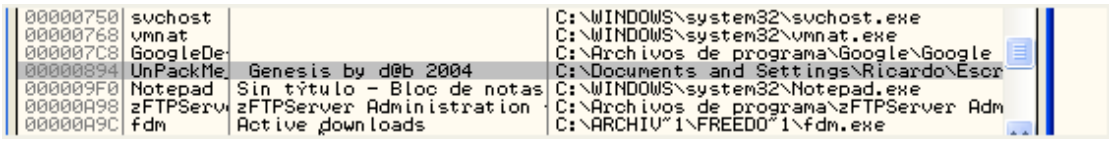
第四十五章-ReCrypt v0.80 脱壳

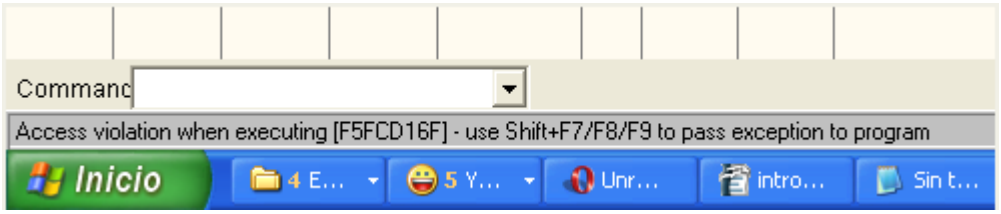
本章我们一起来分析一个用 OllyDbg 比较难脱的壳,这款壳就算我们配置好隐藏插件程序也无法正常运行,会弹出一个错误框。(PS:其实用海风月影大哥的 StrongOD 插件就可以完美解决,但是,04 年的时候 StrongOD 还没有出来,啧啧!!!)。其实用别的调试器也可以搞定,但是我们整个教程的核心是 OllyDbg,所以这里我们还是用 OllyDbg 来进行调试,嘿嘿。

我们的实验程序名称叫做 UnPackMe_ReCrypt0.80。这个程序的反调试比较强力(PS:当时算比较强力,现在不算什么了)。我们首先不加载调试器直接运行该程序。

好,程序运行起来了,但是大家细心的话会发现一个比较蛋疼的问题,这个程序的 CPU 占用率居然高达 99%。我们一起来看看它做了哪些手脚。

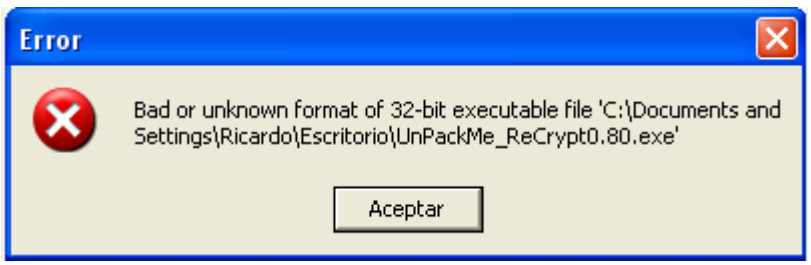
我们打开 OllyDbg,选择菜单项 File -> Attach,打开可以附加的进程列表。



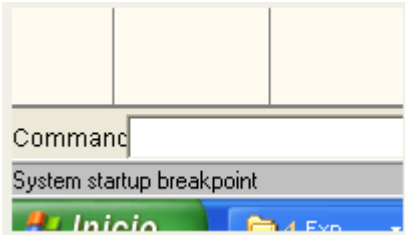


我们可以看到无法正常加载。

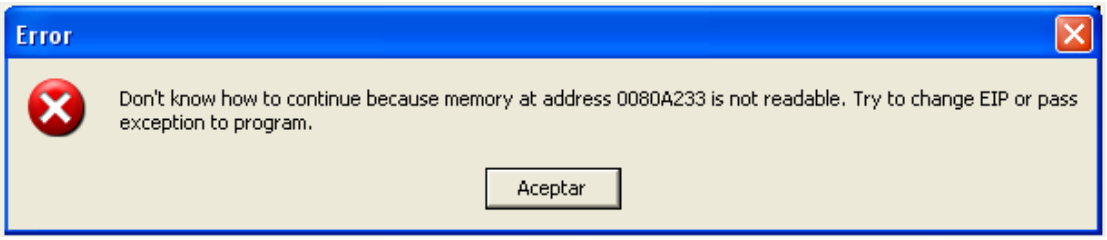
好,那么我们就用前面章节介绍过的专门用于定位 OEP 的 OllyDbg 或者 Patched 5 这个 OllyDbg 来打开它。



我们可以看到弹出了一个错误框提示无效的 PE 文件,这里我们单击 Aceptar(西班牙语译为:接受,这里译为确定)按钮。

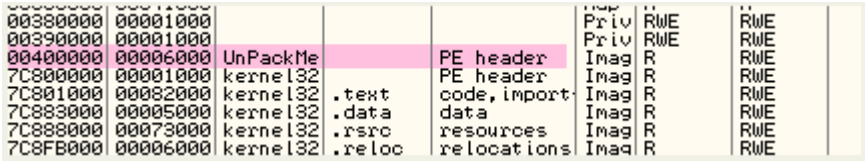


断在了系统断点处,我们直接运行起来。

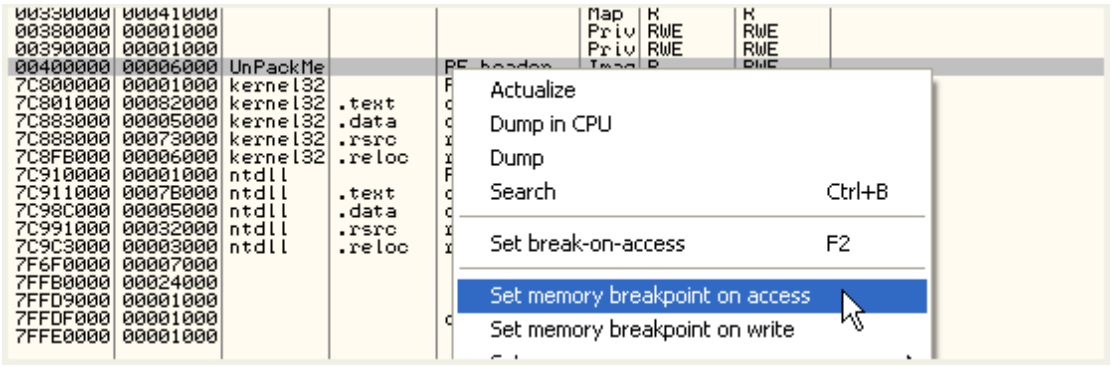


又弹出了一个错误框。

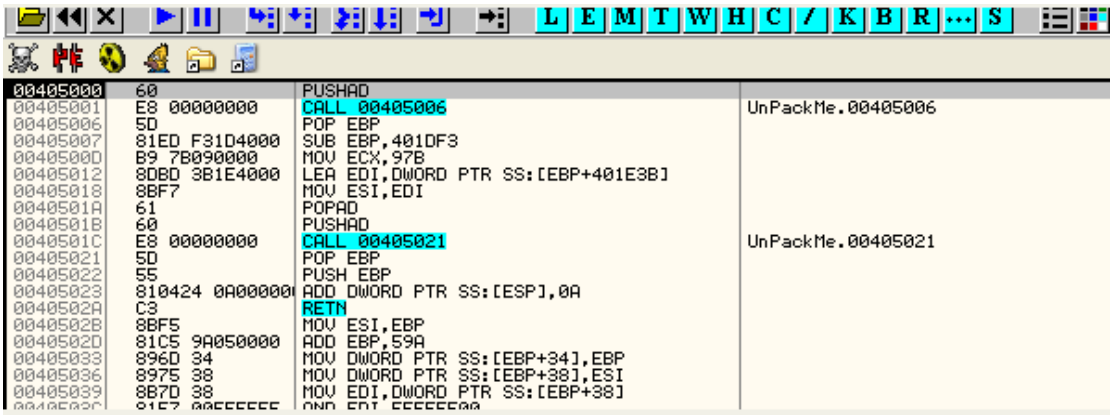
好,那么我们重启 OD,再次断在了 Sytem startup breakpoint(系统断点)处,现在我们单击工具栏上的 M 按钮打开区段列表窗口。



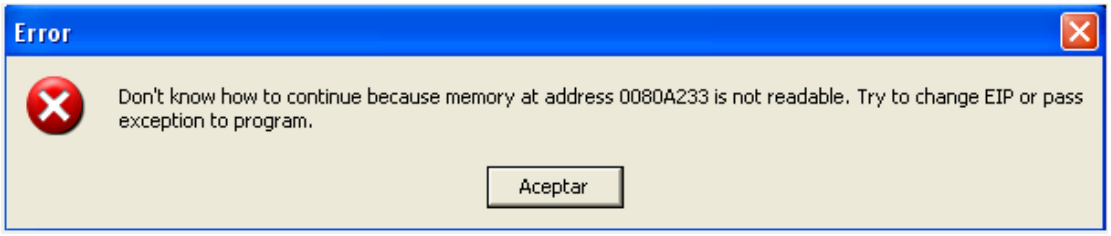
这里我们可以看到主模块仅仅只有一个区段,所以刚刚加载的时候弹出的那个无效 PE 格式的错误框是由于修改了原程序 PE 头中 NumberOfRvaAndSizes(数据目录结构数组的项数)字段导致的。接下来我们对该区段设置内存访问断点,大家应该还记得 Patched 5 这款 OD 吧,前面章节我们介绍过,这款 OD 是打过补丁的,其内存访问断点仅仅在执行代码的时候才会断下来,读取或者写入并不会断下来,这样对于我们定位 OEP 非常有帮助。



运行起来。

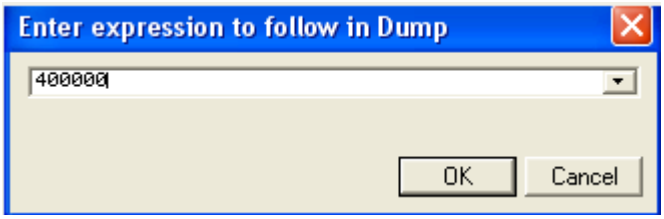


断在了 405000 地址处,好,现在我们删除内存访问断点,接着直接运行起来。

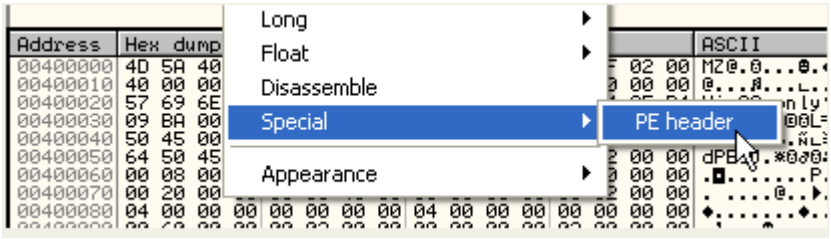


我们可以看到还是弹出了错误框。好,那么我们来看看 PE 头中 NumberOfRvaAndSizes(数据目录结构数组的项数)字段的值是多少。

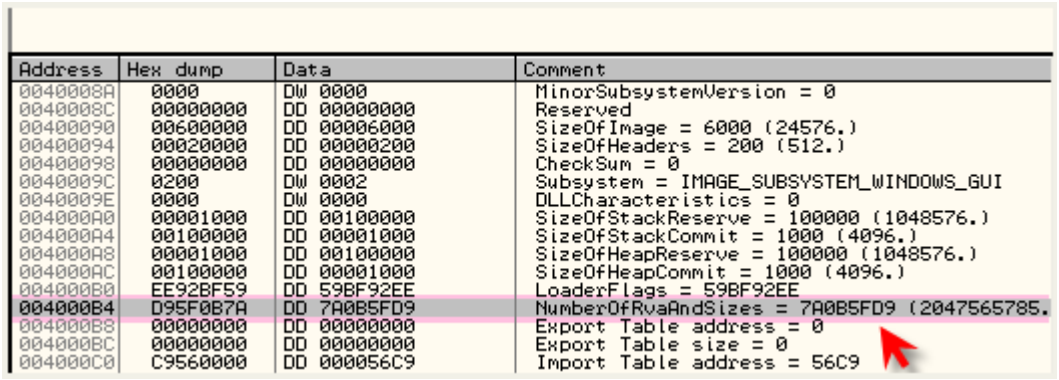
我们在数据窗口中单击鼠标右键选择 Go to-Expression,输入 400000(主模块基地址)。



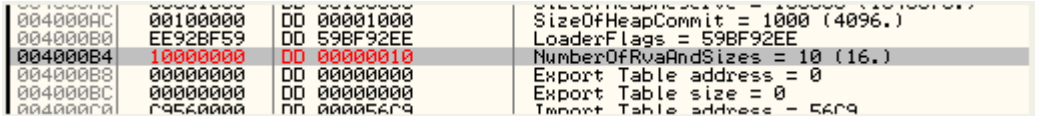
接着还是单击鼠标右键选择 Special-PE header 切换到 PE 结构解析模式。



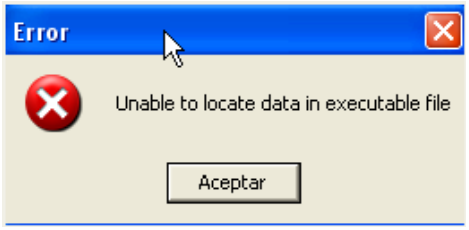
往下拉,定位到 NumberOfRvaAndSizes 字段。



我们可以看到该字段的值是 7A0B5FD9(PS:数据目录结构数组项数怎么可能会这么大,明显被做了手脚),通常情况下该字段的值为 0x10,好,这里我们把字段的值修改为 0x10,看看会发生什么。

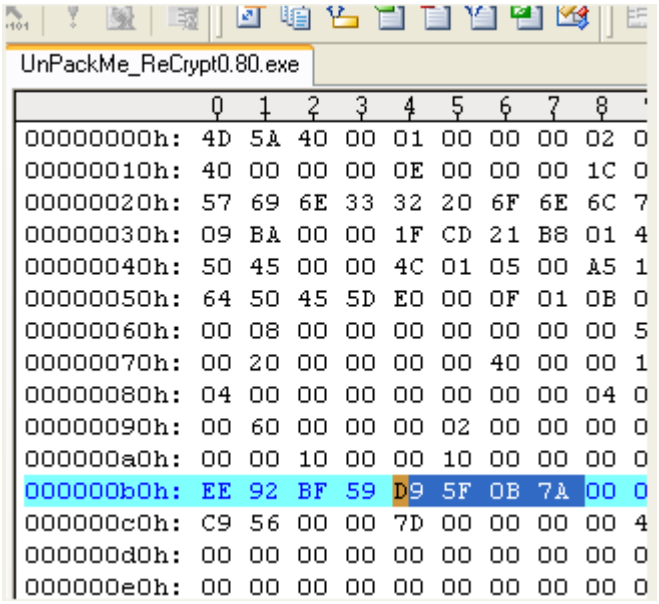


接下来我们在选中该字段,单击鼠标右键选择 Copy to executable file 尝试将所做的修改保存到文件。



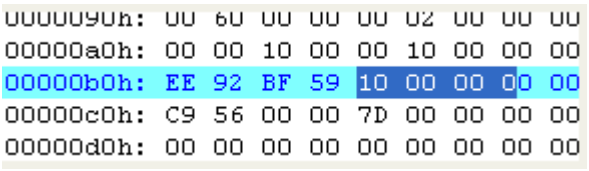
我们可以看到报错了,无法在可执行文件中定位到该数据。

好那我们尝试用十六进制编辑器来修改吧。

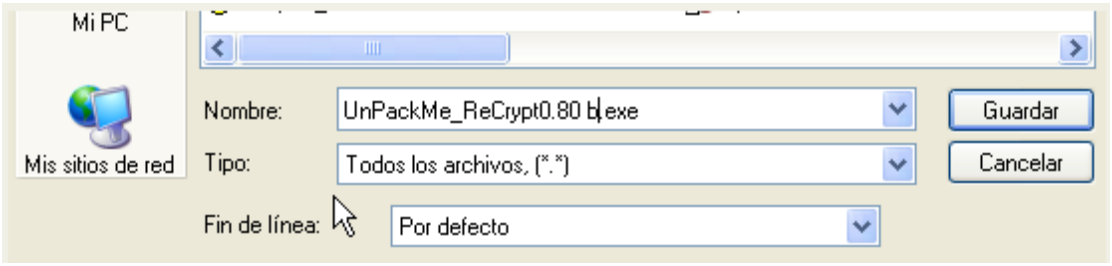


(PS:这里使用的十六进制编辑器是 UltraEdit,当然你也可以使用其他十六进制编辑器,比如 WinHex)。

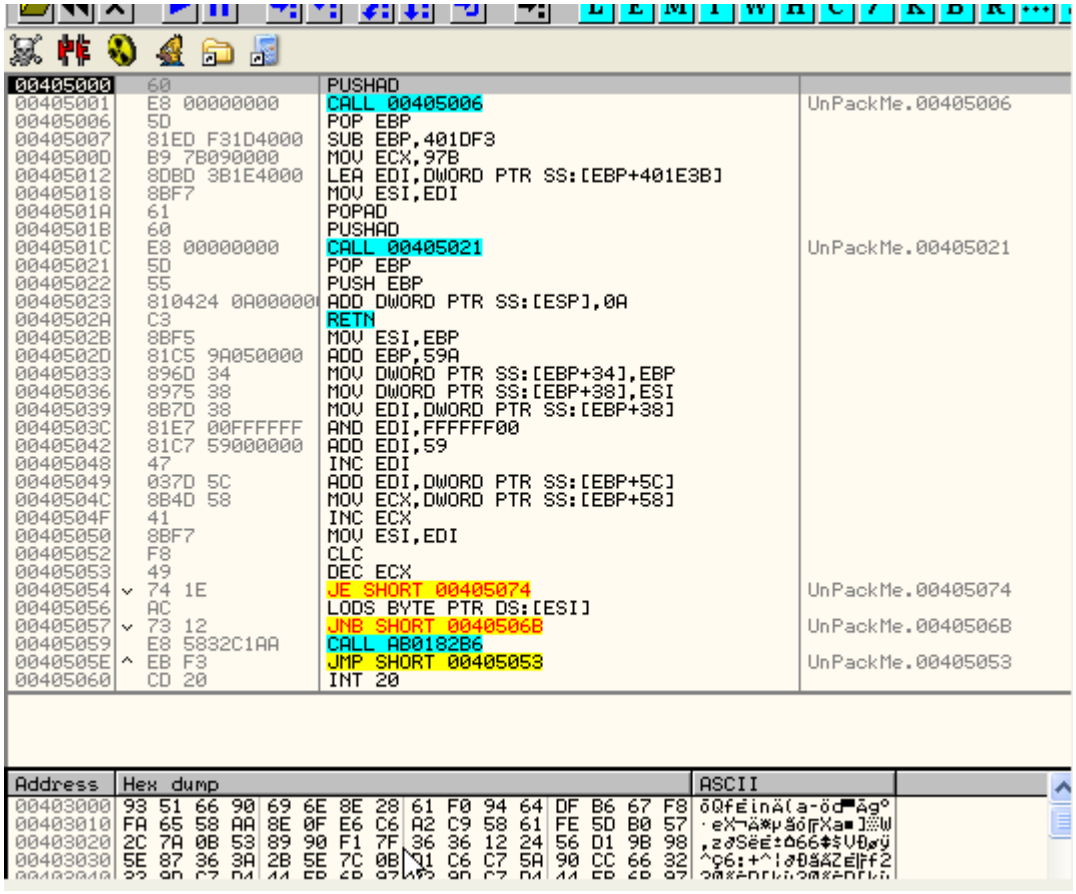
这里我们定位到 0xB4 偏移处,也就是 NumberOfRvaAndSizes 字段处,就该字段的值修改为 0x10。



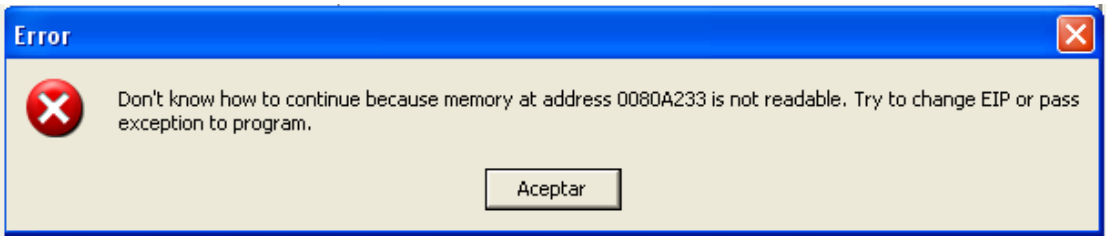
接着将所做的修改保存到文件。



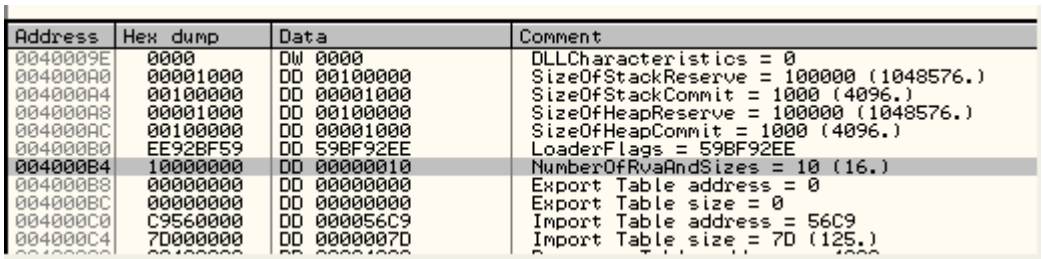
这里我将修改过的文件重命名为了 UnPackMe_ReCrypt0.80 b.exe。



我们可以看到用 OD 加载修改过的 UnPackMe_ReCrypt0.80 b.exe,正常的断在了入口点处。说明这一处反调试我们的修复是正确的,接下来我们直接运行起来。



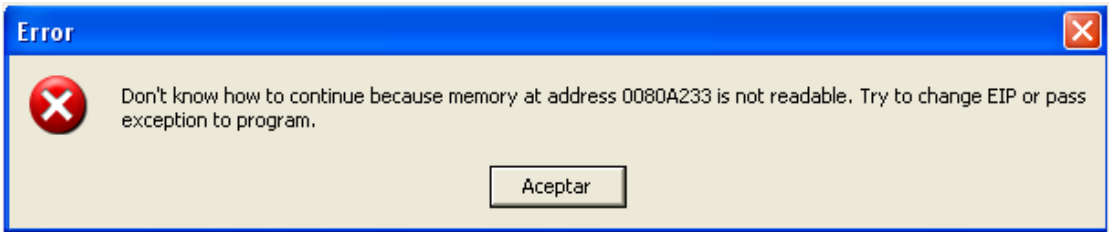
依然报错。



那么我们尝试将 NumberOfRvaAndSizes 字段的值还原看看能不能正常运行。

Address	Hex dump	Data	Comment
0040009E	0000	DW 0000	DLLCharacteristics = 0
004000A0	00001000	DD 00100000	SizeOfStackReserve = 100000 (1048576.)
004000A4	00100000	DD 00001000	SizeOfStackCommit = 1000 (4096.)
004000A8	00001000	DD 00100000	SizeOfHeapReserve = 100000 (1048576.)
004000AC	00100000	DD 00001000	SizeOfHeapCommit = 1000 (4096.)
004000B0	EE92BF59	DD 59BF92EE	LoaderFlags = 59BF92EE
004000B4	D95F0B7A	DD 7A0B5FD9	NumberOfRvaAndSizes = 7A0B5FD9 (20475657
004000B8	00000000	DD 00000000	Export Table address = 0
004000BC	00000000	DD 00000000	Export Table size = 0

好,这里 NumberOfRvaAndSizes 的值我们已经还原了,运行起来。



很明显有什么地方我们没有绕过。提示正在访问一个不存在的区段地址,OD 无法继续运行。

这里常规思路是行不通的,我们需要一点奇招。大家需要做的就是:

琢磨

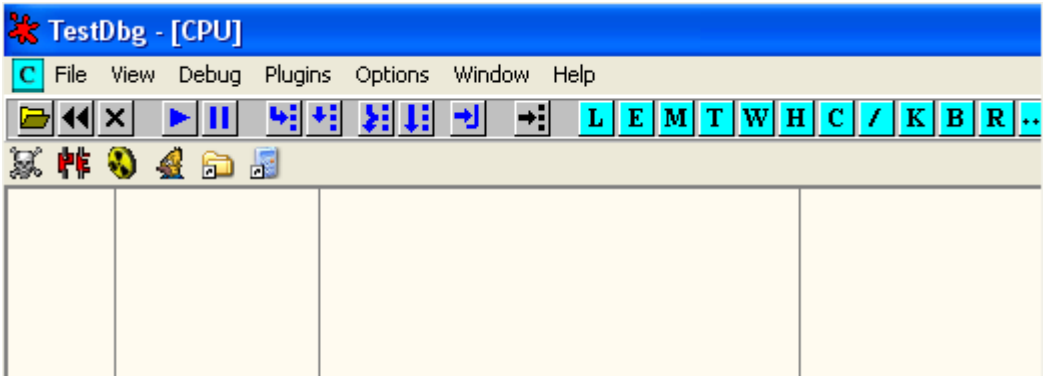
琢磨

琢磨

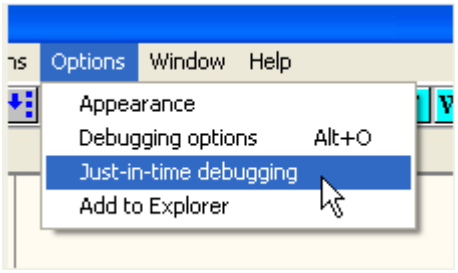
嘿嘿

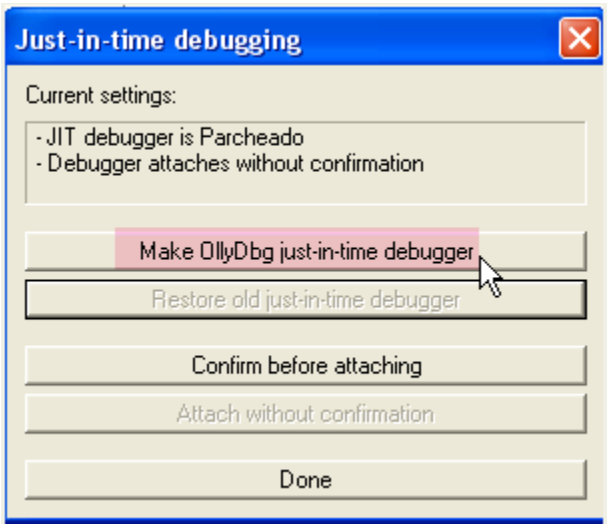
多思考思考

好了,下面我就来给大家介绍一些思路,这里并不是每一种思路都能奏效,但是你需要熟悉这些思路,因为以后经常会碰到。



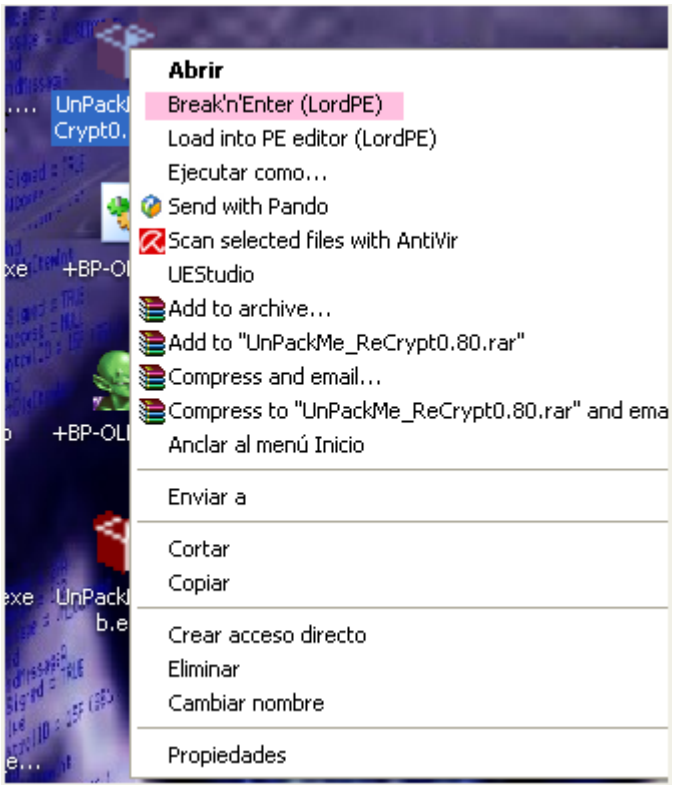
下面打开 Patched 5 这款 OD,选择主菜单项中的 Option-Just-in-time debugging,将其设置为即时调试。(Just-in-time debugging)即时调试:(PS:这个功能大家应该不会陌生吧)当有程序崩溃的时候,OD 就会自动附加上该程序。





我们单击 Make OllyDbg just-in-time debugger(将 OD 设置为即时调试器)按钮。如果以后你不想 OllyDbg 作为即时调试器的话,可以单击 Restore old just-in-time debugger(恢复原来的即时调试器)按钮。因为每当应用程序崩溃,就被 OD 附加上是很烦的。

现在我们关闭 OD,修改了 NumberOfRvaAndSizes 字段的 UnPackMe_ReCrypt0.80 b.exe 暂时不用了,因为修改 NumberOfRvaAndSizes 并不起作用,我们选中原程序 UnPackMe_ReCrypt0.80.exe,单击鼠标右键。

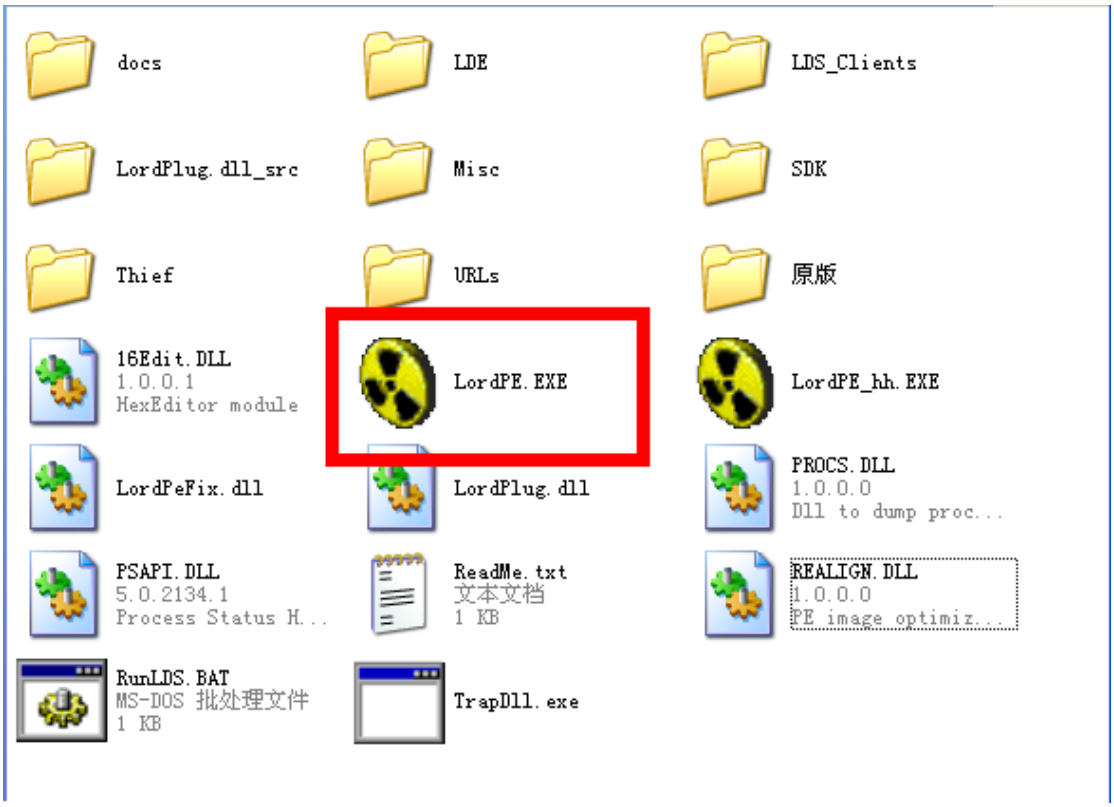


我们可以看到系统右键菜单上有个选项 Break'n'Enter(LordPE)。这个选项的作用是给指定的应用程序入口点处设置一个 INT 3 断点,直接运行该程序就会崩溃,然后即时调试器就会附加之,方便我们的调试。

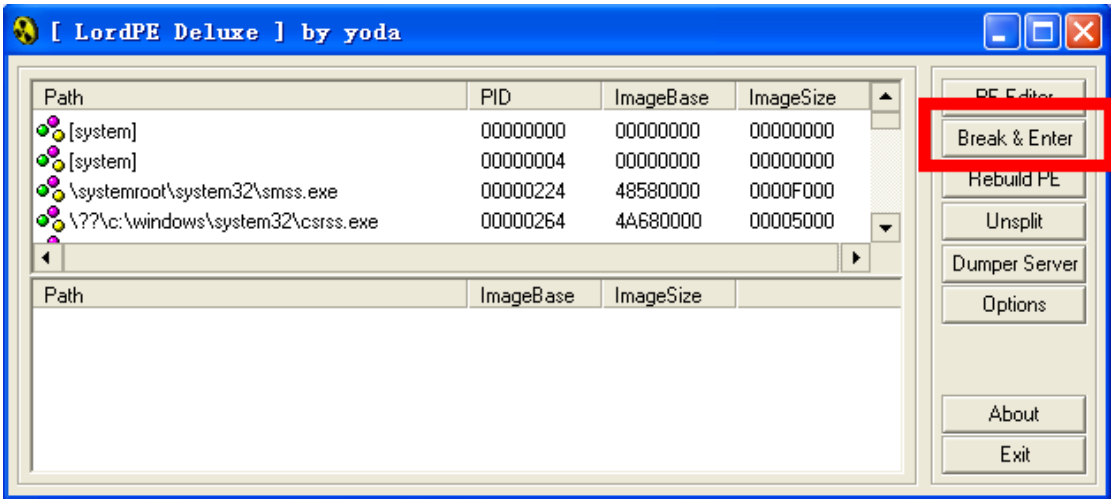
(PS:如果大家没有配置 LordPE 的话,系统菜单是不会出现 Break'n'Enter(LordPE)以及 Load into PE editor(LordPE)这两个菜单项的)。

大家做如下配置就可以出现以上两个菜单项了。

打开 LordPE 主程序。

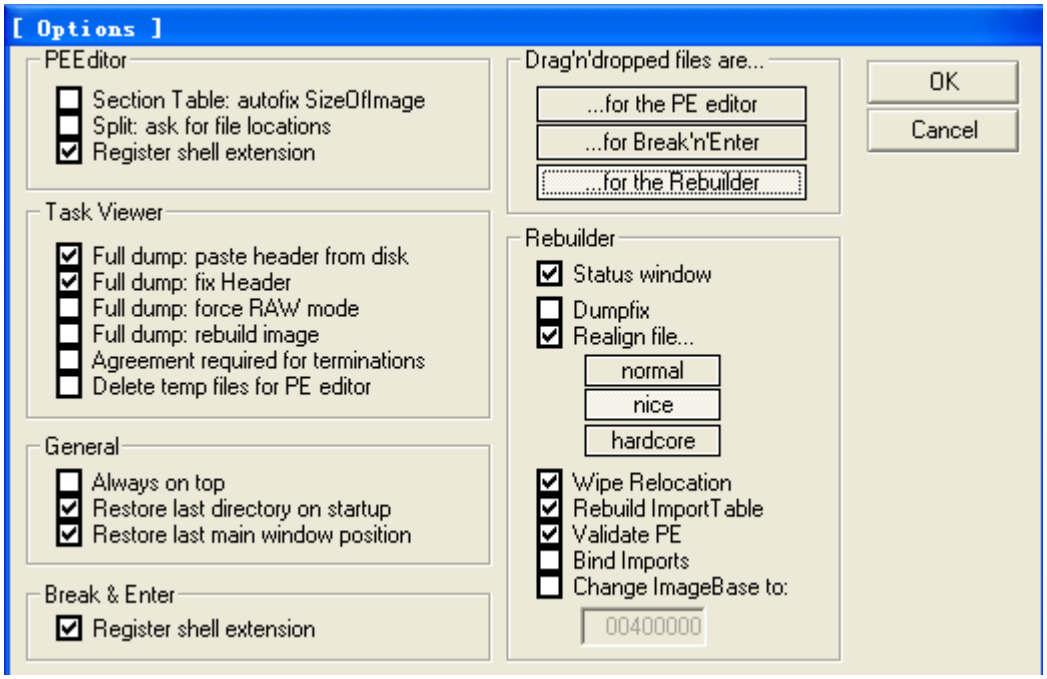


我们可以看到 LordPE 主界面中有 Break & Enter 这么一个按钮。

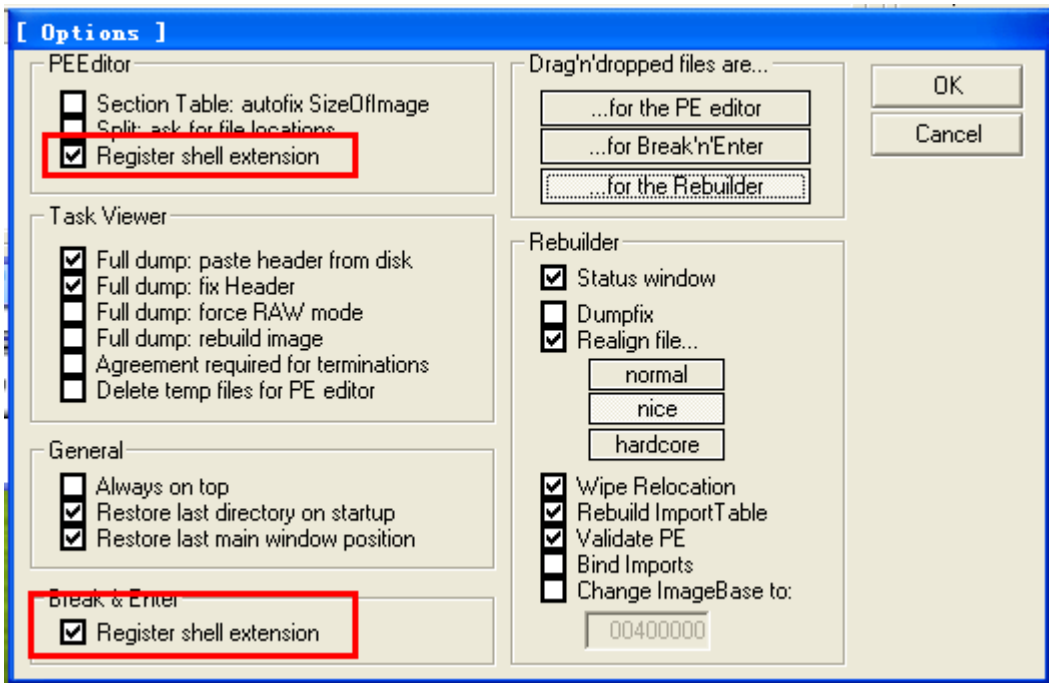


这个跟系统菜单项 Break'n'Enter(LordPE)的作用是一样的。

下面我们单击 Options 按钮对 LordPE 进行配置。

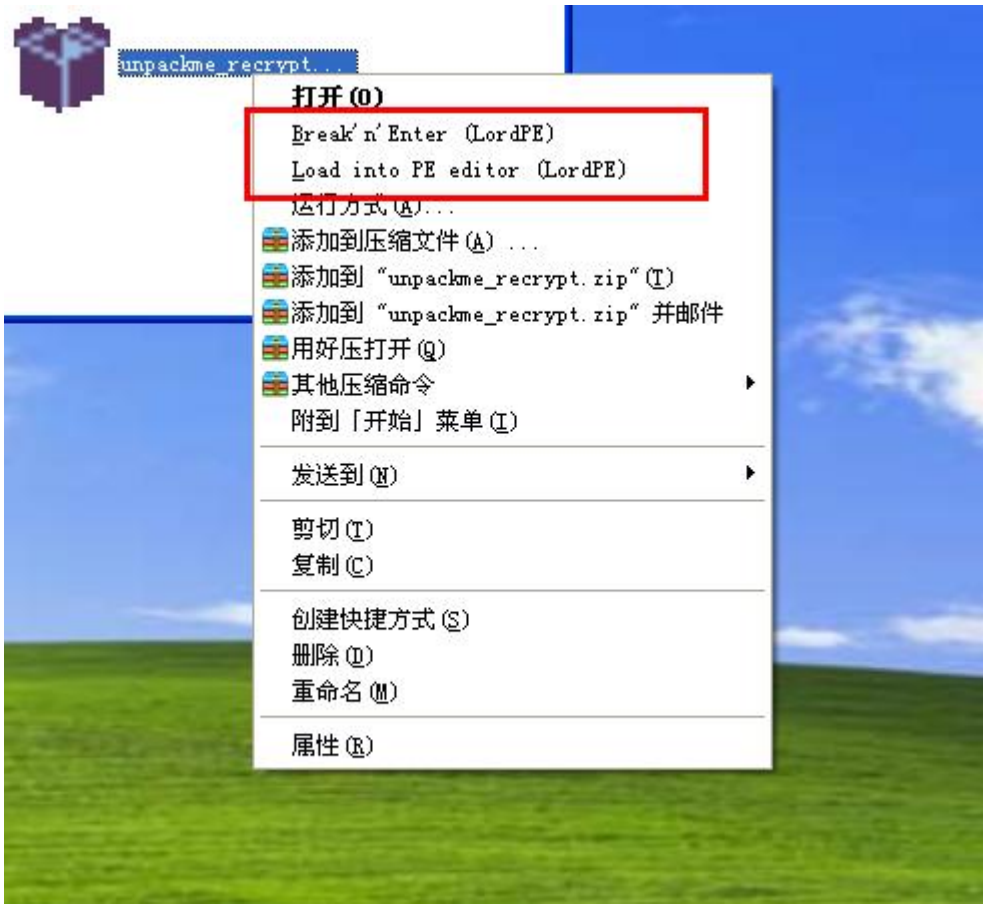


这里我们勾选上 PEEditor 以及 Break & Enter 分组中的 Register shell extension 选项就可。

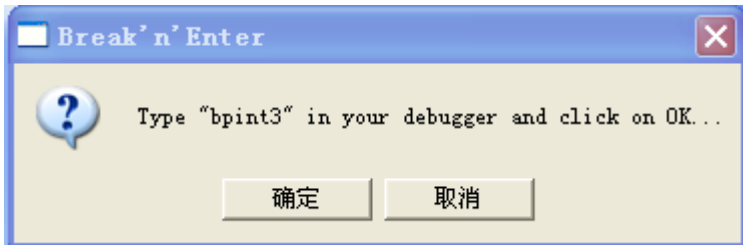


单击 OK 按钮。然后单击 LordPE 主界面中的 Exit 按钮退出。

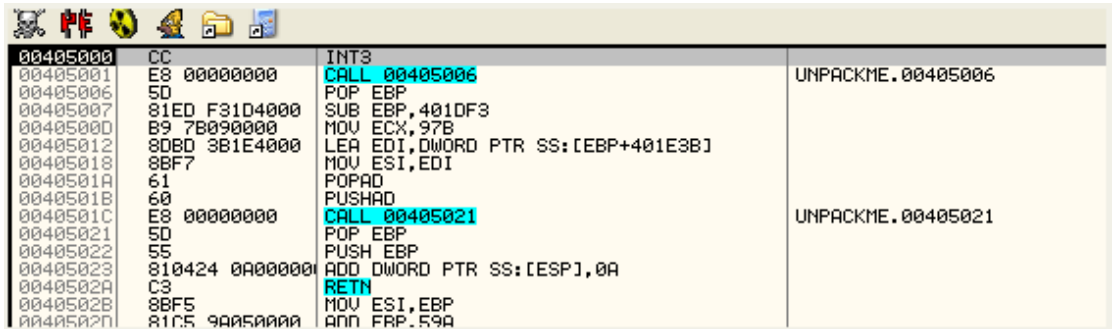
LordPE 的目录下就会生成一个名为 LordPE.iNi 的配置文件。大家在选中 UnpackMe_Recrypt 0.80.exe,单击鼠标右键,看看是不是多了两个菜单项,嘿嘿。



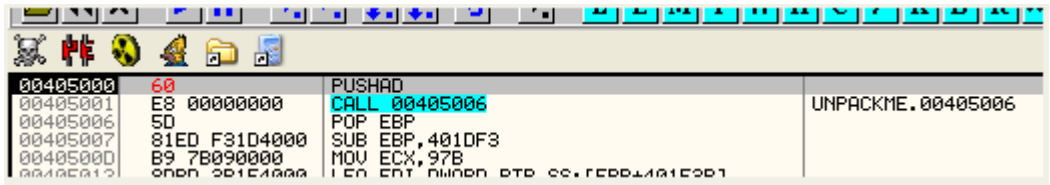
我们单击 Break'n'Enter(LordPE)菜单项。



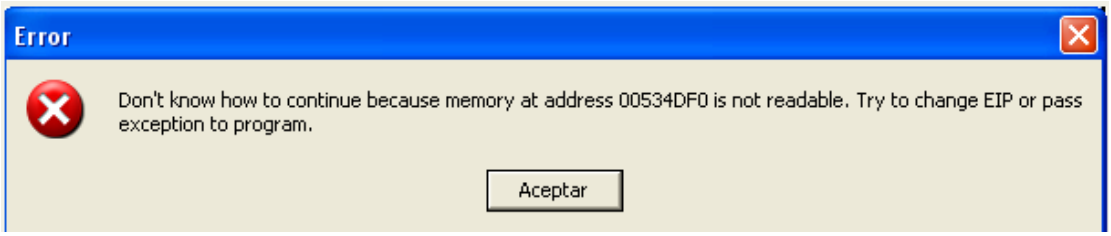
我们单击确定按钮。



OD 立马附加上了该程序,我们可以看到入口点处的第一个字节被 LordPE 修改为了 INT 3 指令。我们将 INT 3 恢复为 PUSHAD 指令。

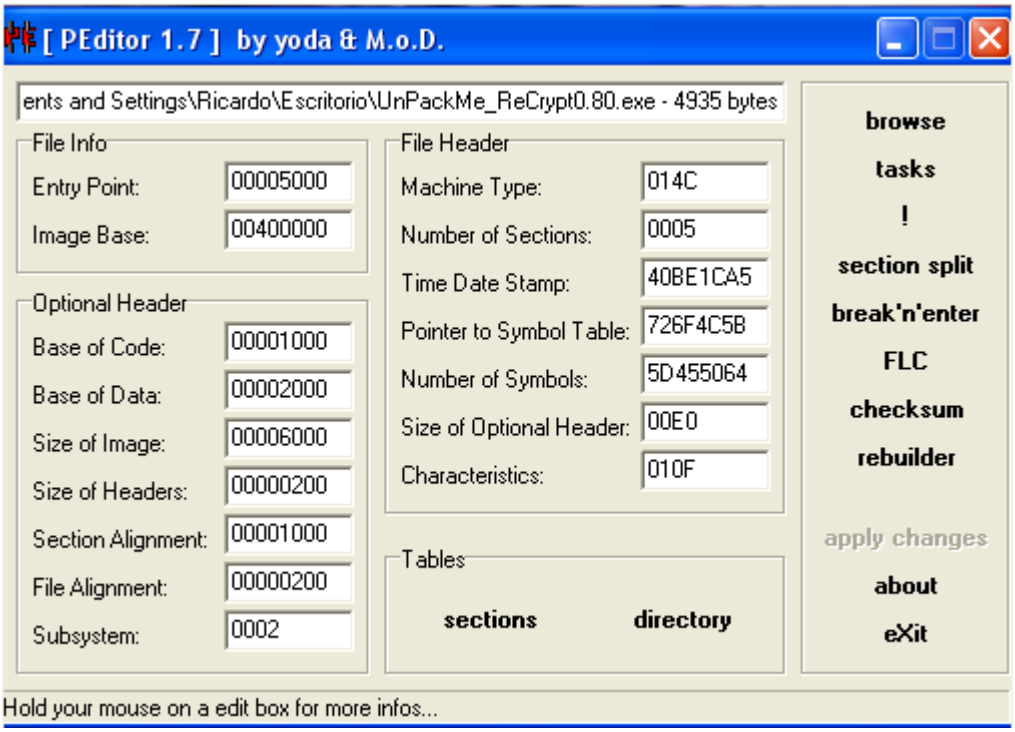


运行起来。

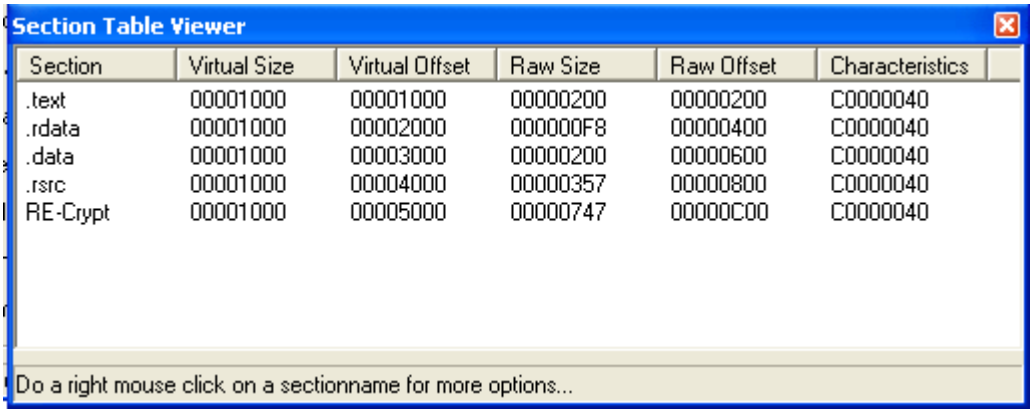


还是报这个错误。那我们只能一步步来跟踪咯,看看它都做了什么。

我们用 PEEditor 打开 UnpackMe_Recrypt 0.80.exe 文件。

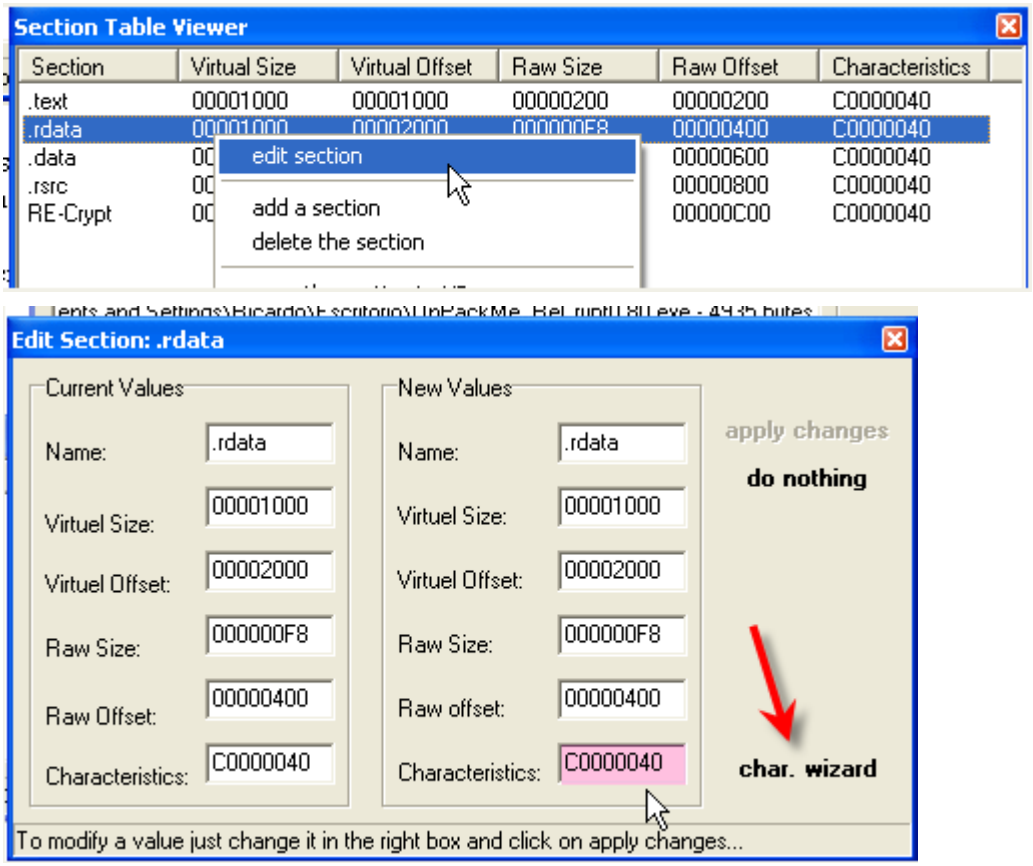


我们单击 sections 按钮查看一下区段。

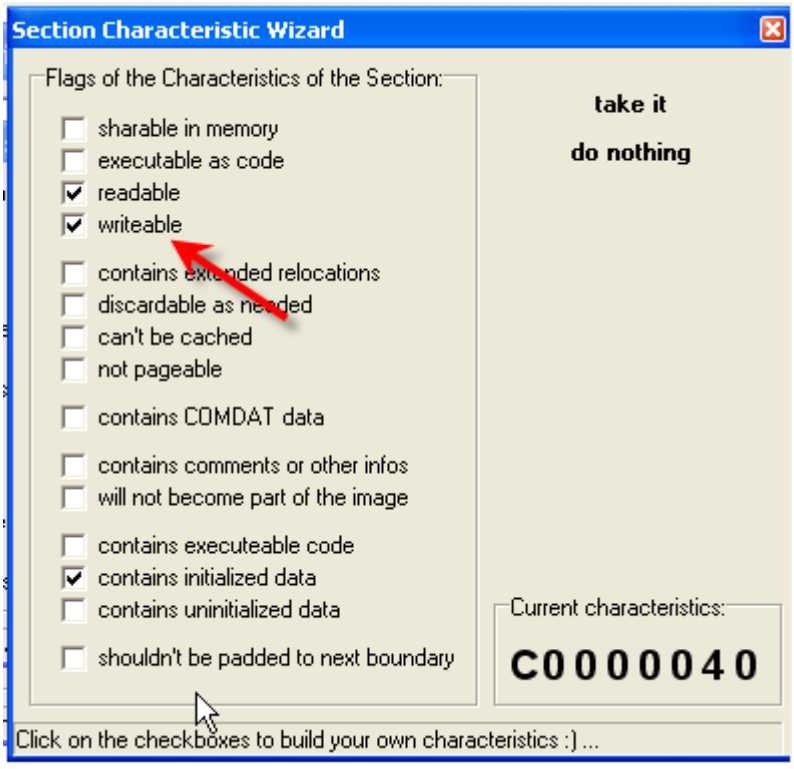


我们可以看到通过 PEEditor 可以完美的查看各个区段。现在我产生了一个想法:希望这个想法可以奏效,嘿嘿。

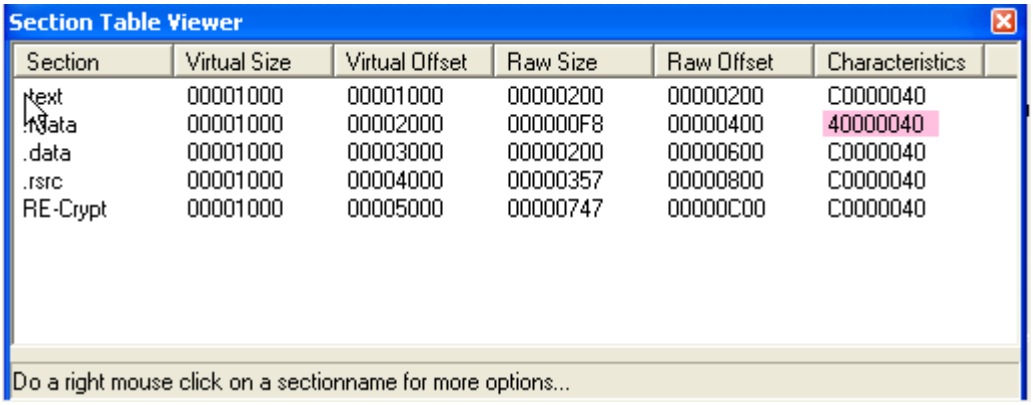
首先我们对这个程序具体情况一无所知。那我们就考虑一下常规情况的吧,一般来说,针对于该程序大部分壳会从 401000 地址处开始解密区段。我们可以尝试修改某些区段的访问权限,比如说我们去掉某个区段的可写权限。当壳尝试写入该区段的话,就会报错,此时我们之前设置 Patched 5 这个即时调试器就能派上用场了。我们一起来试一试吧。我们可以从第一个区段开始尝试,但是考虑到壳解密区段过程中会将 IAT 的中一些 API 函数地址写入到.rdata 这个区段中(也就是紧随其后的一个区段),所以这里我们首先尝试.rdata 这个区段,如果.rdata 不起作用的话,我们再来尝试第一个区段。



这里单击右下角的 char.wizard 按钮打开向导页面修改该区段的访问权限。

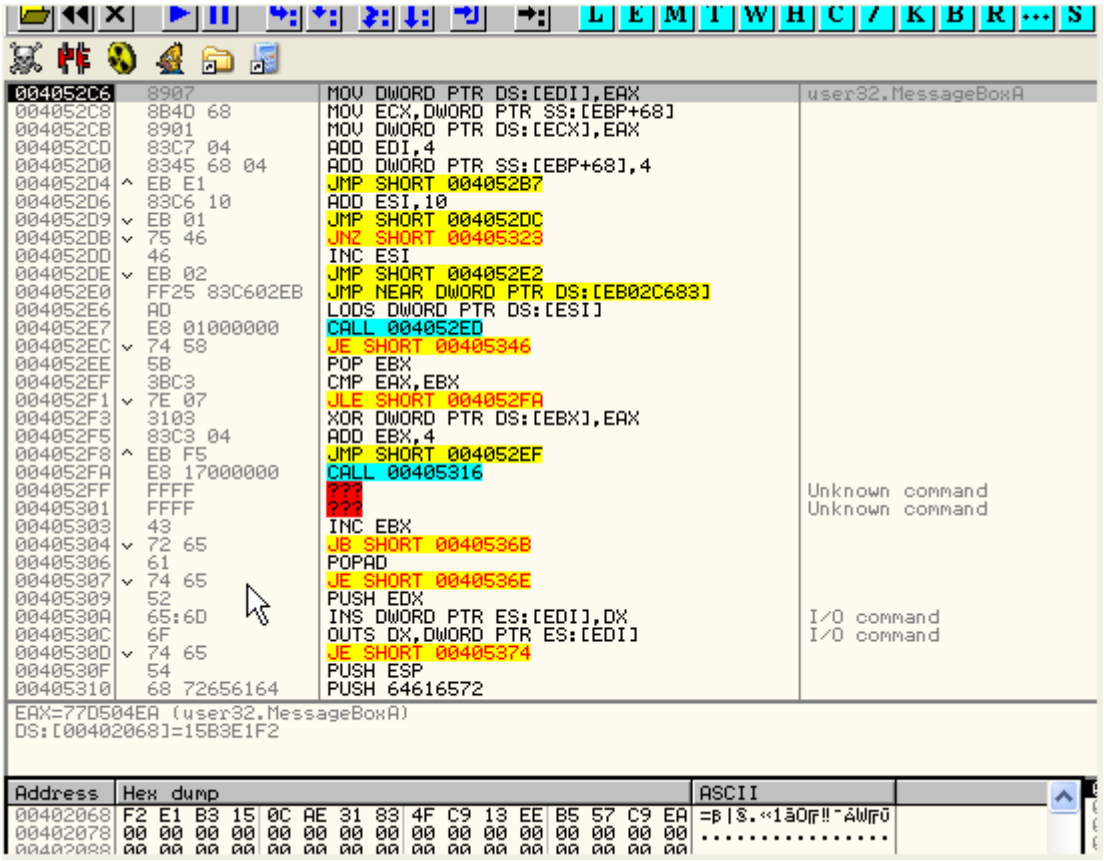


这里我们去掉 writeable(可写)复选框上的对勾。



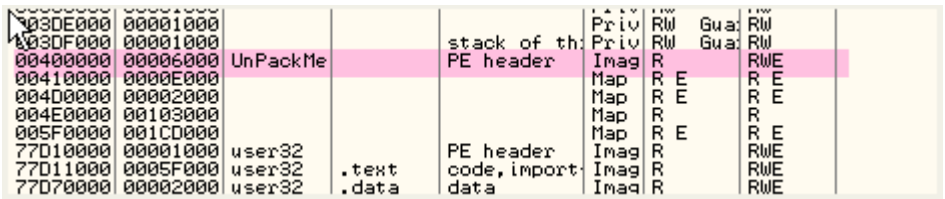
单击右上方的 take it(确定)按钮。这里我们可以看到.rdata 这个区段的访问权限已经修改成功了。有可能壳的作者会在运行时调用 VirutalProtect 之类的 API 函数将各个区段的访问权限修改回去。没有关系,我们还是来看看先。

这里我们直接双击运行已经被修改区段访问权限的 UnpackMe_Recrypt 0.80.exe。即时调试器 Patched 5 立马启动了。



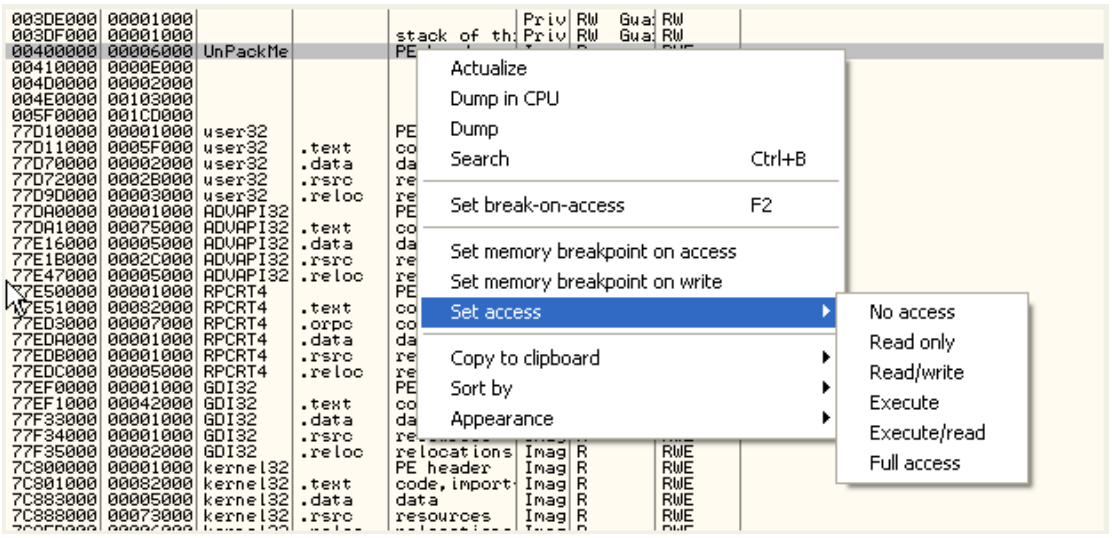
断在了这里,该指令尝试将 MessageBoxA 这个 API 函数的地址保存到.rdata 这个区段对应的地址空间中(嘿嘿,说明我的猜想是正确的)。由于我们刚刚以前去掉了.rdata 段的可写属性,说明当执行到该指令时,程序就会发生异常崩溃,即时调试器 OD 就会自动附加之。

好,我们继续。下面我们需要手工将该区段的访问属性修改回去,让其能够对该区段进行正常的写入。

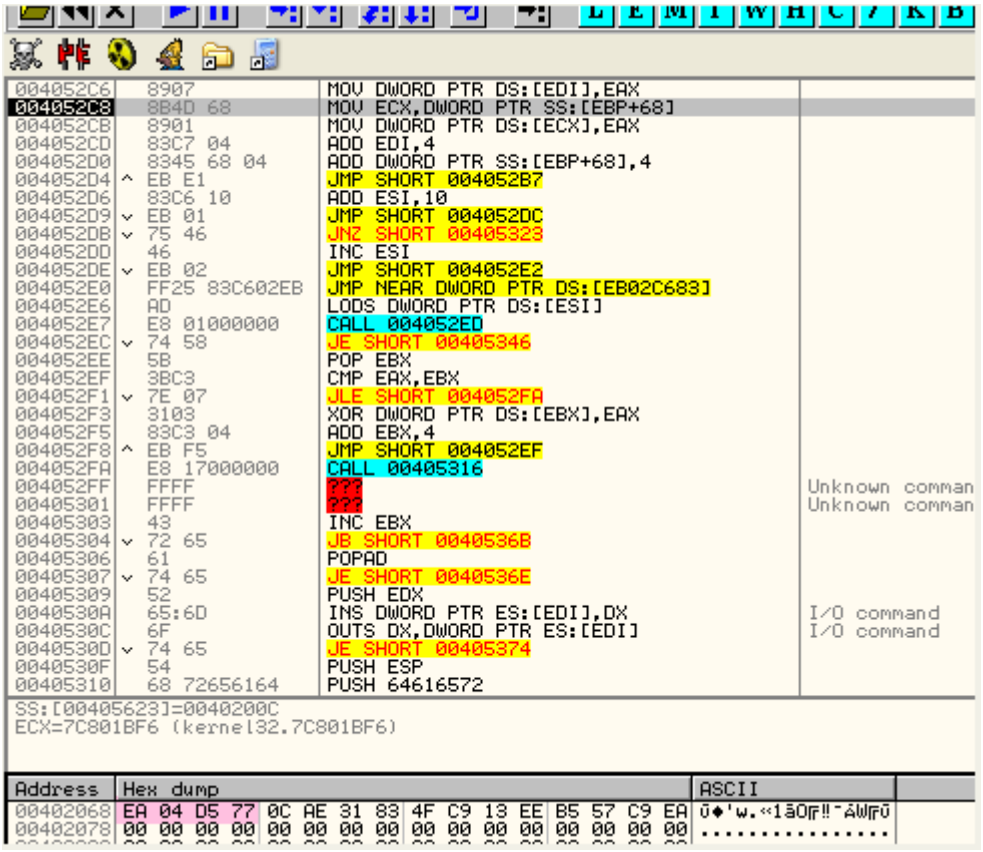


这里我们可以看到 OD 仅仅显示了一个区段,我们刚刚在 PEditor 中看到显示的是 5 个区段,没有关系。我们直接在该区段上单击

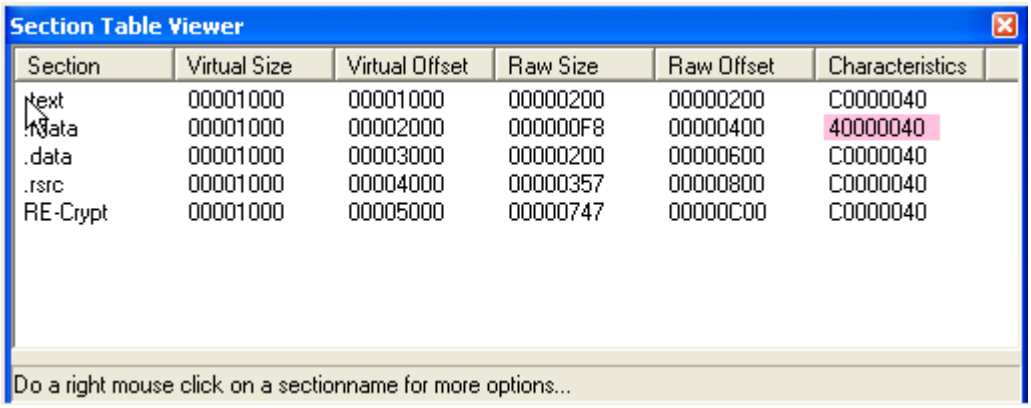
鼠标右键选择 Full access,给该区段赋予所有权限。



接下来我们单击 F7 单步试试,看看能不能执行该指令。(PS:存在两种情况,情形 1:该指令成功执行 情形 2:报错)

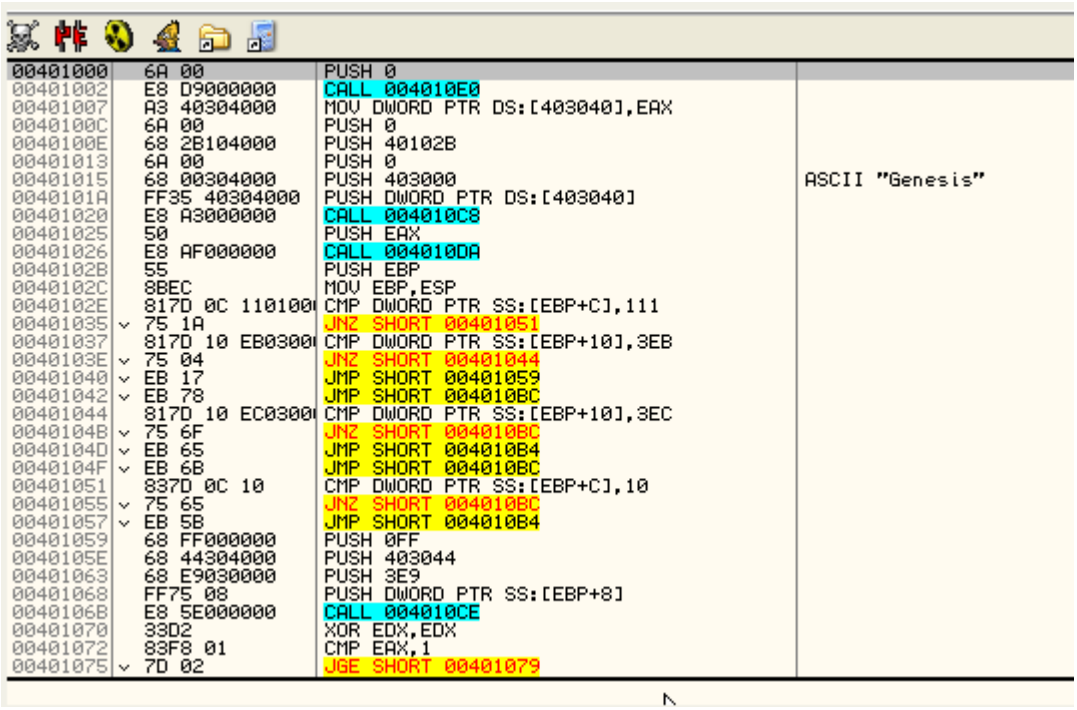


嘿嘿,我们可以看到执行功能执行了,并没有报错。由于 OD 中只显示了一个区段,看起来不是很方便,我们还是来看看 PEEditor 中的区段排列情况。

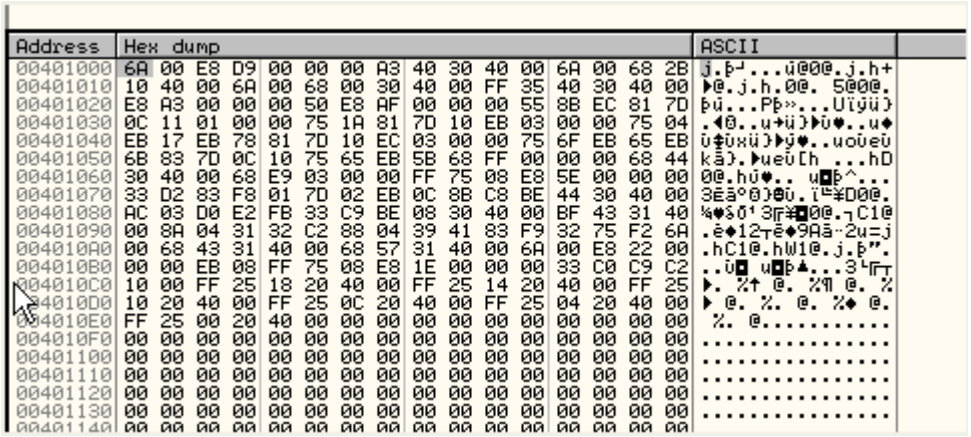


我们可以看到第一个区段(.text)相对虚拟地址是 0x1000,并且 Virutal Size 也是 0x1000,所以下面一个区段.rdata 起始地址应该是 0x402000,也就是说我们修改的是虚拟地址范围为 0x402000~0x403000 对应的区段的访问属性,嘿嘿。

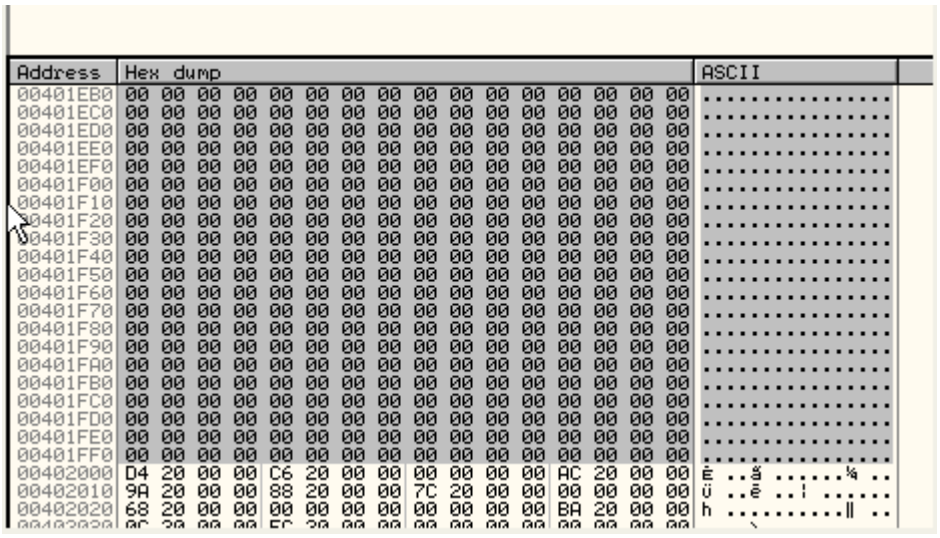
如果我们在反汇编窗口中定位到 0x401000 这个地址,可以看到区段已经解密完毕了。



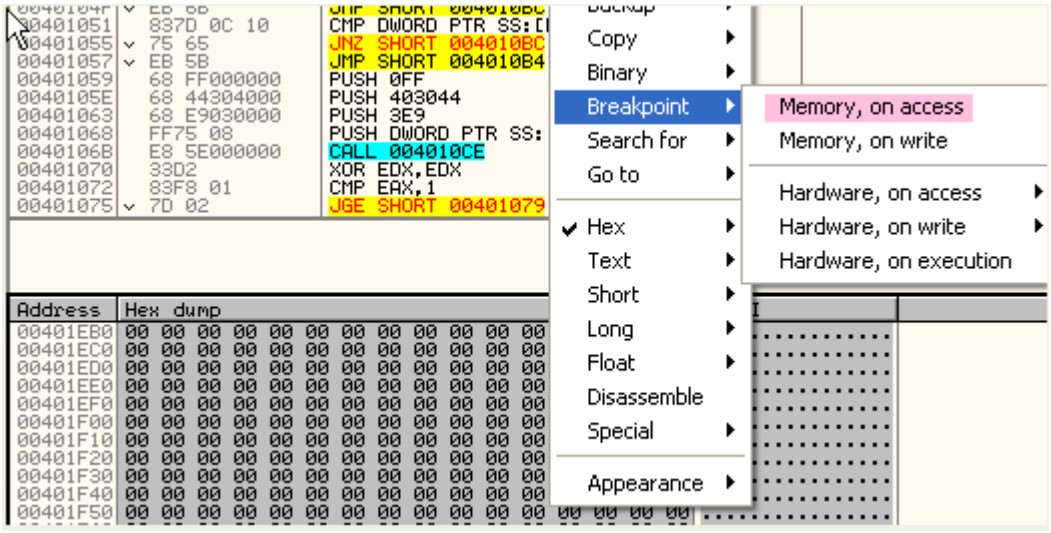
这里我们不要盲目的对 OD 中显示的唯一的那一个区段设置内存访问断点,OD 这里对区段的解析是有误的。这里我们在数据窗口中定位到 0x401000~0x402000 这一块内存地址,这块内存单元正好对应的就是.text 段。我们选中这块内存单元,给其设置内存访问断点。



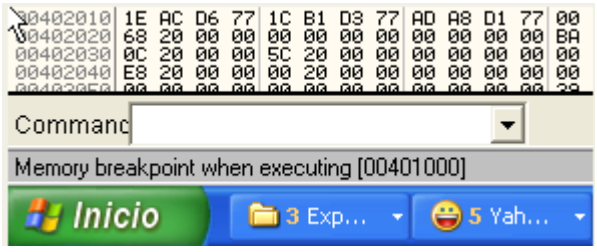
这里我们选中 0x401000 地址处的第一个字节,按住鼠标左键不放,将鼠标往下拖,直到选中 0x401000~0x402000 整个范围位置。



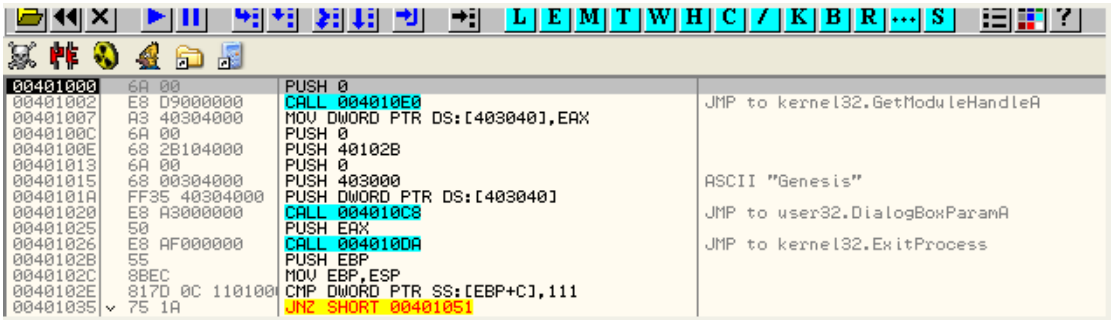
接着单击鼠标右键选择 Breakpoint-Memory,on access。



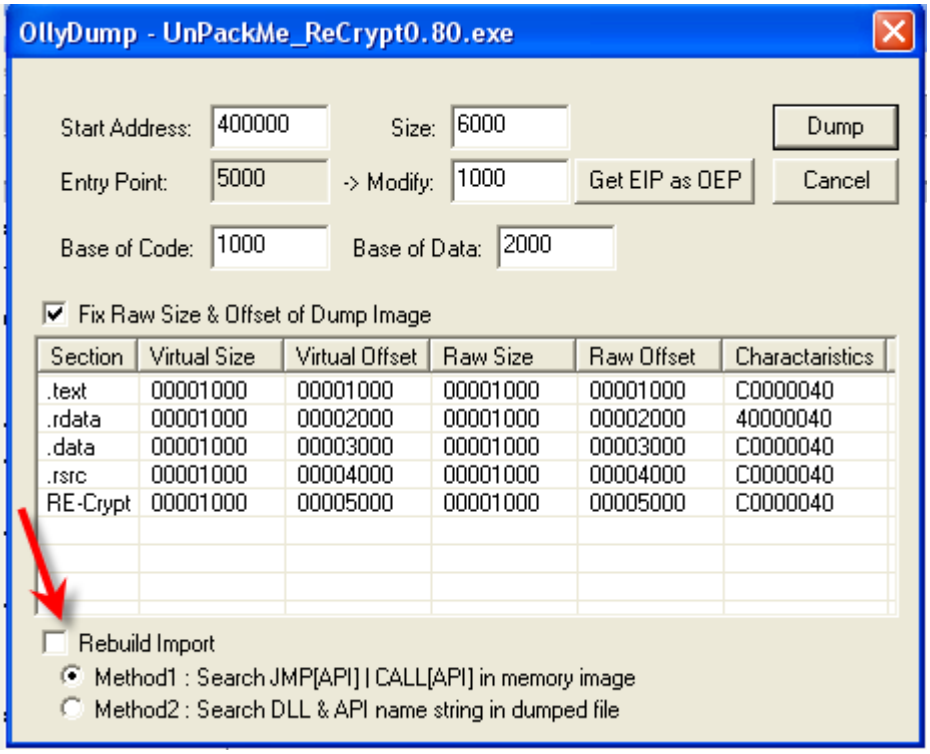
运行起来。



这里我们可以看到断在了 401000 地址处,也就是 OEP 处。



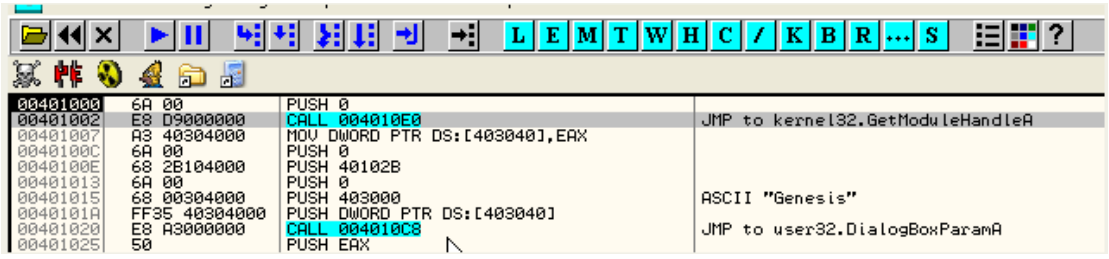
好了,最困难的一步我们已经迈过了,下面我们打开 OllyDump 对该程序进行 dump。



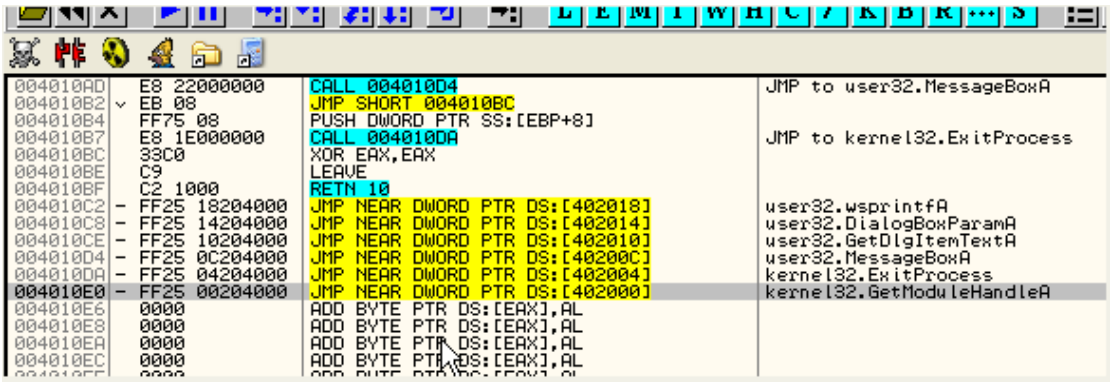
这里我们去掉 Rebuild Import 前面的对勾,不使用 OllyDump 来修复导入表。仅仅用它来 dump。

下面我们来定位 IMP REC 重建 IAT 所需要的数据。

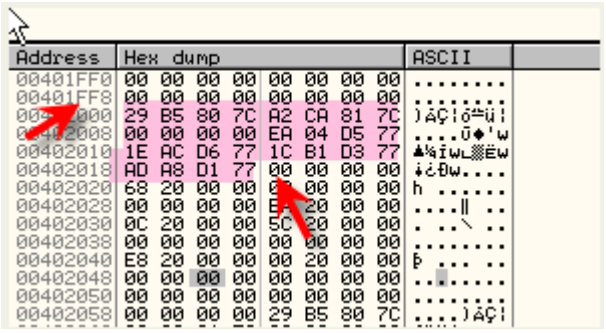
这里我们就拿 OEP 下方的 GetModuleHandleA 这个 API 函数的调用处来说吧。



这里我们在该指令上单击回车键或者单击鼠标右键选择 Follow 跟进。



可以看到我们很容易的就定位到了 IAT。很明显这几项都没有被重定位过。例如:GetModuleHandleA 这个 API 函数对应的项值是 0x402000,很显然是属于 IAT 的。

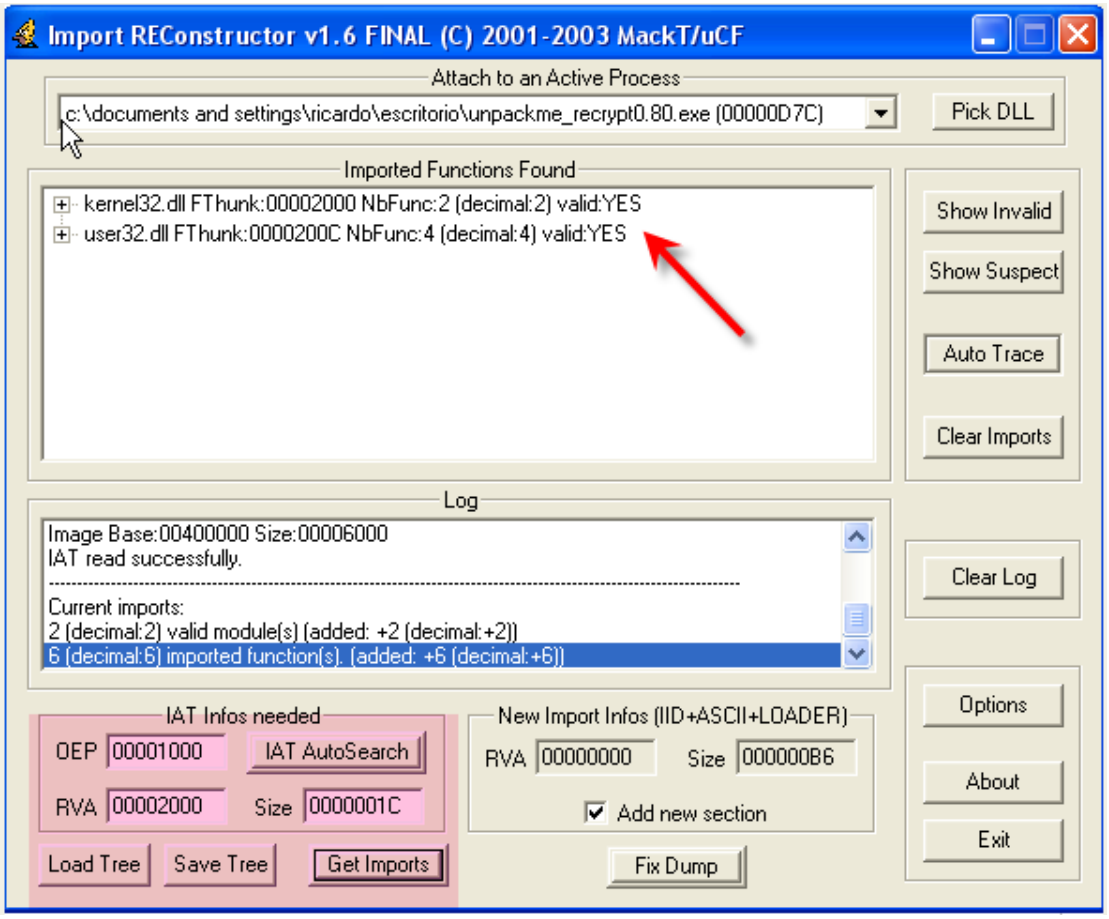


我们在数据窗口中定位到 IAT。可以看到该程序的 IAT 很小。IAT 的起始地址为 0x402000,结束地址为 0x40201C,所以 IAT 的长度的为 0x1C。下面我们打开 IMP REC 将 IAT 的信息填充进去。

IAT 起始地址(RVA):0x2000

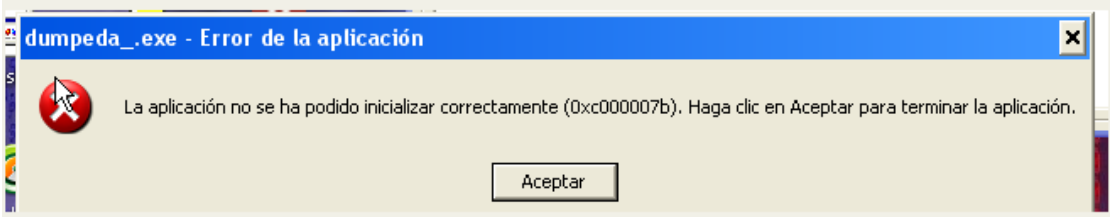
IAT 大小:0x1C

OEP(RVA):0x1000

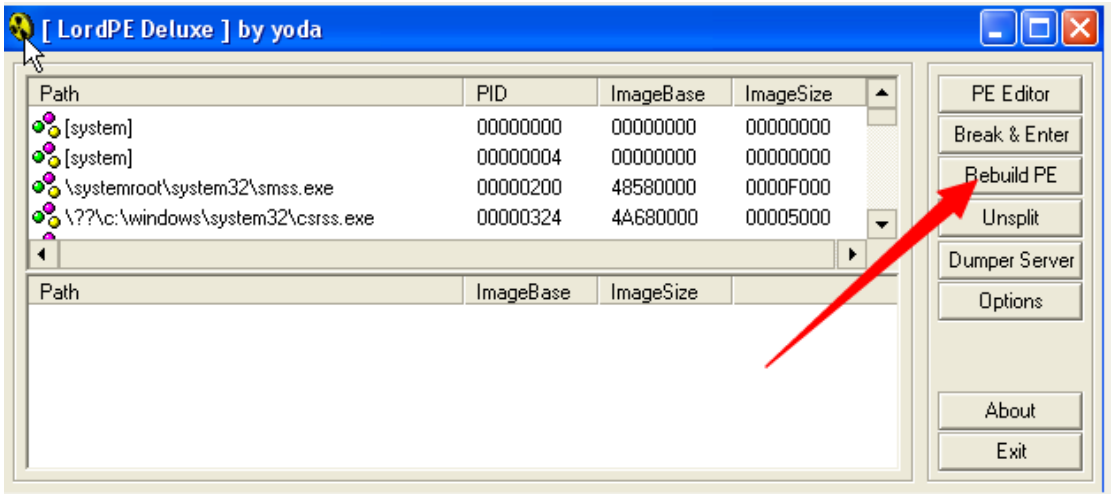


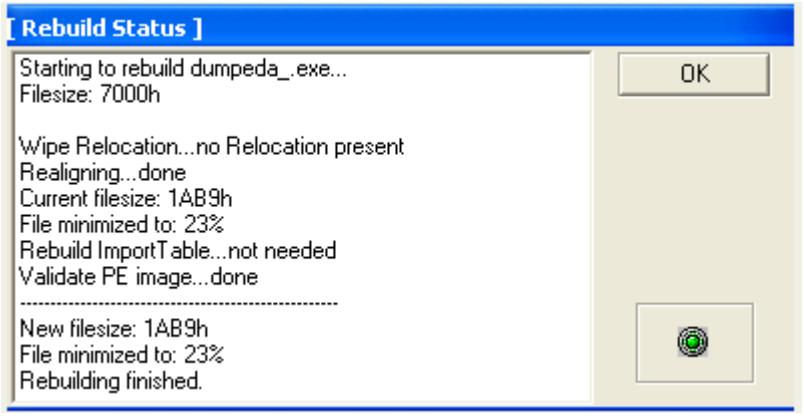
这里我们将 OEP,IAT 的起始地址以及大小填充到 IMP REC 中。接着单击 Get Imports 按钮。我们可以看到获取的 IAT 项都是有效的,啧啧!!!

接下来单击 Fix Dump 修复刚刚我们 dump 出来的文件。修复完毕以后直接运行起来,可以看到报错了。

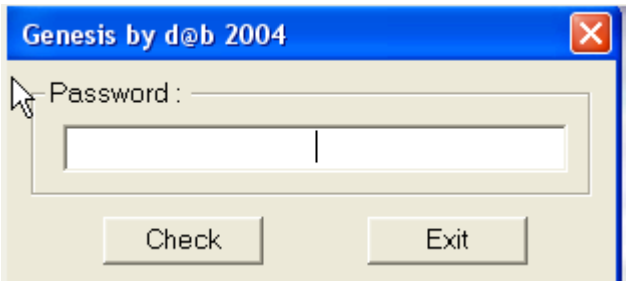


我们将其拖拽到 LordPE 中的 Rebuild PE 按钮上,这样就可以重建 PE 了。





重建 PE 完毕,直接运行起来,嘿嘿,完美运行。



看来我们的剑走偏锋还是有所成效的。总结:大部分壳的加密思路都是有共同之处的,但是有些壳会玩一些花招来抵御常规的脱壳思路,剑走偏锋往往能收到奇效,嘿嘿。