第四十章

本章在修复 PELock 的 IAT,AntiDump 之前,我们先来讨论知识点,第一个知识点我们上一章已经碰到了,只不过没有展开讲而已,第二个知识点就是如何绕过 PELock 或者类似软件对硬件断点的检测。

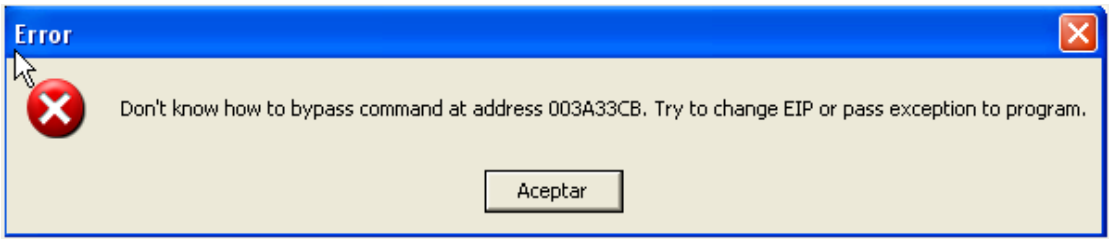
知识点 1 在上一章节我们遇到过,大家可能没有在意,大家是否还记得上一章节中,OD 加载目标程序以后,直接运行起来会弹出一个错误框,提示不知道该如何处理 xxx 处的非法指令。

为了解决这个问题,我们需要用到 Patched4 这个程序(又一个打过补丁的 OD),我们将其置于原 OD 所在的文件夹中,因为原版 OD 在处理非法指令的时候有个 BUG,就算我们勾选上调试选项中的忽略非法指令异常,当 OD 遇到非法指令的时候还是会弹出一个错误框,Patched4 这个 OD 修复了这个 BUG。

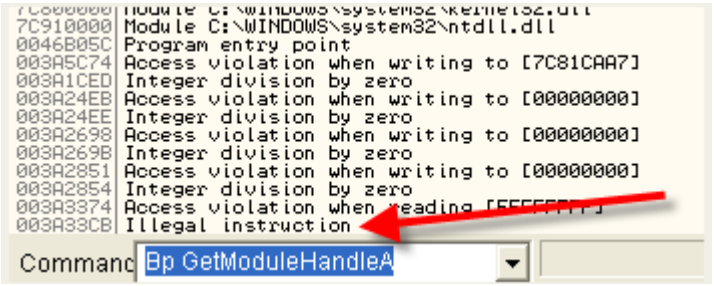
大家将 Patch4 这个 OD 跟原版 OD 放到同一个文件夹下面,下面我们分析 PELock 的时候需要用到。

好,再来给大家解释一遍(PS:老外比较细心,也比较啰嗦,嘿嘿)。

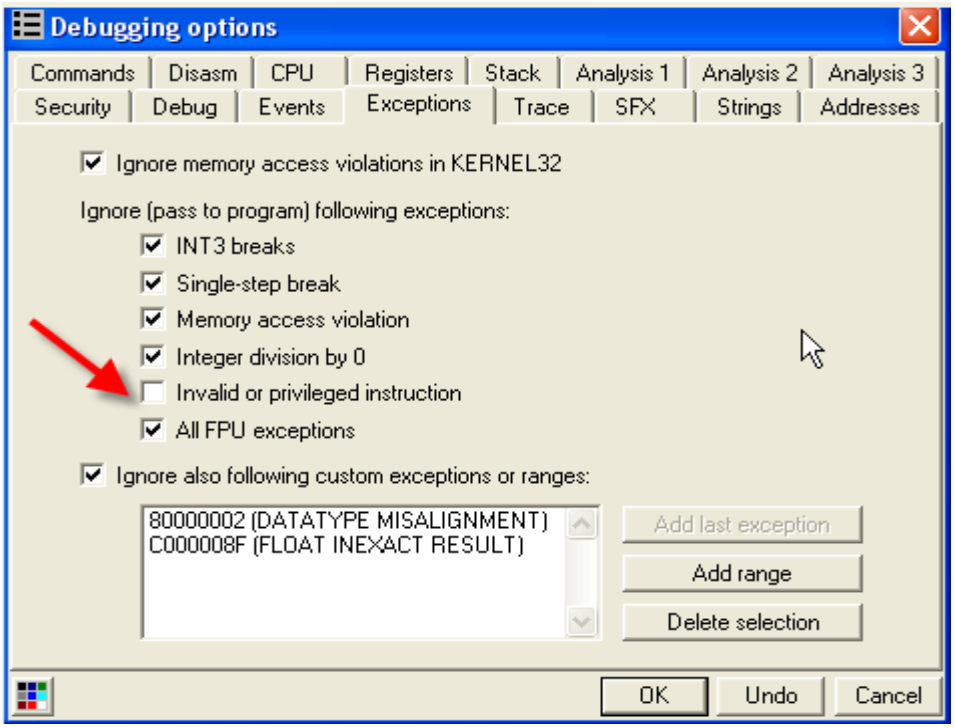
原版 OD 处理非法指令的时候有个 BUG,勾选上调试选项中忽略非法指令异常,加载 UnPackMe_PELock1.06,接着运行起来。



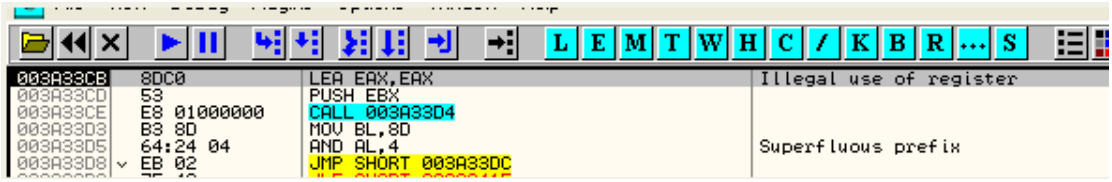
我们可以看到弹出了一个错误框,但是我们用 Patch4 这个 OD 来加载 UnPackMe_PELock1.06 的话,就可以完美运行。我们来看看日志信息。



我们可以看到最后一次异常是一个非法指令异常,现在我们不勾选的忽略非法指令异常选项。



这里我们去掉 Invaild or privileged instruction 这个选项的对勾,加载 UnPackMe_PELock1.06,运行起来。

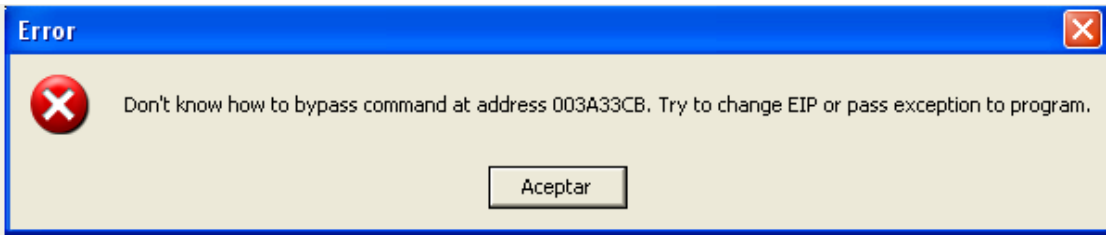


断在了异常处,如果我们按 Shift + F9 也可以继续运行下去。

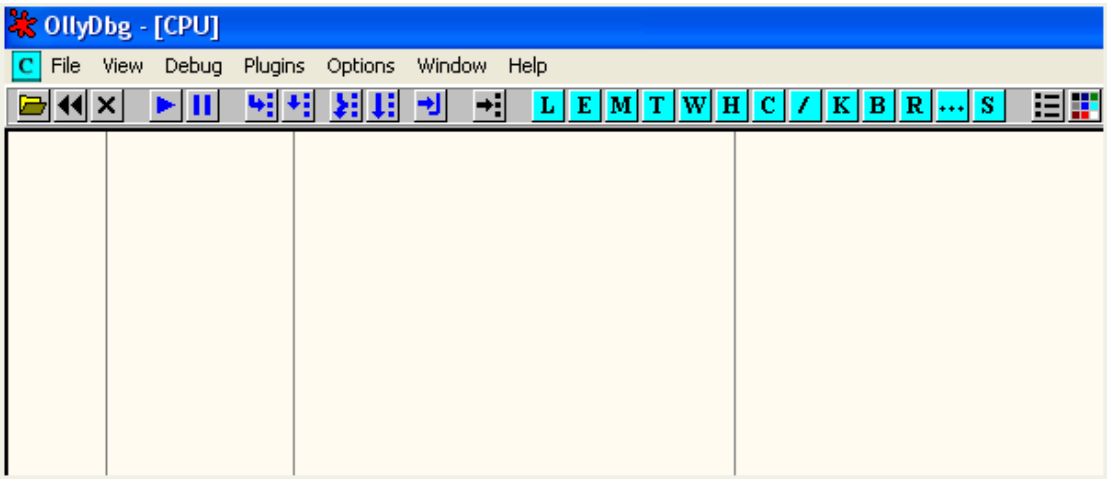
也就是原版 OD 是就算你勾选了忽略非法指令异常这一项,OD 并没有自动忽略。

如果是仅仅有一个这类异常,我们按一下 Shift + F9 也没什么。但是如果目标程序会产生 200 或者 300 个这样的异常,难道我们也手工去按 Shift + F9 不成(PS:连续按 Shift + F9 200 多次,那还不得疯掉哇,哈哈)。

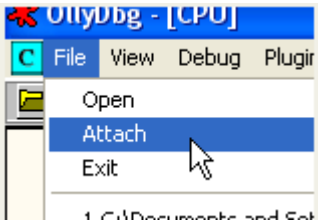
我们再次勾选上 Invaild or privileged instruction 这个选项,运行起来,再次弹出了错误框。

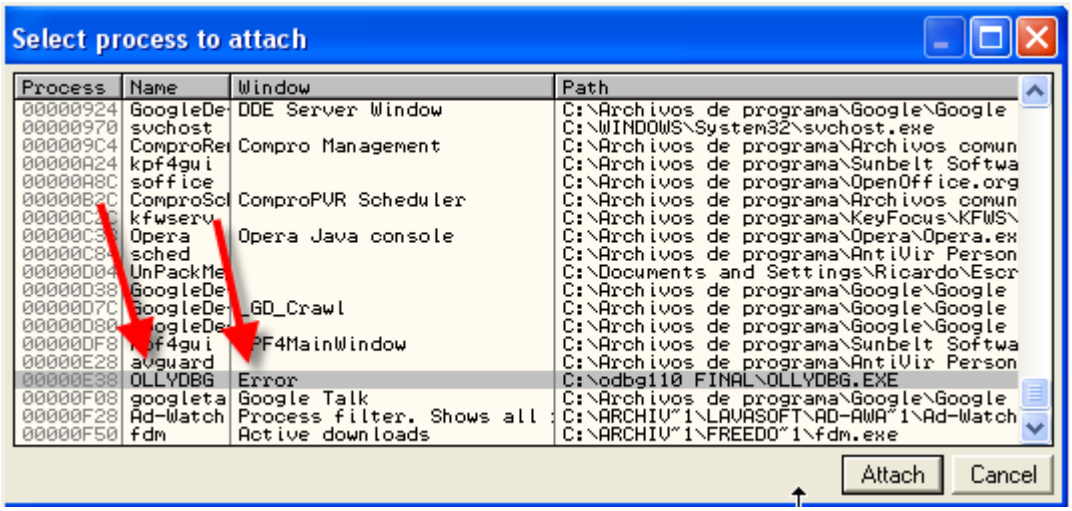


下面我们来尝试修复 OD 的这个 BUG,再新开一个 OD。

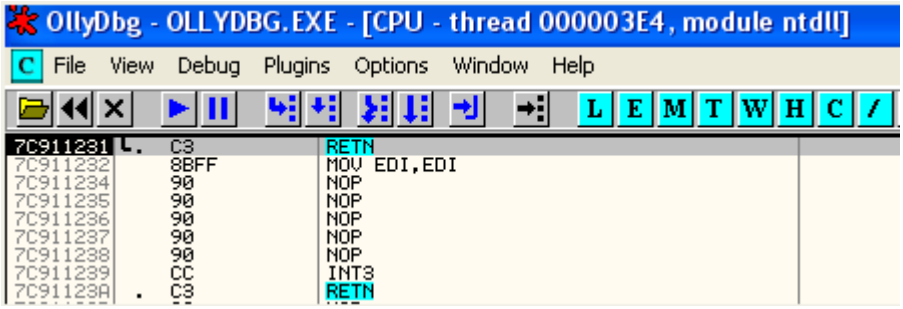


选择新的 OD 主菜单中的 File-Attach。定位到弹出错误框的 OD 所在的进程。





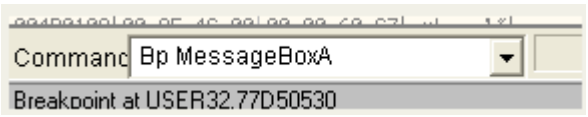
定位到弹出错误框 OD 所在的进程,单击 Attach(附加)。

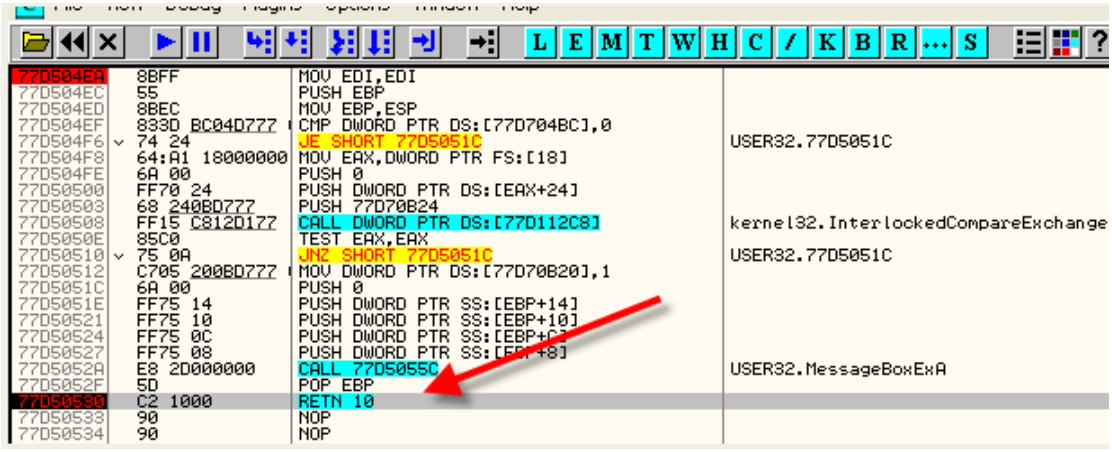


断在了这里,我们运行起来。

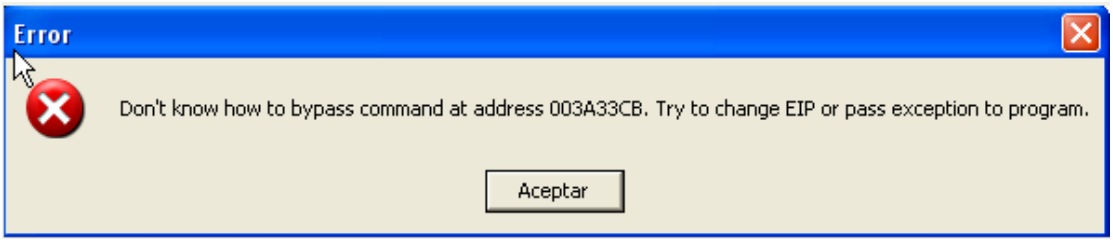
接着我们来看一下区段列表窗口。

现在我们可以看到 OllyDbg 的各个区段情况,接下来我们给 MessageBoxA 这个 API 函数设置一个断点。

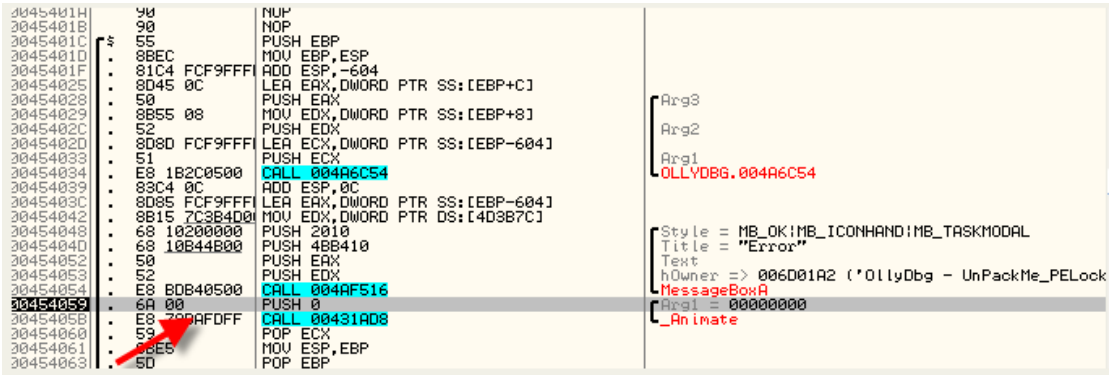




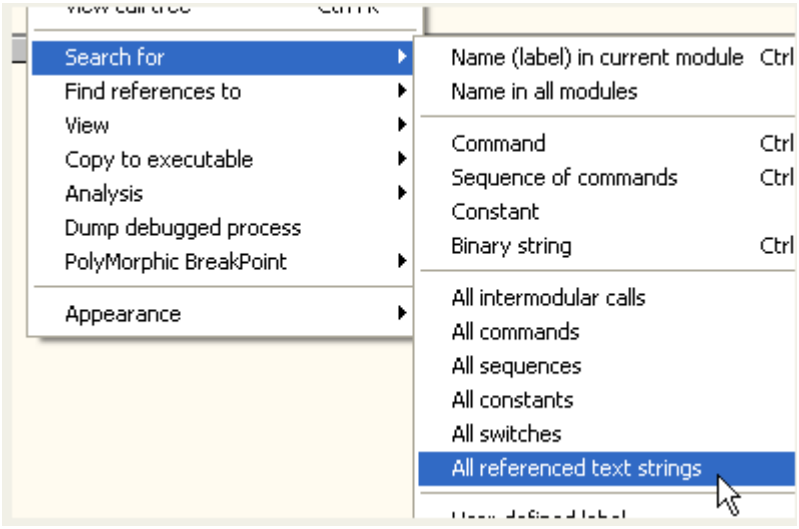
由于我们单击错误消息框上面的 OK 按钮,就到达 MessageBoxA 函数的 RET 指令处,我们给该 RET 指令处也设置一个断点。

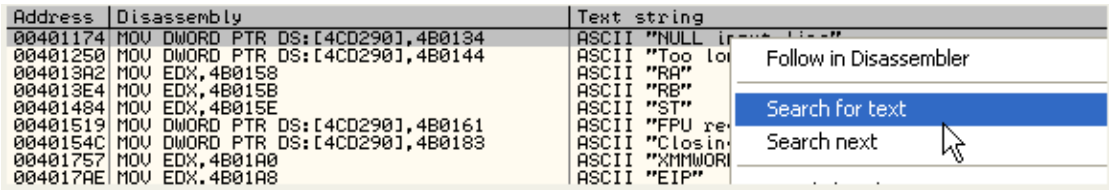


弹出错误消息框后,我们单击 Aceptar(OK)按钮。就会断在 MessageBoxA 这个 API 函数的返回 RET 指令处,我们单击 F7 键单步。

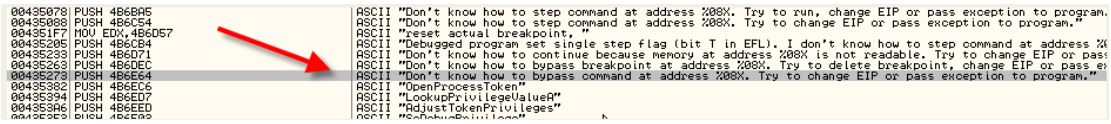


下面我们查看一下字符串列表,看看有没有消息框提示的错误信息字符串。

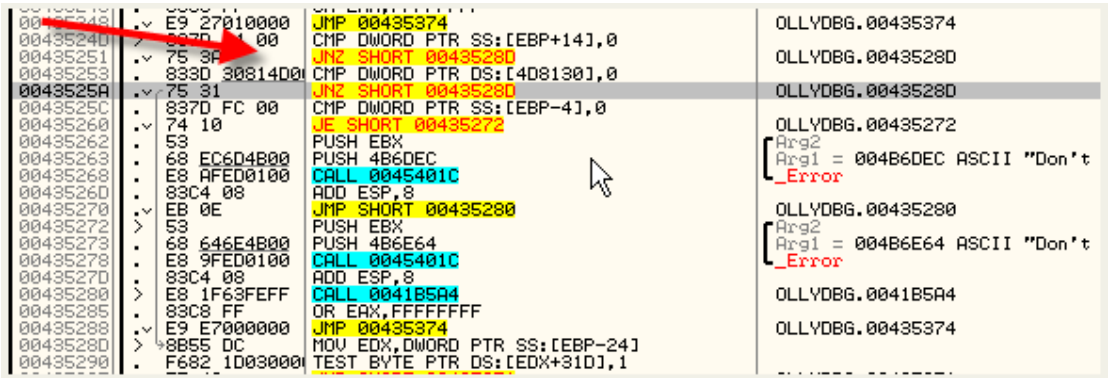




我们看看有没有字符串里面含有 bypass 这个单词。

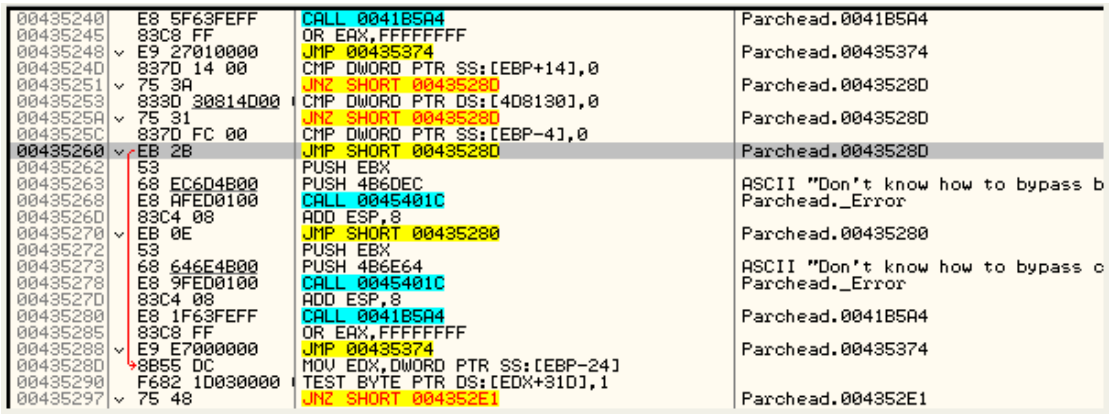


我们可以看到我们要找到字符串在这里,我们在这个字符串上面双击。



我们可以看到这里有几处 JNZ 可以跳过弹出错误消息框的代码,这里我们将 435260 处的 JE 修改为 JMP 43528D,保存到文件中。

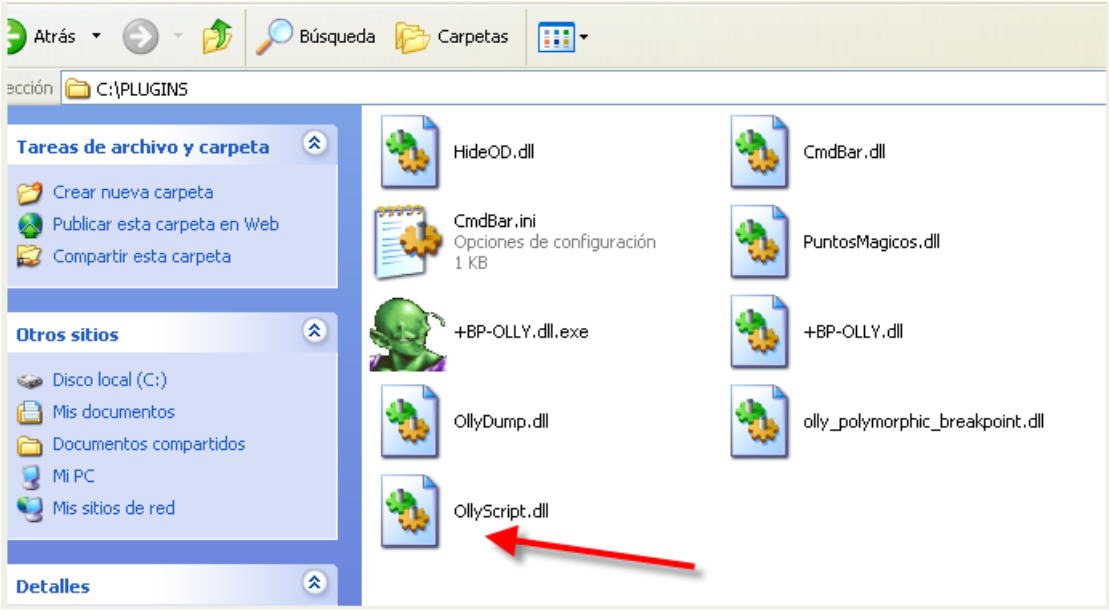
接下来我们看看 Patched4 这个 OD 该处的代码是怎么样的。



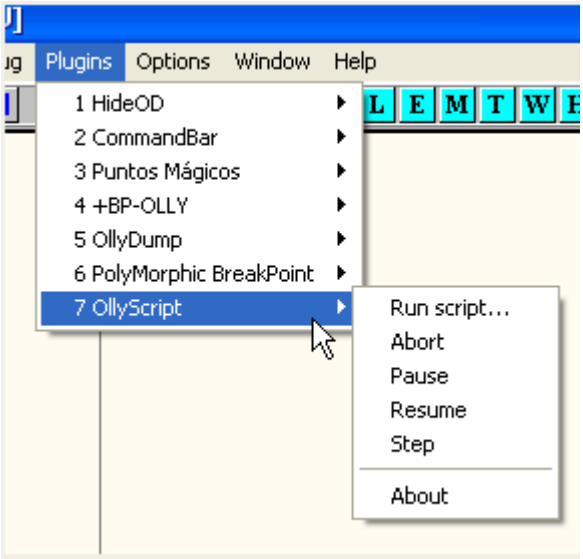
这里我们可以看到 435260 处的条件跳转被修改为无条件跳转,即修改为了 JMP 43528D,这样就能跳过填出错误消息框的代码。

这就是 Patched4 修复非法指令这个 BUG 的原理。大家可以自行给 OD 打补丁也可以直接用 Patched4 这个 OD。

知识点 2 就是要给大家演示如何编写脚本,我们需要用到 OllyScript0.92.dll 这个插件,将其放到 OD 的插件目录下。



打开 Patched4 这个 OD。

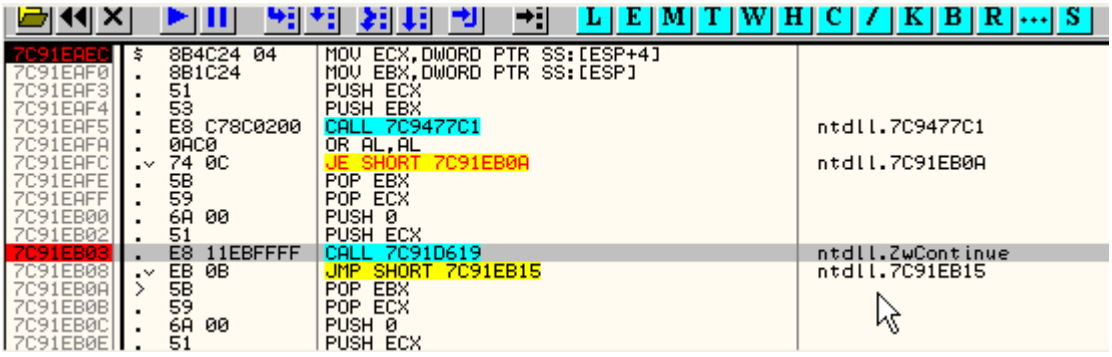


我们可以看到主菜单中出现 OllyScript 的插件子菜单项。

现在我们加载 UnPackMe_PELock1.06,下面我们编写一个简单的脚本让该壳无法检测到我们设置的硬件断点。

关于这个插件的使用说明大家可以参考插件文件夹下面的 readme.txt。

这个脚本的思路很简单:每次异常触发后都会断在 KiUserExceptionDispatcher,接下来就会由操作系统去调用异常处理函数,这个时候我们设置硬件断点可以被检测到,所以这个脚本要做的就是当前断在 KiUserExceptionDispatcher 时,清除掉硬件断点,当断在下面的 CALL ZwContinue 时再将硬件断点恢复。



由于这是我们要编写的第一个脚本,所以不会让大家写太难的。首先给 KiUserExceptionDispatcher 和 ZwContinue 分别设置断点。

现在我们开始编写脚本。一般来说脚本都有一个开始标签:

beginning:

设置这个标签可以让我们方便的 JMP beginning,接下来我们来设置硬件断点,举个例子:

bphws 12ffc4, “r”

bphws <BreakPoint HardWare Set> - 该命令可以对指定地址设置硬件断点,这里是对 12ffc4 设置硬件断点, “r” 表示读取,”w”表示写入,”x”表示执行。

beginning:

bphws 12ffc4,“r”

work:

设置完硬件断点以后我们再来添加一个 work 标签。

beginning:

bphws 12ffc4,“r”

work:

eob to_process

run

这里的 eob,<Execution On BreakPoint>-表示在下次中断发生时,跳转到指定标签处,这里该标签是 to_process,即当程序中断下来时将跳转到 to_process 标签处,接下来 run 命令表示运行起来。

这里就是关键的部分了,当前 OD 中断后,我们就需要检查是否断在了 KiUserExceptionDispatcher 入口处,所以我们在 to_process 标签下面来进行检查。

to_process:

cmp eip,7C91EAEC

je to_clear

cmp eip,7C91EB03

je to_recover

jmp to_final

我这里 7C91EAEC 是 KiUserExceptionDispatcher 的入口地址,当断在 KiUserExceptionDispatcher 入口处时,eip 为 7C91EAEC,所以跳转到 to_clear 标签处清除硬件断点,接下来当断在 CALL ZwContinue 指令处时,就跳转到 to_recover 标签处恢复硬件断点。

清除内存断点如下:

to_clear:

bphwc 12ffc4

jmp work

当前跳转到 to_clear 标签以后,通过 bphwc <BreakPoint HardWare Clear>命令删除 12ffc4 处的硬件断点,接着跳转到 work 标签处继续运行程序。

另外一个 to_recover 标签处是当断在 CALL ZwContinue 处时重新设置硬件断点的。

to_recover:

jmp beginning

接下来就是直接跳转到开头 beginning 标签处了,也就是将重新设置硬件断点。

接下来就是程序断在硬件断点处时,也就是 jmp final。

to_process:

cmp eip,7C91EAEC

je to_clear

cmp eip,7C91EB03

je to_recover

jmp final

最后是:

final:

MSGYN “是否继续?”

cmp $RESULT,1

je beginning

ret

这里的 MSGYN <Message Yes or No>命令会将指定消息,显示到一个对话框中,这个对话框上面有”是”,”否”按钮。如果点击”是”,保留变量$RESULT 等于 1,否则保留变量$RESULT 等于 0。如果我们想继续的话,就单击”是”,这样就会跳转 beginning 标签处继续执行脚本,如果我们不想继续,就单击”否”,这样就会直接 ret,脚本就执行完毕了,OD 就会停在硬件断点处。

整个脚本就是这样的:

beginning:

bphws 12ffc4,“r”

work:

eob to_process

run

to_process:

cmp eip,7C91EAEC

je to_clear

cmp eip,7C91EB03

je to_recover

jmp to_final

to_clear:

bphwc 12ffc4

jmp work

to_recover:

jmp beginning

final:

MSGYN “是否继续?”

cmp $RESULT,1

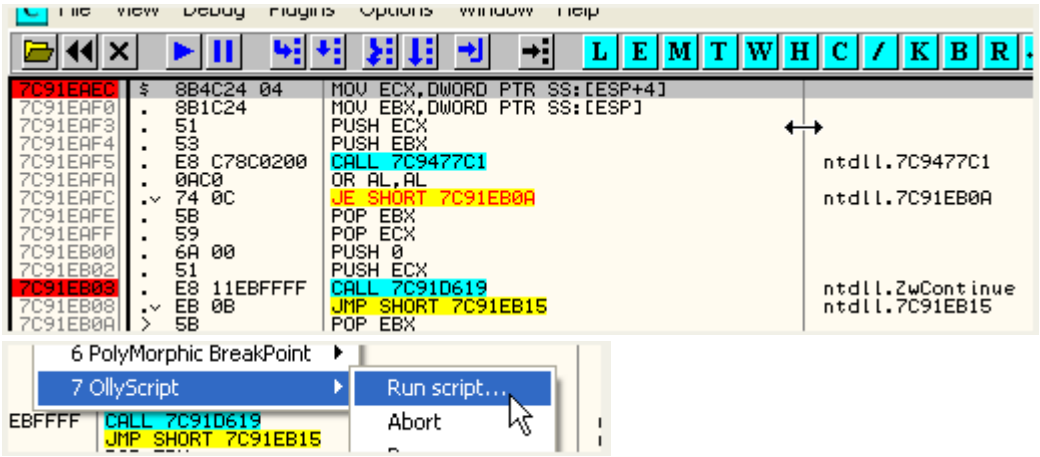
je beginning

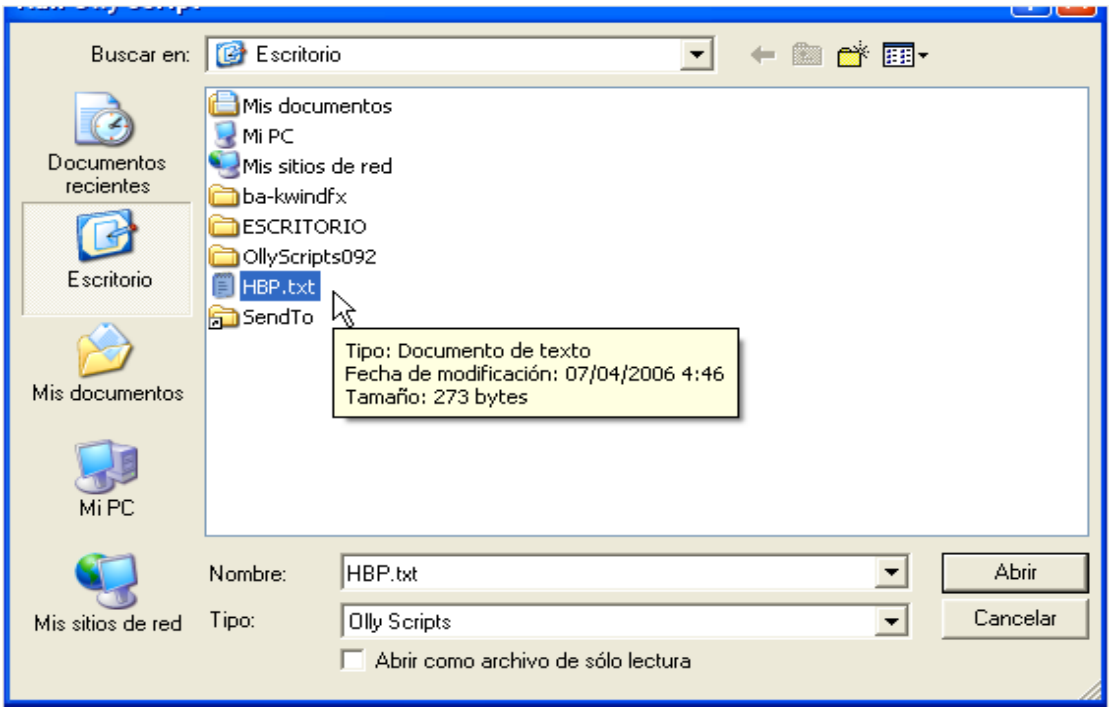
ret

该脚本的整个执行流程,大家可以参考下图中的解释,这里我就不再赘述了。

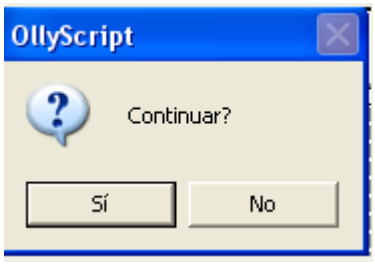
```
0000 beginning:
0001 bphws 12ffc4,"r"
0002 work:
0003 eob to_process          //当发生中断的时候,就会跳转到to_process标签处
0004 run
0005
0006 to_process:
0007 cmp eip,7C91EAEC
0008 je to_clear             //当断在KiUserExceptionDispatcher入口处时,跳转到to_clear标签处
0009 cmp eip,7C91EB03
0010 je to_recover           //当断在call ZwContinue处时,就跳转到to_recover标签处
0011 jmp final               //当断在硬件断点处时,就跳转到final标签处
0012
0013 to_clear:               //
0014 bphwc 12ffc4            //首先清除硬件断点
0015 jmp work                //接着跳转work标签处继续往下执行,等待下一次中断
0016
0017 to_recover:             //
0018 jmp beginning           //之前跳转到beginning标签处重新设置硬件断电
0019
0020 final:
0021 MSGYN "是否继续?"
0022 cmp $RESULT,1           //判断是否单击的是"Yes"按钮,如果是继续执行脚本,否则,直接返回,控制权交予OD,
0023 je beginning            //此时OD断在硬件断点处
0024 ret
0025
```

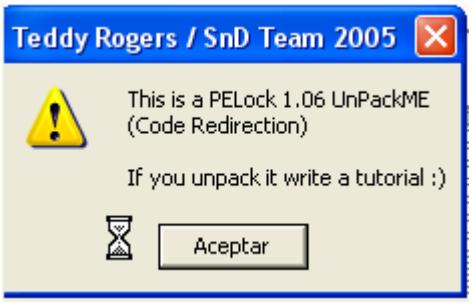
我们将上面的脚本保存到一个 txt 文本中,接着用 OD 加载 UnPackMe_PELock1.06,然后给 KiUserExceptionDispatcher 入口处和 call ZwContinue 指令处分别设置一个断点,接着在主菜单中找到 OllyScript 插件。



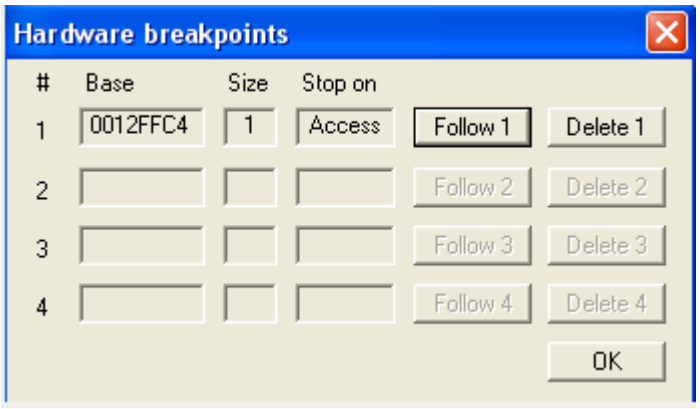


这里我们看到断在了硬件断点处,如果要继续执行脚本的话,单击 YES 按钮。





这里我们可以看到完美运行,我们来看看硬件断点窗口。



这里我们可以看到硬件断点设置成功了,也可以正常断下来,达到了我们预期的效果。

本章到此结束,下一章我们来介绍如何编写脚本修复 PELock 的 IAT。