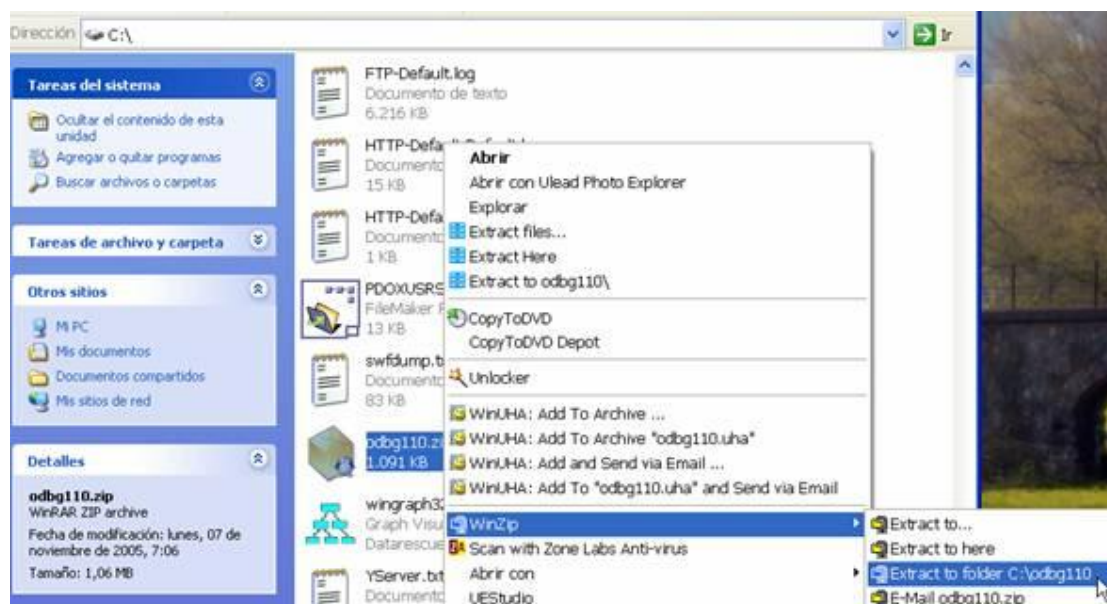
我们不会去探讨 SoftICE 和 OllyDbg 之间的对抗, 我认为即使是 SoftICE 的狂热追随者也可以很容易的入手 OllyDbg。而 OllyDbg 拥有更多的资料且更容易学习。我们需要从 OllyDbg 这扇大门进入 Cracking 的世界。以后可能会在必要的时候换成其它调试器, 但变化的只是它们的使用方法, 不变的是它们的本质。

让我们从头开始。

首先, 需要装备我们以后将主要使用的工具 (OllyDbg), 点此下载 (译注 3)。



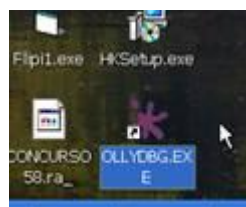
下载成功后, 将其解压到你容易访问的硬盘文件夹内, 一个好的办法是在 C 盘创建一个目录, 尽管它在哪里都可以运行, 在这里我选择了 C: /。



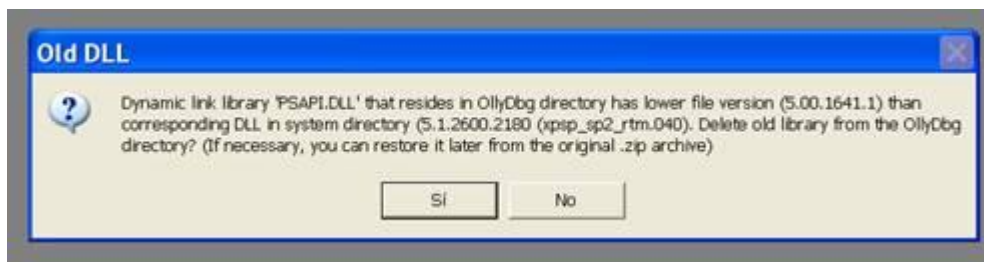
解压后，进入你创建的文件夹内可以看到：



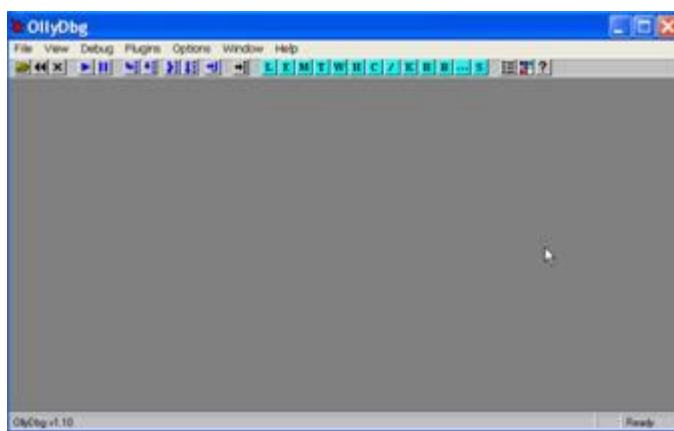
它的可执行文件是 OLLYDBG.exe，为了方便，我们可以将其快捷方式发送到桌面。



好的，启动软件，点击 OllyDbg 图标。



上图的消息框显示，PSAPI.DLL，一个库文件旧于我们系统上的同名 DLL 文件。如果你点击 Yes，那么这个旧文件就会从你的文件夹内消失，然后使用系统自身的。尽管我看不出两个文件有什么不同，但是最好选择随软件包自带的，所以永远要点击 No（译注 4）。

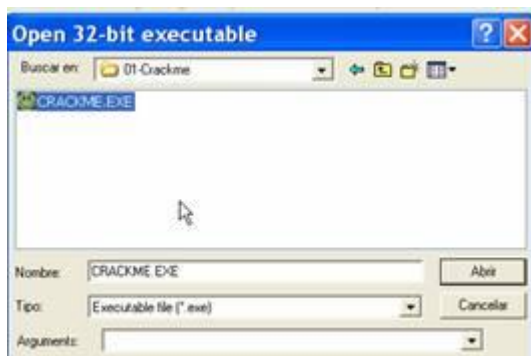


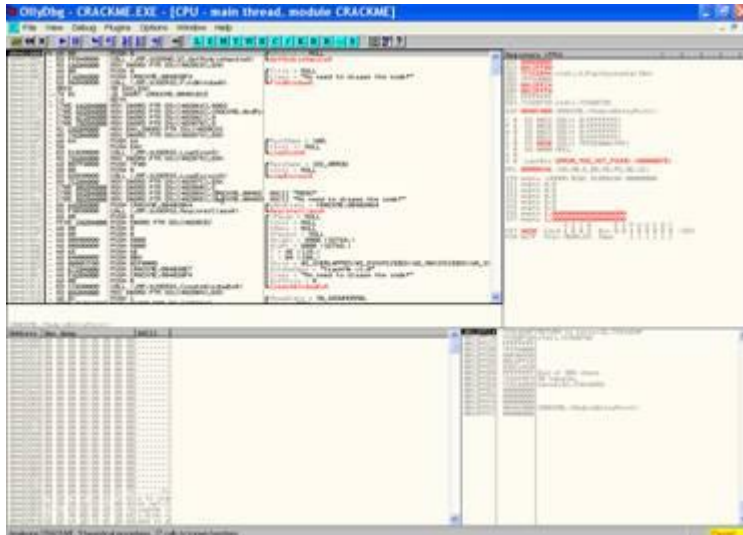
这是纯净的 OllyDbg（译注：未加载插件非修改版），为了逐渐熟悉 OllyDbg，我们打开的第一个程序来自著名的 CrueHead'a 的 CrackMe，这个文件随本文附带。

要在 OllyDbg 中打开文件，从 File -> Open 或者点击图标。

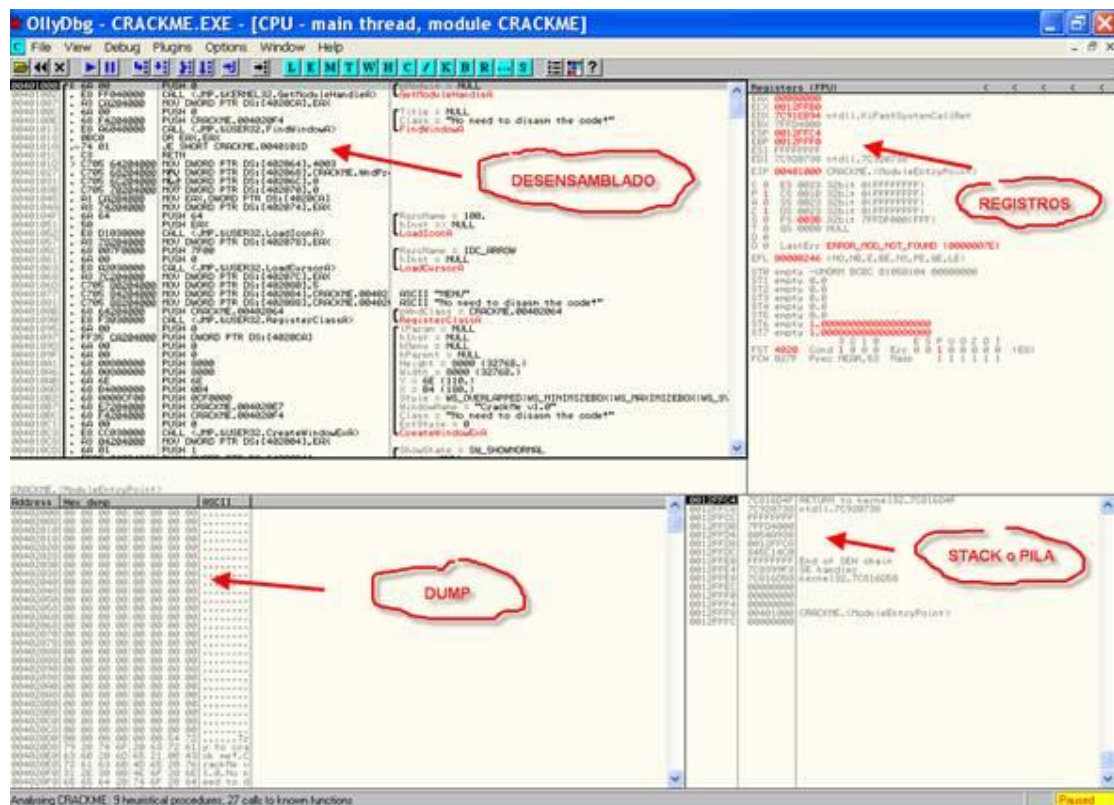


随后打开一个窗口，找到 CrackMe 。





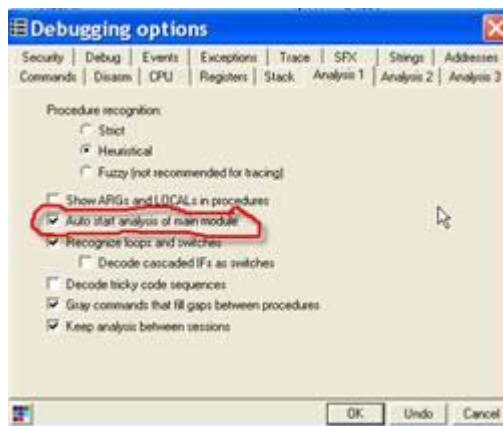
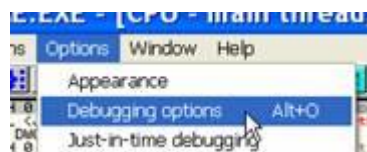
CrackMe 打开了，此时出现的内容对我们是否清楚并不重要。在对 OllyDbg 不同部分的功能和少许的设置有所了解之前，我们现在关心的只是是如何将它打开。之后，当学习下一节课程时，当说到 Dump，至少你应该知道到哪里去找这个选项。



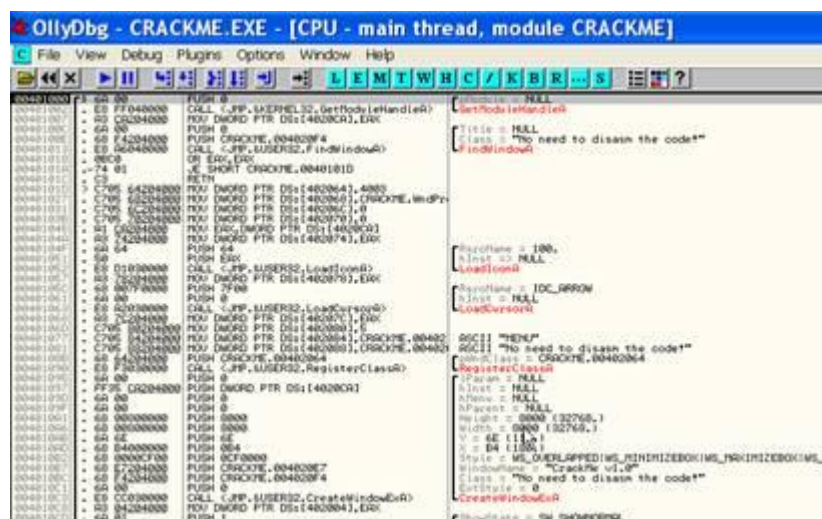
这里，我们看一下 OllyDbg 四个主要窗口：

1) 反汇编窗口

OllyDbg 在这里显示反汇编代码，我们将要以 OllyDbg 的默认配置调试分析你打开的程序。调试选项可以在 Options -> Debugging options 里更改。




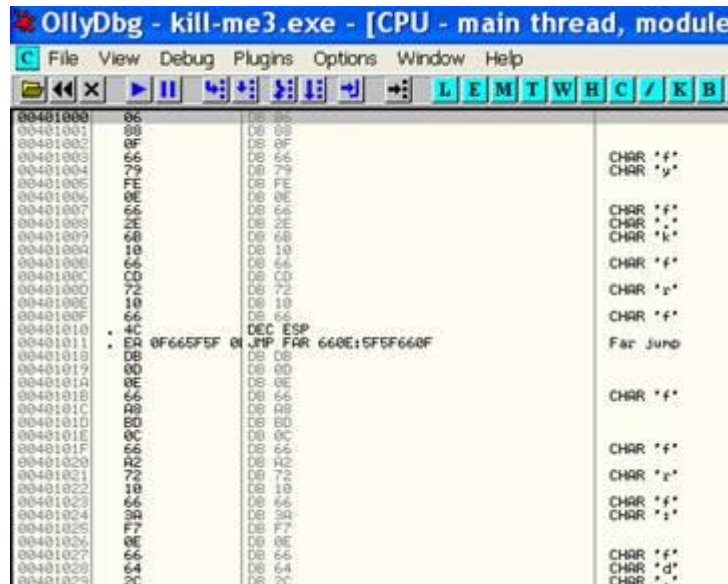
如果勾选了自动对主模块进行分析（译注 5），OllyDbg 会分析程序显示它的附加信息。



这是 CrackMe 的分析结果的开头部分，如果我们不予分析，我们看到的会有所不同。



通常 OllyDbg 显示程序的某些部分时是不正确的，错误的将可执行代码解释为数据。然后，它会显示类似于下图中的内容。



View

Copy to executable

Analysis

Bookmarks

Appearance

Analyse code

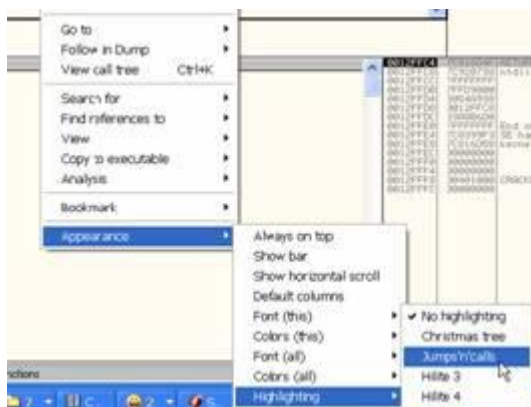
Remove analysis from module

Scan object files

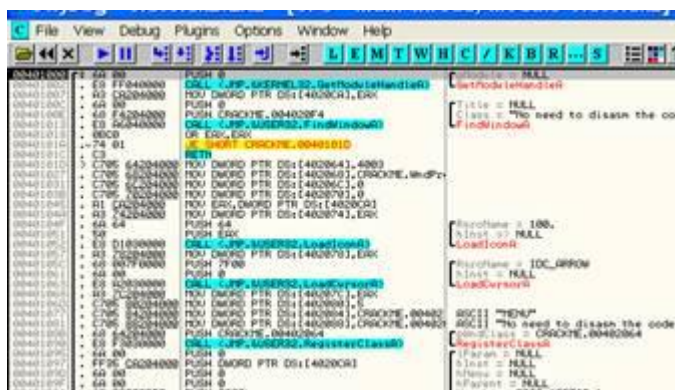
Remove object scan from module

Address	Disassembly	Comment
04010007	661E16810 66	INVL DX, WORD PTR CS:[EAX], 66
0401000C	CD 72	INT 72
04010010	4C 1064	ADD BYTE PTR DS:[ESI+4C], AH
04010011	EA 0F665F5F 0E64	JMP FAR 660E5F5F660F
04010018	DB	???
04010019	0D 0E64A8BD	OR EAX, EDI&A8BD0E
0401001A	BC 64	OR AL, 64
04010020	72128663A	MOV BYTE PTR DS:[3A661072], AL
04010025	F7	???
04010026	BE	PUSH CS
04010027	661642C 8D	SUB AL, 8D
04010028	6413B9A 0D66C1F1	CMPL BX, WORD PTR DS:[EDX+FC01660D]
04010032	BE	PUSH CS
04010033	661A6	CMPS BYTE PTR DS:[ESI], BYTE PTR ES:[EDI]
04010034	3E10E	PUSH CS
04010037	66105 E3	AND EB3
0401003D	BC 66	OR AL, 66
0401003F	6A	POP EDX
04010040	C2 0C66	RETN 660C
04010040		IN AL, DX
04010041	9C	PUSHFD
04010042	0D 66EEF6E	OR EDI, 66EF6EE6
04010047	0D 3FDE	OR EDI, 3FDE3F
04010048	661699A 0D66F3C1	INVL BX, WORD PTR DS:[EDX+CF3668D], 660D

额外的选项，高亮显示 **jumps** 和 **calls**，你使用它可能会更为方便但我个人不喜欢（这会有不同的体验），右键点击，选择 **Appearance -> Highlighting --> Jumps'n'Calls**。



会看到如下图所示。

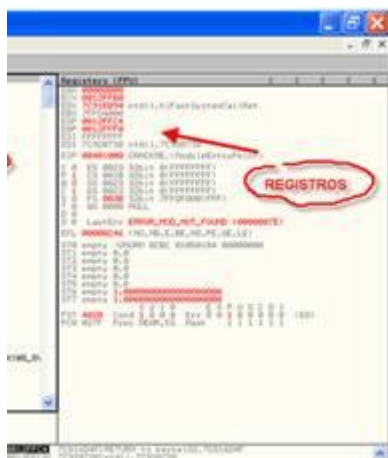


这里，我们看到 Call 句变为绿色，跳转变为黄色。

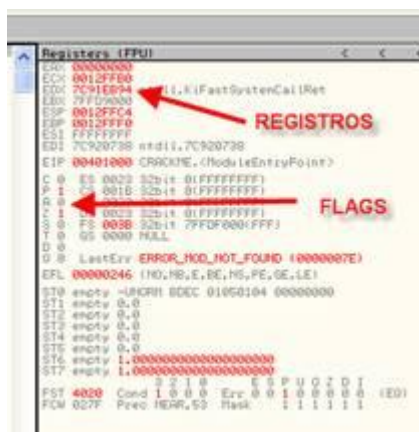
现在，反汇编窗口更加容易阅读。虽然到现在为止，我们头脑里对这些是什么意思还没有概念，但是这为将来的使用已经做了很好的准备。

2) 寄存器

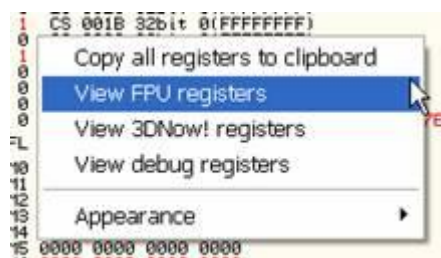
第二个重要的窗口——寄存器窗口



看一下这个在 OllyDbg 最右边的窗口，它出现了很多信息。



还有一些重要的信息我们没有看到。但是你可以设置显示模式的不同类型。右键单击，**View FPU registers** 显示 FPU 寄存器，**View 3DNow! registers** 显示 3DNow! 寄存器。**View debug registers** 显示调试寄存器。默认是第一个。

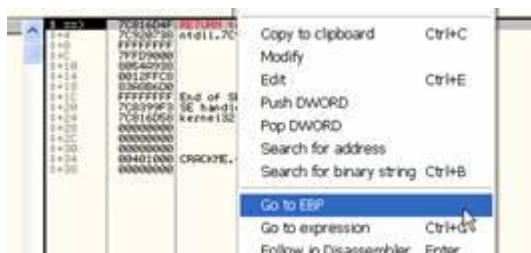


3) 堆栈窗口

现在切换到堆栈窗口。这里没有过多的配置选项，它涉及到 **ESP** 和 **EBP** 寄存器指向地址的信息显示。



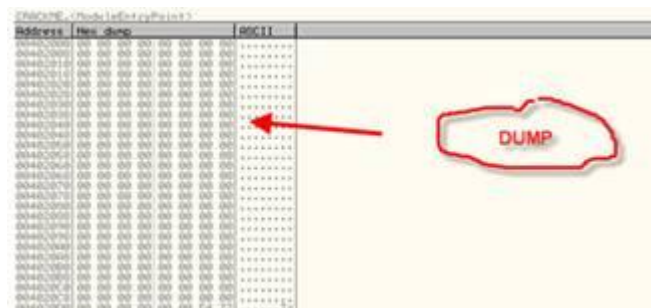
默认情况下，它显示 **ESP** 寄存器指向的信息（也是最重要的），但是你可以改变它的显示模式来显示来自涉及 **EBP** 的信息。这需要在这个窗口上点击右键，选择 **GO to EBP**。再次点击右键选择 **Go to ESP**，回到先前窗口。



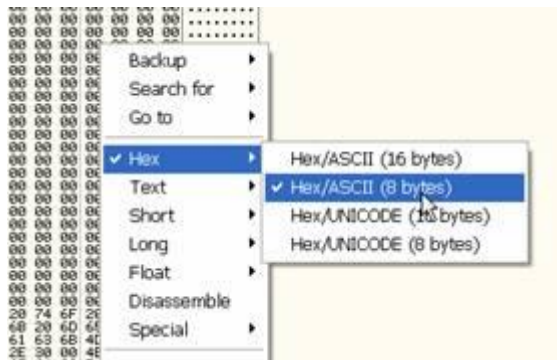
在后续课程中，我将更加详细的解释堆栈的功能。但目前，我们还只能通过修改配置选项看看发生了什么变化。

4) 数据窗口 (dump)

这个窗口有很多显示模式，可以在该窗口右击点击选择你需要的，默认模式为 8-byte Hex / ASCII。



默认的模式是最常用的，我们还可以改变它以显示反汇编代码(Disassemble)，文本 (Text) 和其它格式 (Short, Long, Float)。



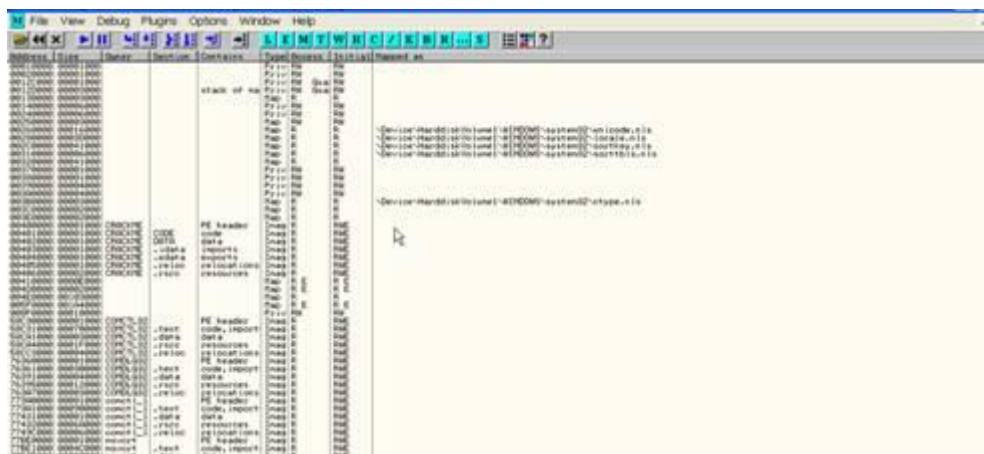
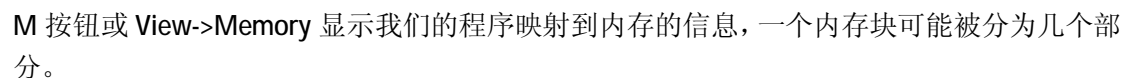
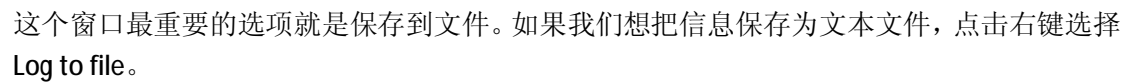
另外，选择 Special -> PE Header,我们将会在下节课中看到，非常有用。



现在我们了解了 OllyDbg 的最主要的四个窗口。还有一些窗口没有直接显示，可以通过菜单或控制面板上的图标按钮访问。

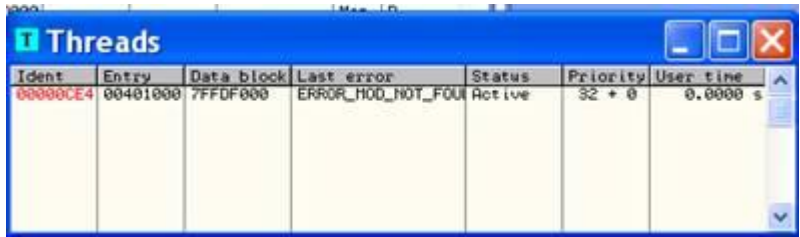


L 图标或 **View-> Log**，显示日志窗口，通过配置，可以显示 **OllyDbg** 启动时保存在日志窗口的不同类型信息，也涉及条件断点的信息。以后我们会再次用到它，现在让我们看一下当前运行进程（**CrackMe**）的信息和它加载的库文件。



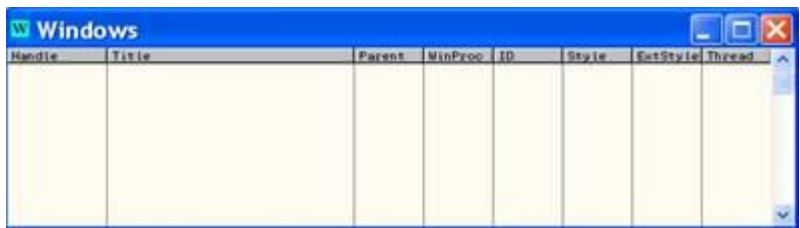
右键点击可以搜索不同种类的字符串，可以在访问上设置中断。

T 按钮或 View->Threads 显示程序的线程窗口，

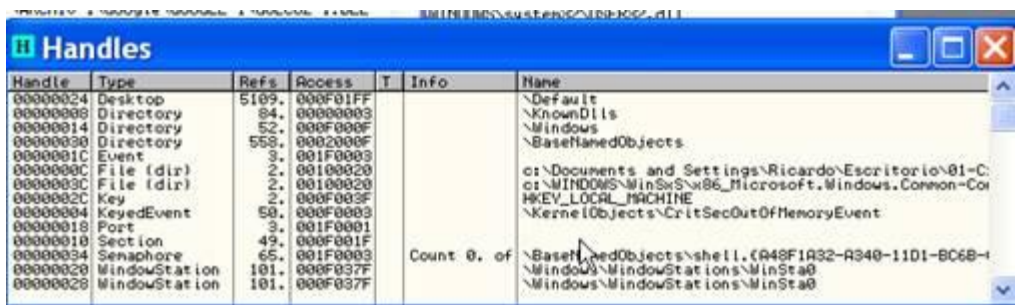


我们现在不知道它是什么，在后续课程中我们会学到和使用。

W 按钮或 View-> Windows 显示程序窗口，现在程序还未运行，窗口为空白。



H 按钮或 View-> Handles，句柄窗口，以后解释它是干什么的。

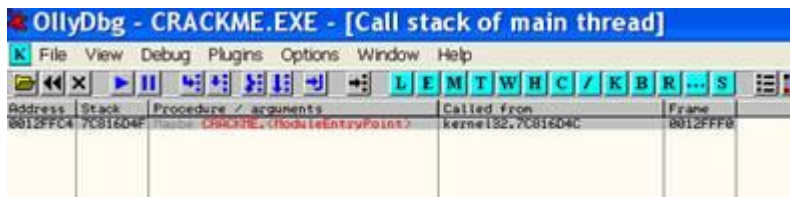


C 按钮或 View->CPU，允许我们返回到 OllyDbg 的主窗口，CPU 窗口。

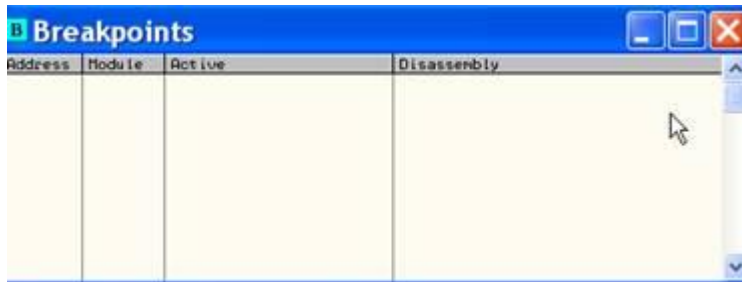
/按钮或 View->Patches，如果程序经过了修改，这里显示修改的信息，现在我们的程序还没有被修改过，所以为空白。



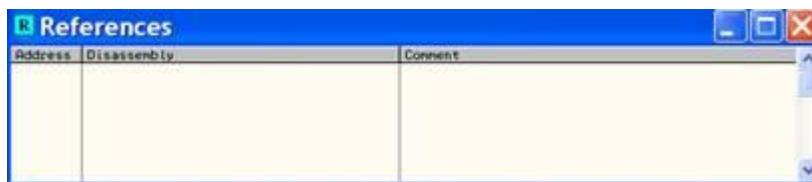
K 按钮或 View->Call stack 显示调用堆栈的窗口信息，可以尝试反向跟踪函数的调用顺序。



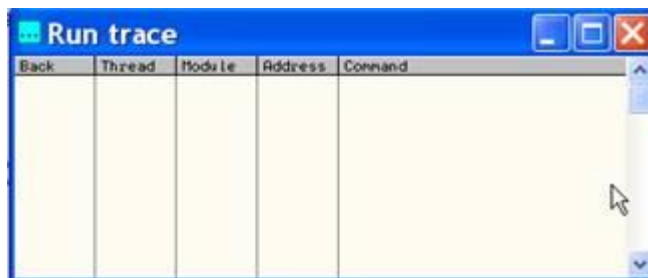
B 按钮或 View->Breakpoints 显示程序普通断点的列表窗口，这里不显示硬件断点和内存断点。



R 按钮或 View->Reference 参考窗口，显示我们在 OllyDbg 中搜索的结果。



“...” 按钮或 View->Run trace 显示 RUN TRACE (RUN 跟踪)命令的结果。这里我们也可以通过 Log to file 保存输出结果到文件。

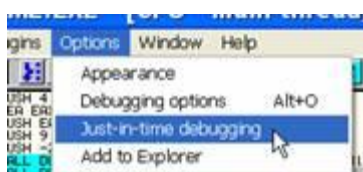


我们遍历了一下控制面板上最重要的图标，在后续课程的学习中，你会逐渐熟悉它们。

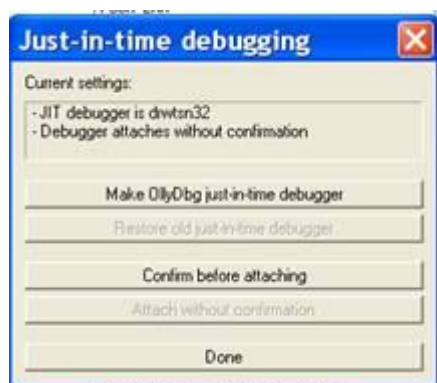
怎样设置实时调试 JIT (JUST IN TIME DEBUGGER)

当然，我们不一定会用的实时调试，除非特殊情况。因为如果我们运行的程序出现崩溃时，我们不需要使用 OllyDbg（默认为 Dr.Watson）。

让 OllyDbg 成为实时调试器，选择菜单 Options-> Just-in-time debugging。



点击 Make OllyDbg just-in-time debugger 后再点击 done。



取消这个特性，你需要在相同的位置点击 Restore old just-in-time debugger 和 Done。



在 OllyDbg 中关联插件

OllyDbg 允许你使用插件，这样会对解决问题有所帮助，通过设置 CommandBar 插件，来学习怎样设置。

点击[这里](#)（译注：随文附带）下载插件。



解压插件压缩包，查看文件夹里的内容。



首先，你需要为插件建立文件夹，我将把它建立在 C:\，名为 **PLUGINS**。

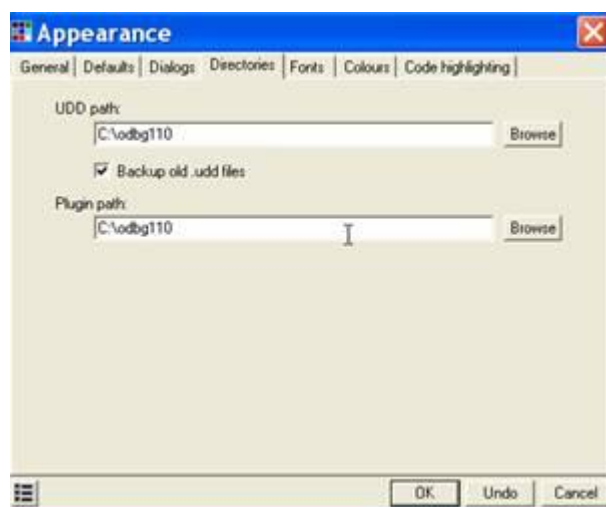


当然，此文件夹可以放在任何位置，但我喜欢将它放在 C 盘，我们现在马上配置 OlllyDbg，使 OlllyDbg 将此文件夹认可为所有插件的存放目录。

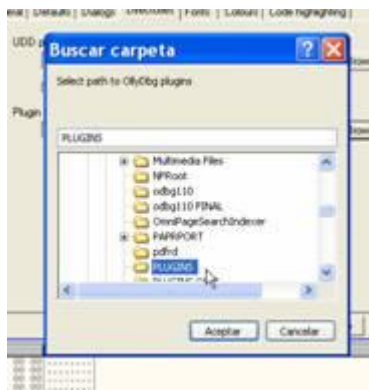
要这样做，选择 **Options->Appearance**。



然后在窗口中选择 **Directories** 标签。



我们看到作为插件路径的目录已被指定, 即 **OlllyDbg.exe** 所在的目录, 可以把插件放在那里, 但是我更喜欢将它们分开, 点击 **Pluginpath->Browse** 选择你创建的文件夹。



选择 **PLUGINS** 文件夹并保存。



出现上图, 说明你需要重启 **OlllyDbg**, 它会认可新的插件文件夹目录, 但是首先你需要拷贝刚下载的插件。

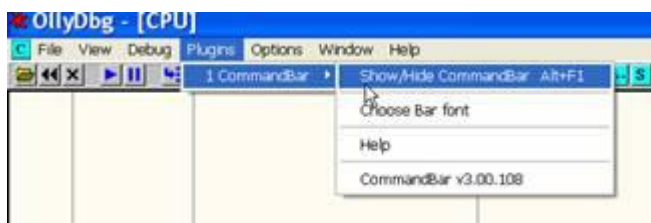


复制所有文档到 **PLUGINS** 文件夹内。

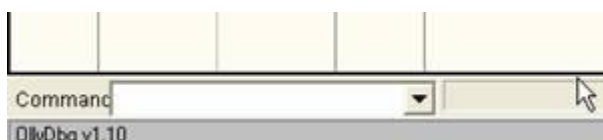


现在 **CommandBar** 插件包里的所有文件都被复制到 **PLUGINS** 文件夹内（通常不需要复制所有文件，只需复制 **DLL** 文件即可）

现在关闭 **OllyDbg**，如果已经关闭，重新打开，我们看到菜单 **Plugins** 出现 **CommandBar** 和它的选项。



CommandBar 出现在 **OllyDbg** 的底部。



这是一个供输入命令的文本框，在很多情况下对我们很有用。以后，我们会看到它的用法。学习怎样关联插件这一部很重要。

卸载插件只需要删除 **PLUGINS** 文件夹内相关的 **dll** 文件然后重启 **OllyDbg**。然而，最好总是保持 **CommandBar** 被启用。

再次从 **OllyDbg** 中打开 **CrackMe**。

OllyDbg 中最常用的快捷键：

F7 执行一行代码，遇到 **CALL** 等子程序时会进入其中，进入后首先会停留在子程序的第一条指令上。

F8 执行一行代码，遇到 CALL 等子程序不进入其代码。

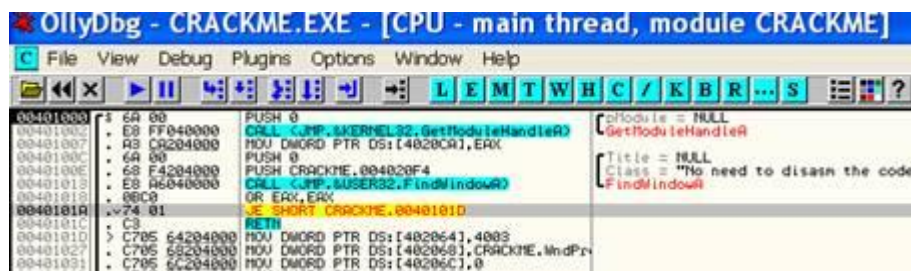
这两个键的功能有所不同，以后将会看到。

F2 在显著行设置断点，再次按 F2 删除断点。

例如：



我们想在确定位置 40101A 处设置断点。

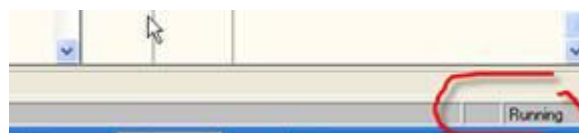


用鼠标单击此行，就会像图中所示，此行被标记为灰色，然后按 F2。

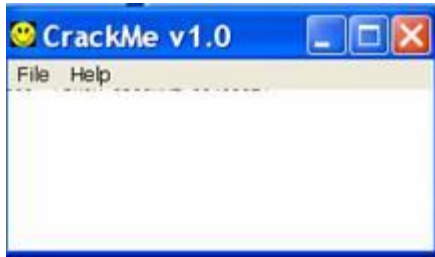


我们看到在第一列的相应位置变为红色，这说明此处已设置断点，再次按 F2 则删除断点。

F9 运行调试程序，直到遇到断点停止运行。当程序运行时，在 OllyDbg 的右下角（译注 6）会显示单词 Running。



通过运行 CrackMe，我们可以看到：



临时暂停程序，可以按 F12 或 Debug-> Pause。



我们可以看到 OllyDbg 显示单词 Paused，继续执行程序，你可以按 F9 或 Debug->Run。

选择 DEBUG-> CLOSE 就可以关闭调试程序。



以上就是 OllyDbg 总的概貌，其它深入的设置，我们将在探讨后续章节时会有所涉及。

译注 1

原文 **крэкинга**，不知何意，根据理解，用 **Cracking** 代替。

译注-2:

INTRODUCCIÓN AL CRACKING，英文译名：**INTRODUCTION TO CRACKING**。

网址：<http://ricardonarvaja.info/WEB/CURSO%20NUEVO/TEORIAS%20NUMERADAS/>
西班牙文，强大的 **Cracking** 教程，这个系列教程目前已有 1200 个以上课程内容。

译者注 3:

请从官方下载，网址：<http://www.ollydbg.de/>

译者注 4

在这里，本人建议使用系统自带的文件。

译注 5:

该选项在 Debugging options 的 Analysis1 中，Auto Start analysis of main module。

译注 6

这个位置显示当前状态，通常我们习惯将其设置在左上角，可以在 Options-> Appearance, General 标签下勾选 Show status in toolbar。

随文附件。

1. OllyDbg1.10, 本文使用原版，请从官网下载，网址: <http://www.ollydbg.de/>
2. CrackMe: ollydbg01-Crackme.zip
3. 插件: cmdbar310109c.zip

翻译说明:

该系列教程目前官方已更新到第 47 章。本文原文为俄语，译者不才，斗胆翻译，采用了能用的所有手段。虽经本人严加审校，但难免讹误。有些词句加入了译者的理解，所以可能部分内容与原文有所出入。翻译本文也是译者学习的过程，所以错误在所难免。如发现错误，敬请指正，以免误人子弟。

该系列教程链接: <http://wasm.ru/series.php?sid=17>

本文原文链接: <http://wasm.ru/article.php?article=ollydbg01>

本文原文版权: [C] Рикардо Нарваха, пер. Aquila

译文版权: BGCoder, <http://www.pediy.com/>