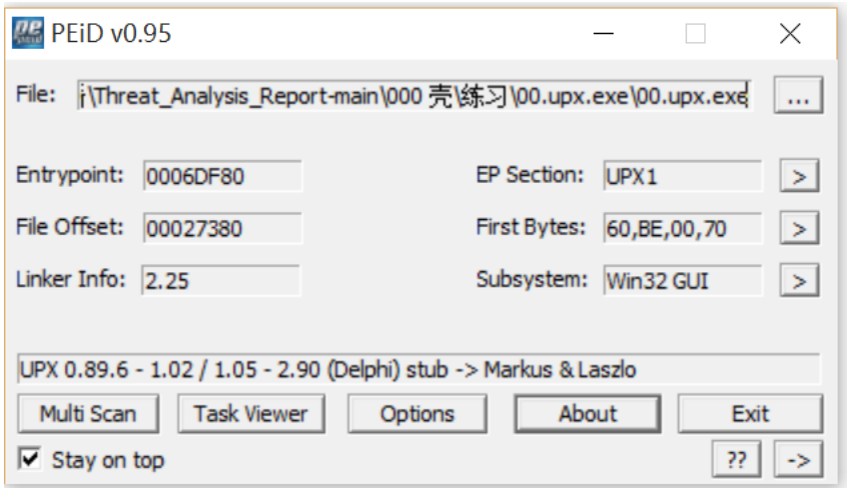
UPX 壳

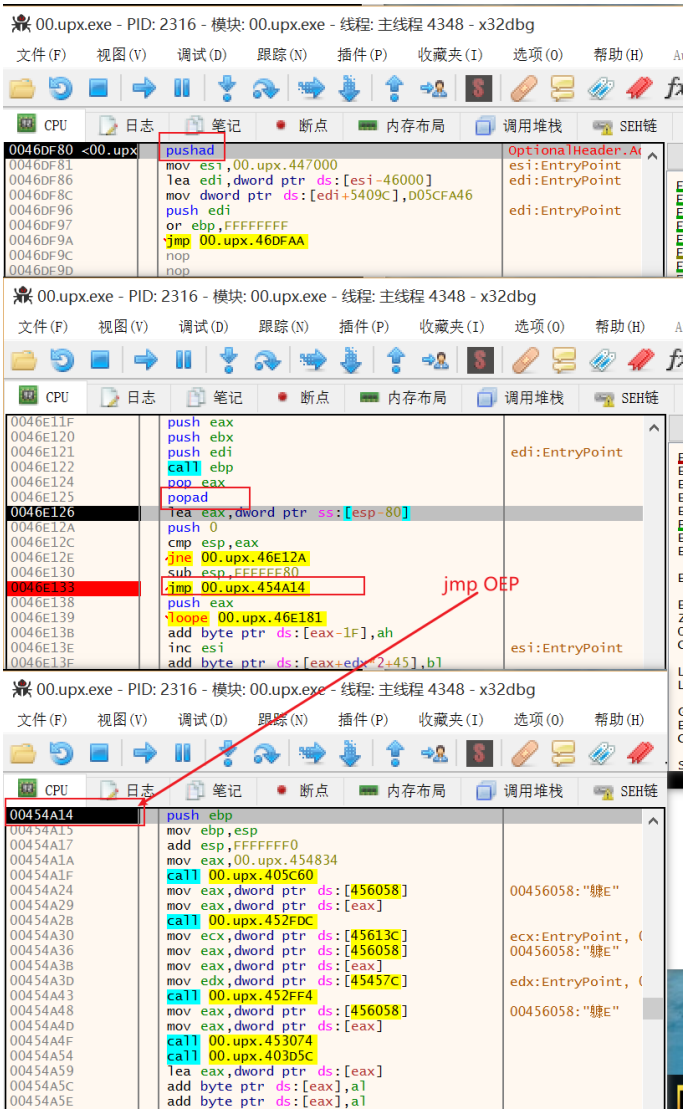
特征：UPX 压缩器 EP 代码被包含在 PUSHAD/POPAD 指令之间，并且跳转到 OEP 处的 jmp 指令在 POPAD 指令之后，在 jmp 指令处设置断点就能找到 OEP。
脱壳手法：UPX 脱壳工具直接脱壳、手动脱壳（找 OEP+修复 PE 导入表）。
脱壳示例：

1、UPX 脱壳工具直接脱壳：

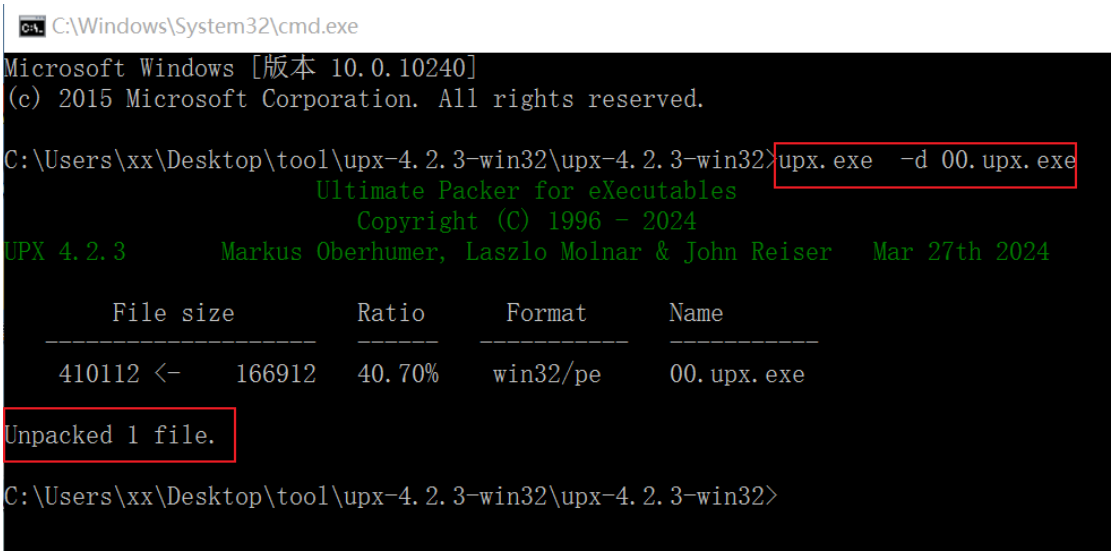
PEID 查壳：UPX0.89.6



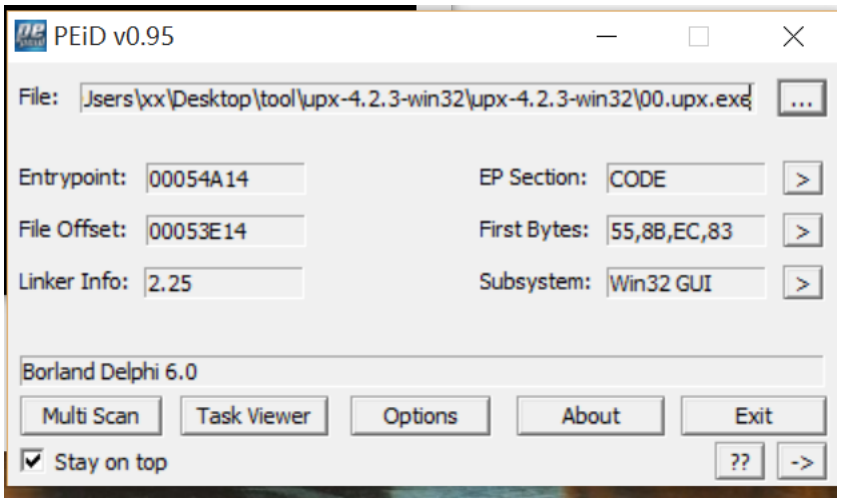
（此法可能因为壳变种问题导致下载的某个工具不能直接脱壳，可以下载对应版本或者多版本试试，或者换手动脱壳法!!!!）
脱壳前，运行程序会经历程序解压缩步骤，脱壳完成标志是跳转到 OEP 进行执行。



下载 UPX 解压缩工具，解压缩后使用-d 参数解压目标 exe(注：提前备份解压目标 exe 文件，解压缩会覆盖原文件。)



验证脱壳：PEID 检测无壳，此次脱壳成功。



2、手动脱壳（找 OEP+修复 PE 导入表）：

（1）、查看 exe 文件信息，ImageBase=00400000。

由下图可推出：

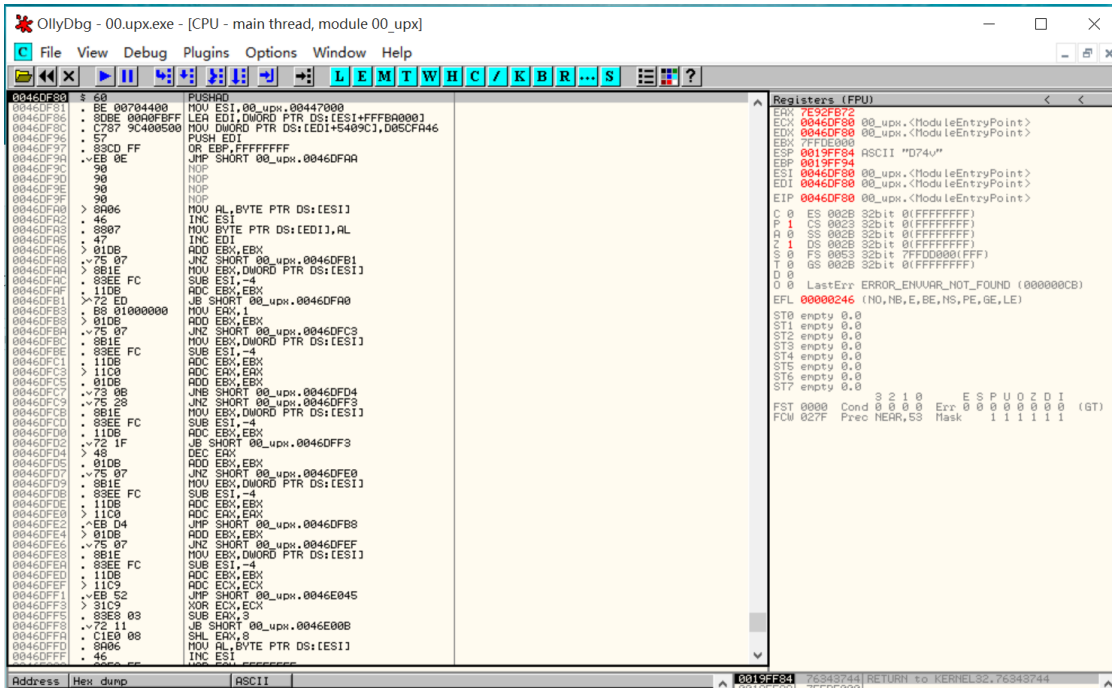
第一节区内存中范围：00401000-00447000

第二节区范围：004470000-0046f000

第三节区范围：0046f000-00471000

#	Name	VirtualSize	VirtualAddress	SizeOfRawData	PointerToRawData	R
0	UPX0	00046000	00001000	00000000	00000400	
1	UPX1	00028000	00047000	00027200	00000400	
2	.rsrc	00002000	0006f000	00001600	00027600	

（2）OllyDbg 调试 exe 程序：找 OEP



由上图可知 EP 地址为 0046DF80，该处在第二节区中。实际压缩的源代码在 EP 地址

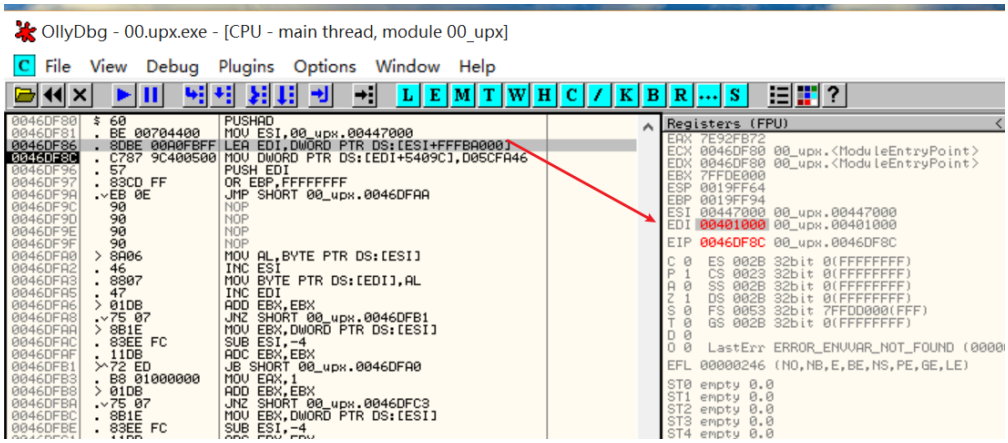
（0046DF80）上方。

PUSHAD //将 EAX~EDI 寄存器的值保存到栈，

MOV ESI,00_upx.00447000 //设置第二节区起始地址到 ESI

LEA EDI,DWORD PTR DS:[ESI+FFFBA000] //将第一节区的起始地址 00401000 存到 EDI；0x00447000+FFFBA000=00401000。

UPX 第一个节区仅存在于内存。该出即是解压缩后保存源文件代码的地方。【第一个节区内存中大小和文件中大小差别很大，其他节区文件内容加载到内存中大小差别不大】



调试时像这样同时设置 ESI 和 EDI，就能预见从 ESI 所指缓冲区到 EDI 所指缓冲区的内存中发生了复制。此时从源地址（ESI）读取数据，解压缩后保存到目的地址（EDI）。**此处调试目标是，跟踪全部 UPX 的 EP 代码，并找到原程序的 EP 代码。此过程有点长此文就不详细跟了。**

下面根据提供一种根据 UPX 特征快速定位 OEP 的方法。

UPX 壳特征一:UPX 压缩器 EP 代码被包含在 PUSHAD/POPAD 指令之间，并且跳转到 OEP 处的 jmp 指令在 POPAD 指令之后，在 jmp 指令处设置断点就能找到 OEP 了。那么如何找到 POPAD 指令呢?

因为 PUSHAD 将 EAX~EDI 寄存器的值保存到栈，所以在运行 PUSHAD 指令后找到 ESP 地址并设置硬件访问断点，当解压缩完成时会恢复到解压缩前的堆栈情况，会再次访问 ESP 地址而中断在此处。

总结： **在调试器中找到 PUSHAD 指令。**

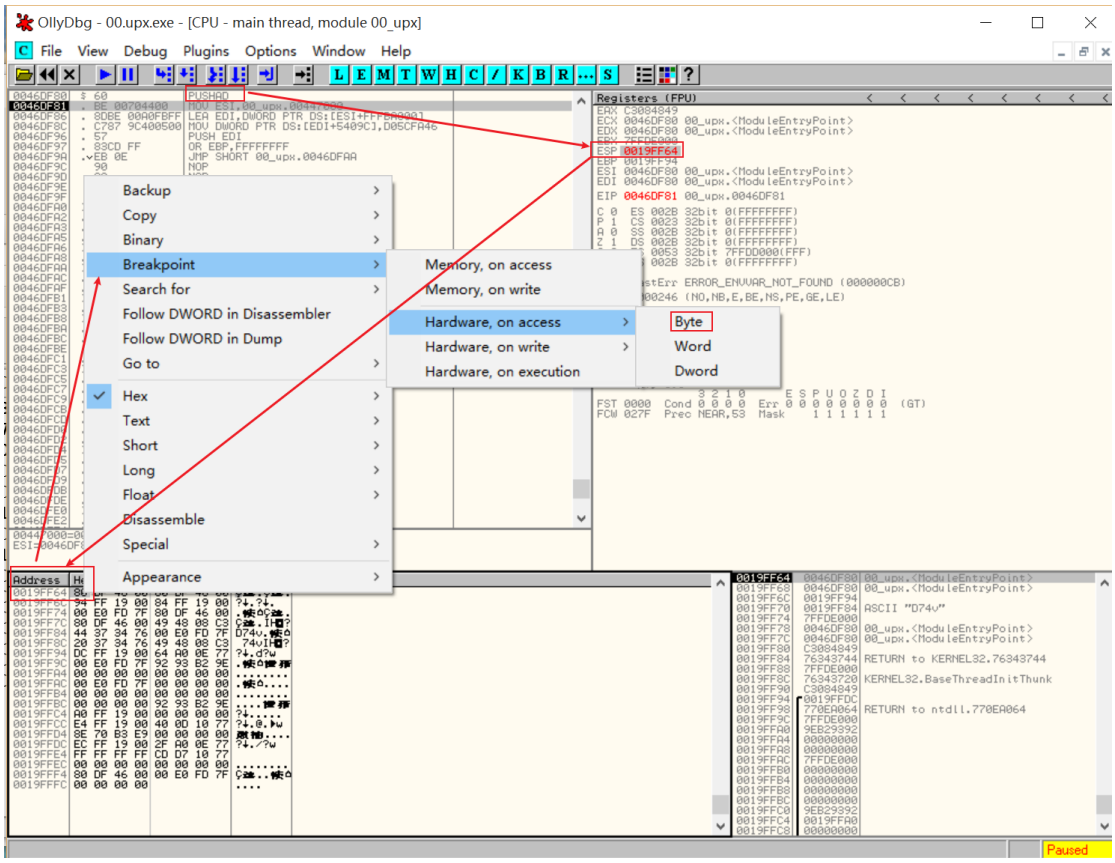
记录 PUSHAD 执行后的 ESP 值。

设置一个硬件访问断点，监视 ESP 处的内存地址。

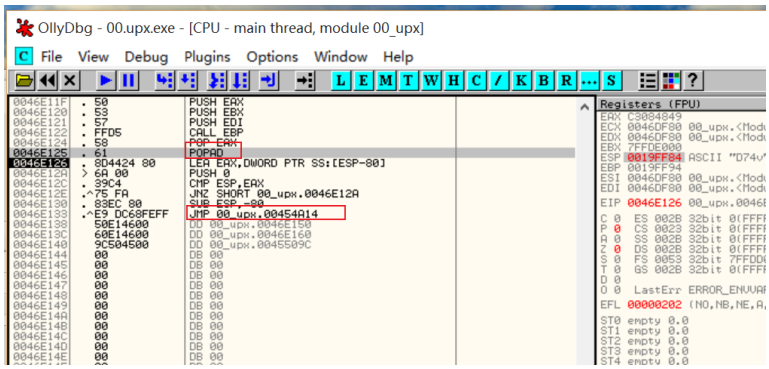
当访问断点触发时，调试器会在 POPAD 恢复寄存器时中断。

从 POPAD 开始单步执行，直到找到跳转到 OEP 的 jmp 指令。

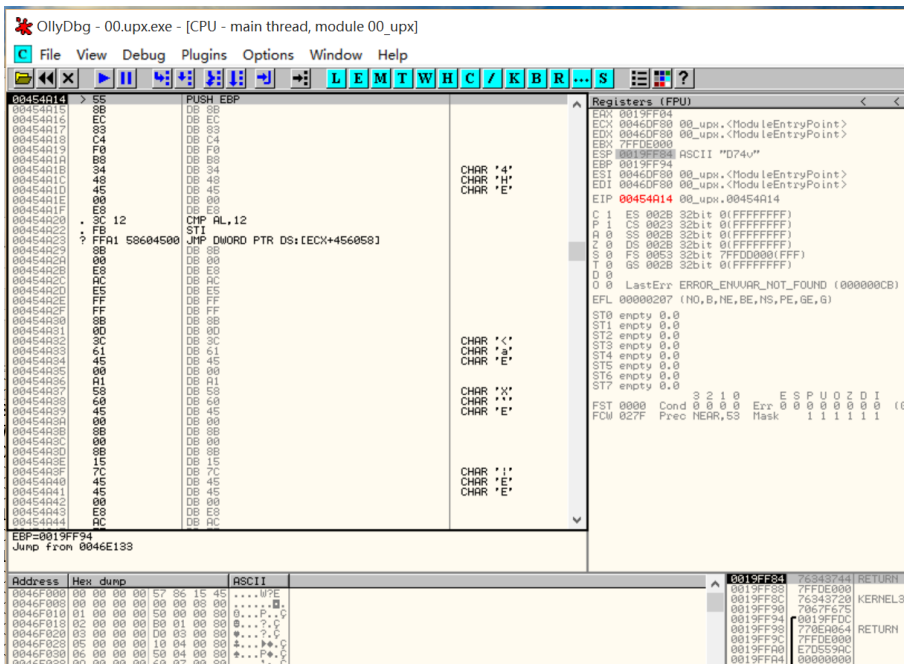
这种方法依赖于 UPX 壳的标准行为，但也要注意某些变种 UPX 壳可能修改了解压流程，添加了额外的反调试或混淆技术，可能会影响这些步骤的有效性。



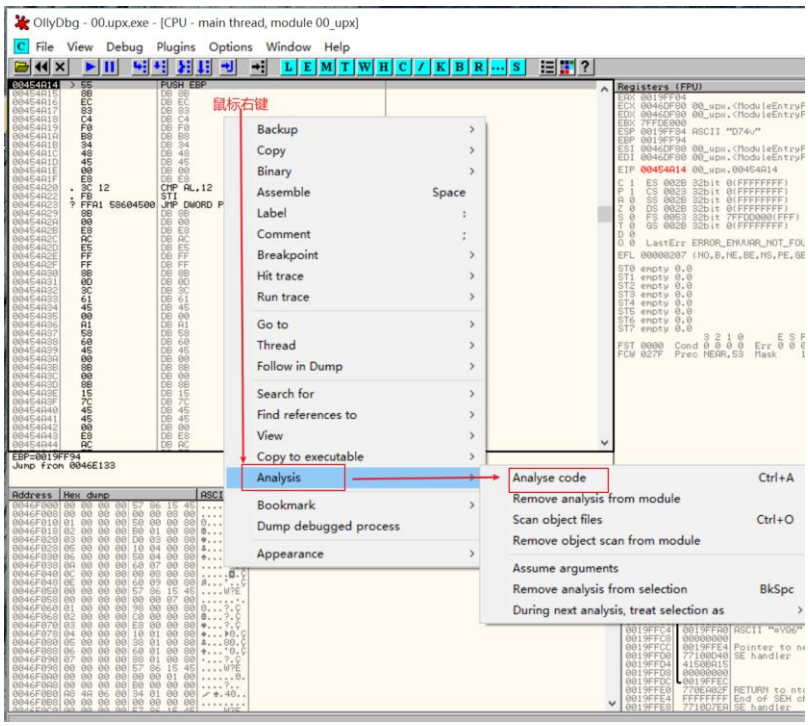
F9 直接运行到硬件访问断点处。跟踪后面 jmp 命令。



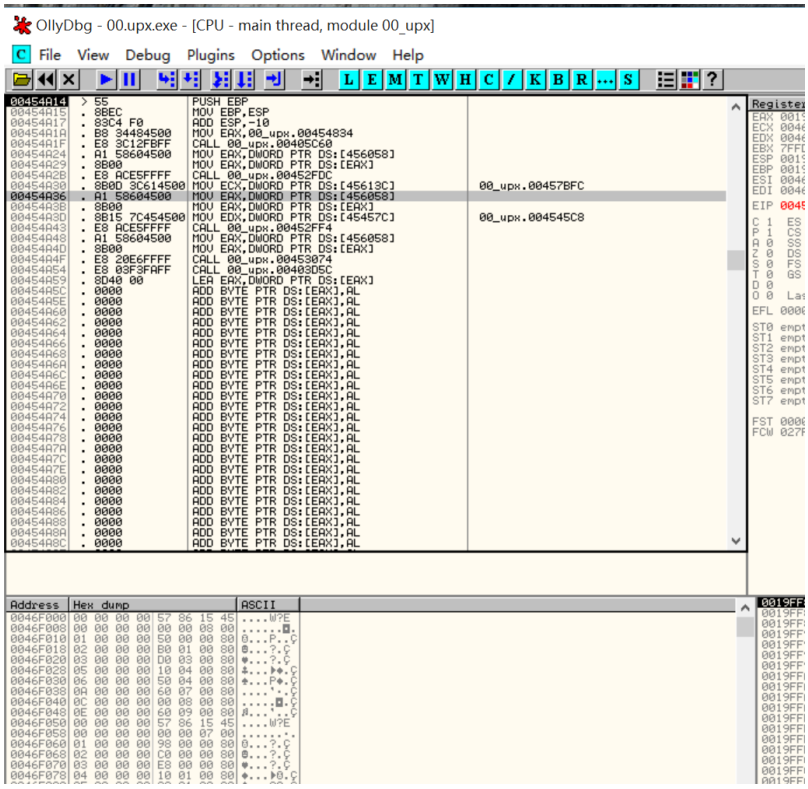
跳到 OEP：汇编指令未被正确识别，怎么办？



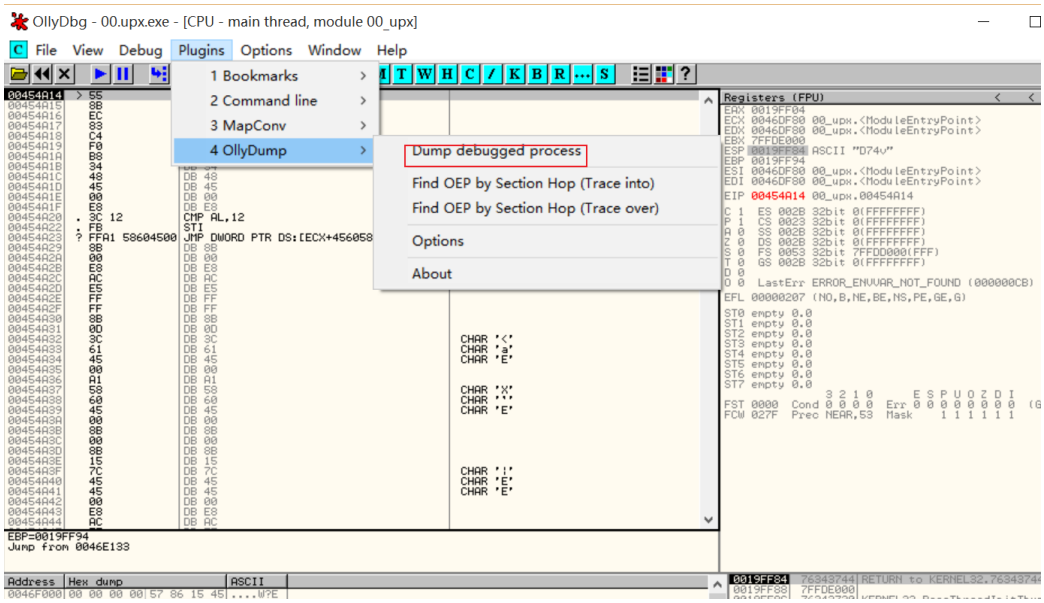
手动反汇编代码识别:



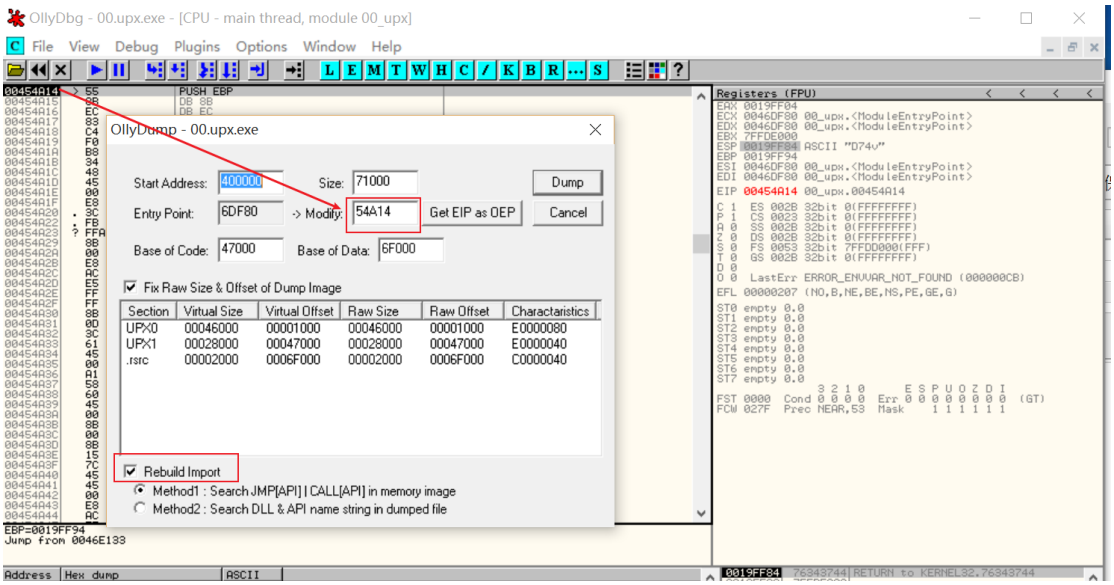
手动反汇编代码识别后:



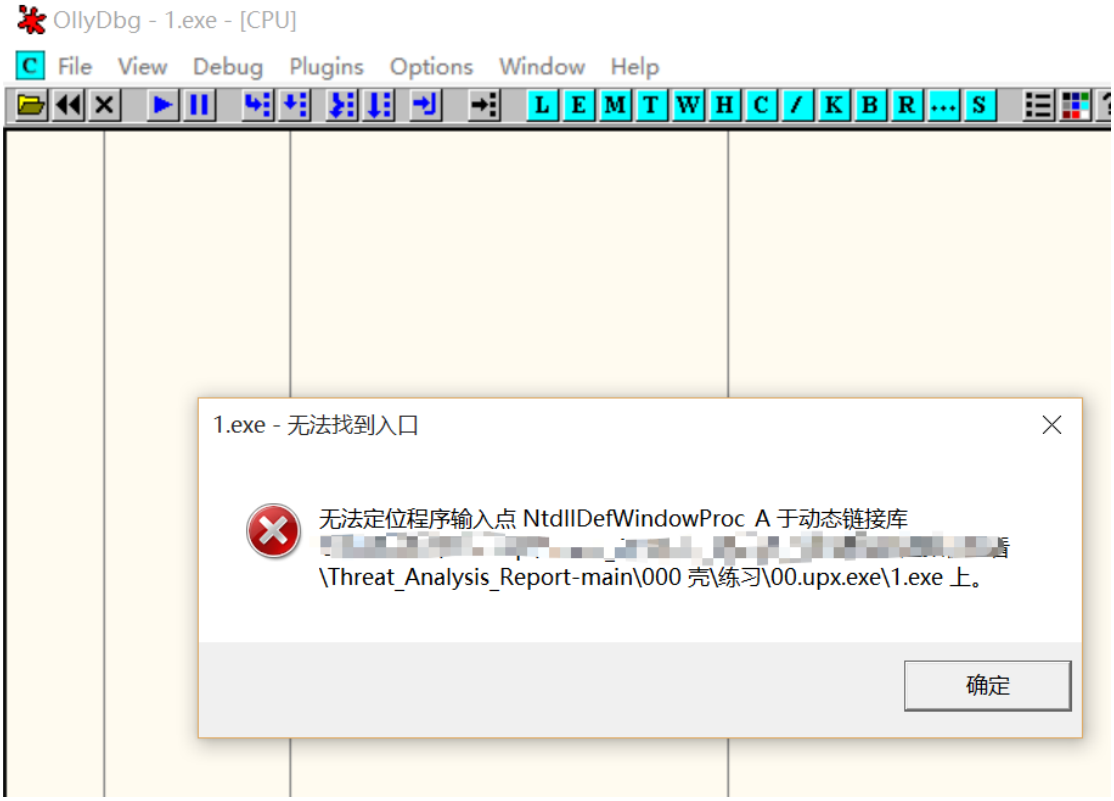
插件 dump 下内存:



设置导出文件 OEP 并选择重建导入表:



调试运行报错：无法找到入口



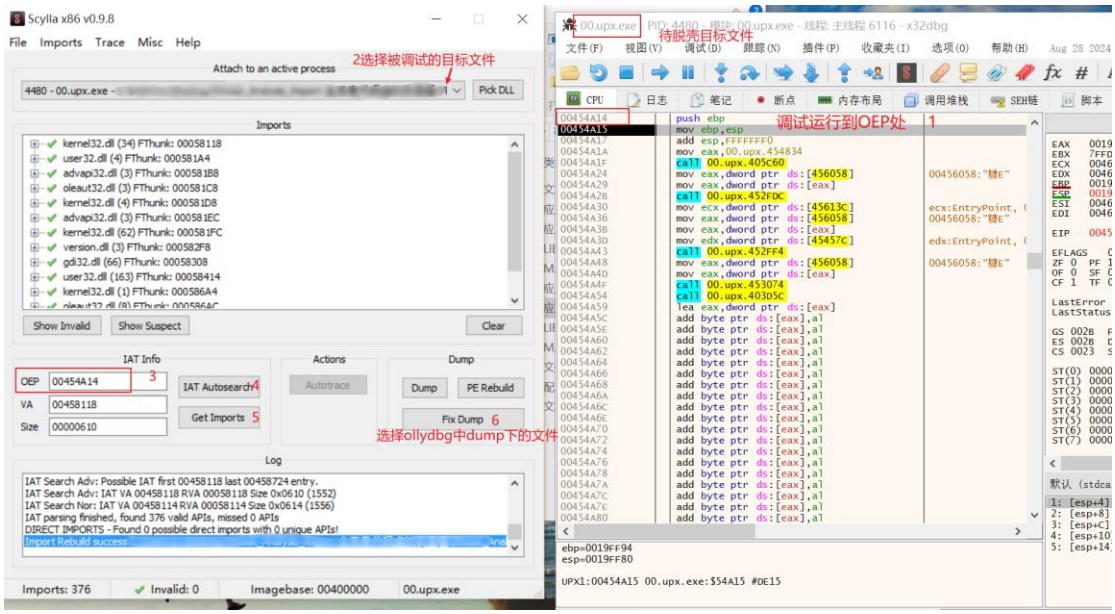
修复 PE 文件：Scylla 工具+Ollydbg

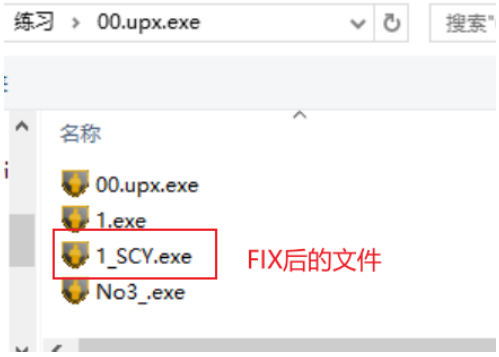
此处被脱壳的程序直接 OllyDbg 插件 dump 下来的文件并不能正常被 OD 调试器打开，需要使用 Scylla 工具进行 PE 导入表修复。

为什么要使用 Scylla 来修复 dump 文件？

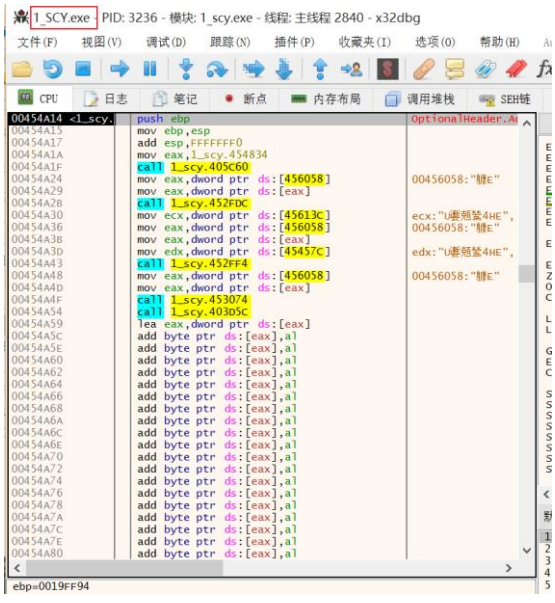
当使用 OllyDbg 插件将解压缩后的程序内存 dump 下来后，dump 出的文件导入表往往是不完整的。加壳程序通常会在解压缩的过程中动态重建导入表，壳本身可能会破坏或覆盖原始的导入表。因此，dump 出来的文件需要经过修复才能正常运行。

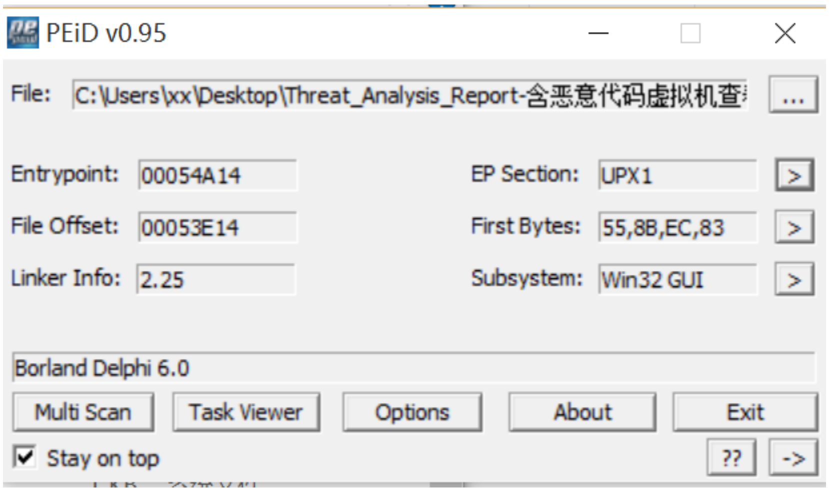
Scylla 是一个强大的 PE 导入表修复工具，它可以在 dump 出的文件中自动搜索并修复损坏的导入表。它通过在程序运行时动态扫描导入表，找到原始的导入并将其修复到 dump 文件中。（即 **Scylla** 是通过扫描内存，重建正确的导入表，使 dump 文件成为一个完整的 PE 文件。）





脱壳成功：OD 调试器正常打开运行到 OEP 处，PEID 检测无壳。




PEiD v0.95
File: C:\Users\xx\Desktop\Threat_Analysis_Report-含恶意代码虚拟机查
Entrypoint: 00054A14
File Offset: 00053E14
Linker Info: 2.25
EP Section: UPX1
First Bytes: 55,8B,EC,83
Subsystem: Win32 GUI
Borland Delphi 6.0
Multi Scan
Task Viewer
Options
About
Exit
Stay on top
??
->