2021/03/29_壳_第1课_压缩壳的原理

笔记本: 壳
创建时间: 2021/3/29 星期一 10:07
作者: ileemi

- 课程安排
- 压缩壳的原理
- 压缩壳工具
- 压缩壳的脱壳步骤

课程安排

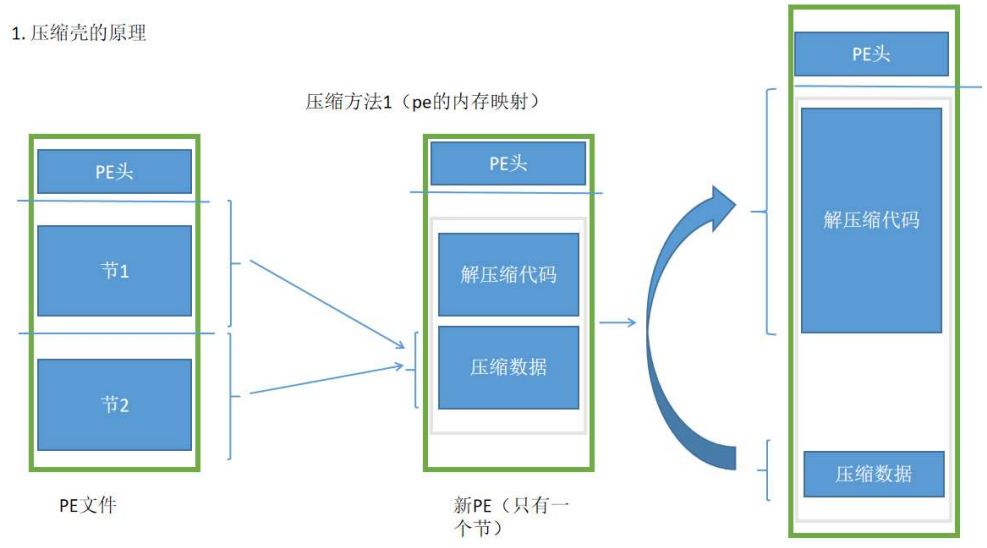
1. 加壳和脱壳的原理
2. 压缩壳实现之加壳部分
3. 压缩壳实现之shellcode部分
4. 样本脱壳实战

压缩壳的原理

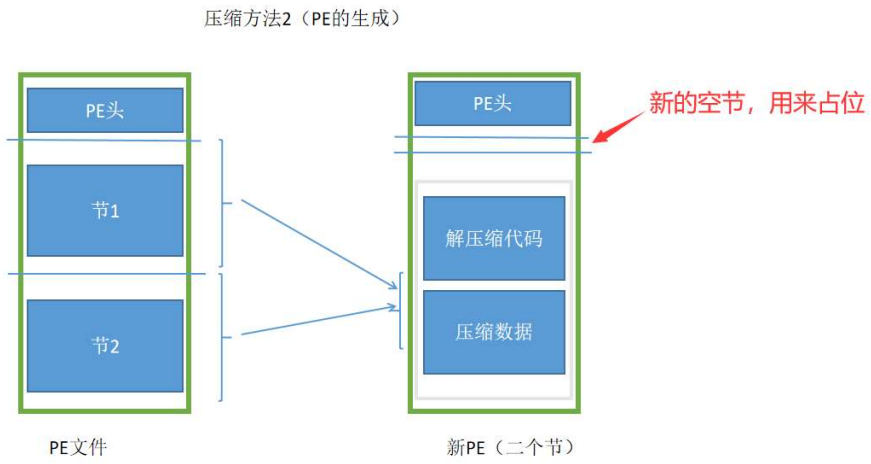
> 壳 --> 保护 可执行文件（* .exe、 * .dll、* .sys）。

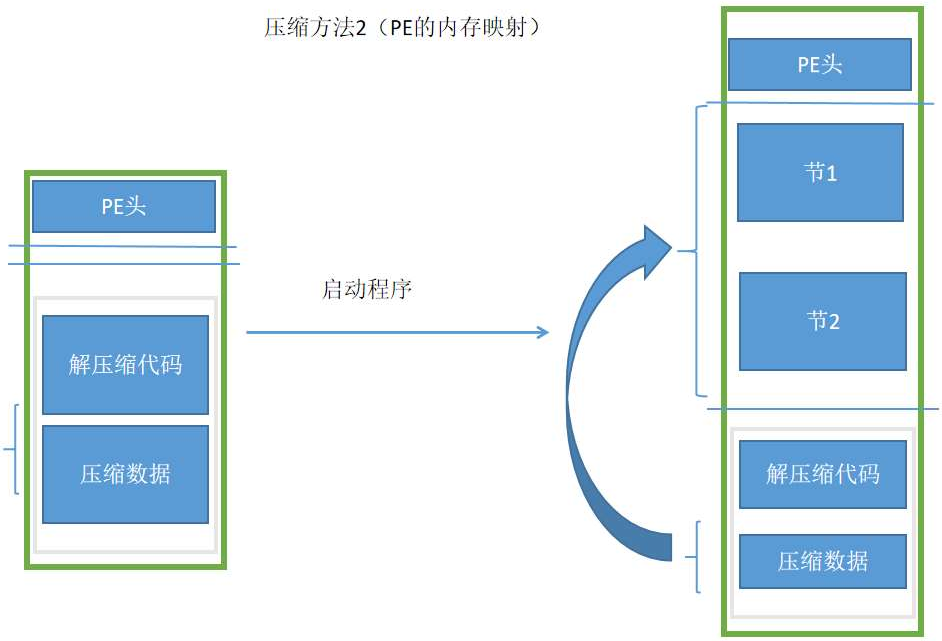
壳的分类:
压缩壳 - 内部使用压缩算法（注重文件的体积）
加密壳 - 内部使用加密算法（注重文件的加固程度）

1. 下图的压缩方法在讲新PE运行加载到内存中时，解压缩代码的内容回被压缩代码覆盖:



2. 方法2，在PE头和解压缩代码之间申请一个新的空节，用来占位，程序运行的时候，将解压缩的数据保存到这个节中：



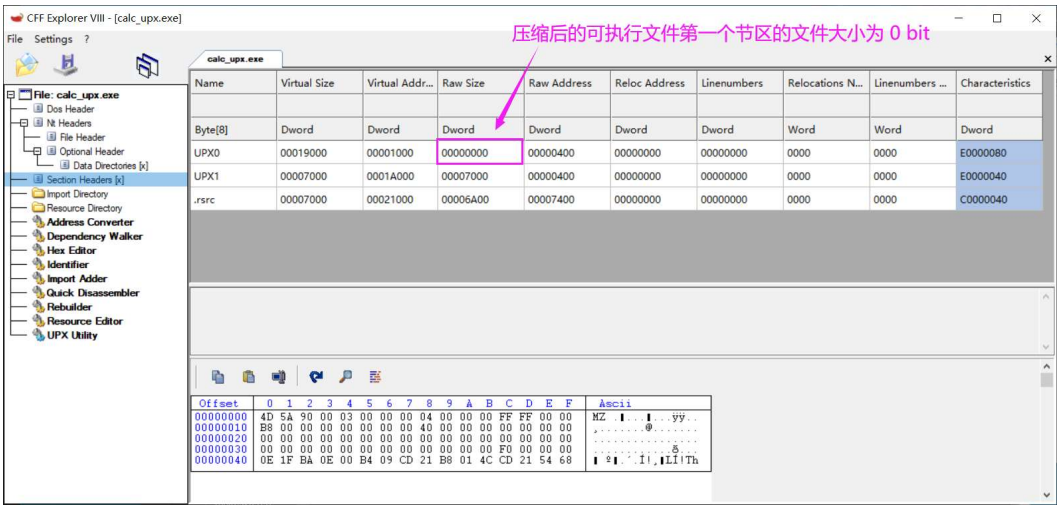


压缩壳工具

> upx：跨平台开源的一款可执行程序文件压缩器。压缩过的可执行文件体积缩小50%~70%，这样减少了磁盘占用空间、网络上传下载的时间和其它分布以及存储费用。

命令解析：
-1（compress faster）：压缩速度更快
-9（compress better）：压缩率更高
-d（decompress）：解压缩

使用示例示例：`upx -9 -o calc_upx.exe calc.exe`



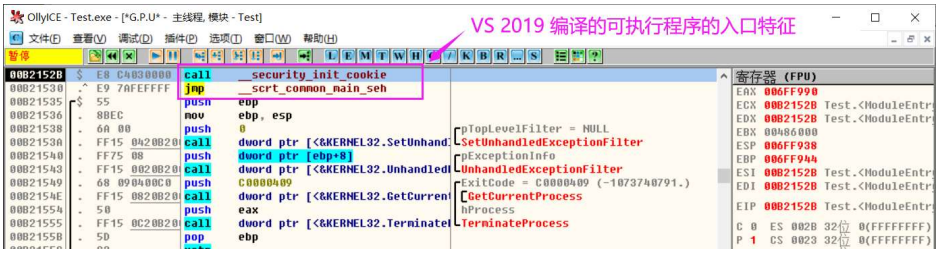
压缩壳的脱壳步骤

1. 查找OEP

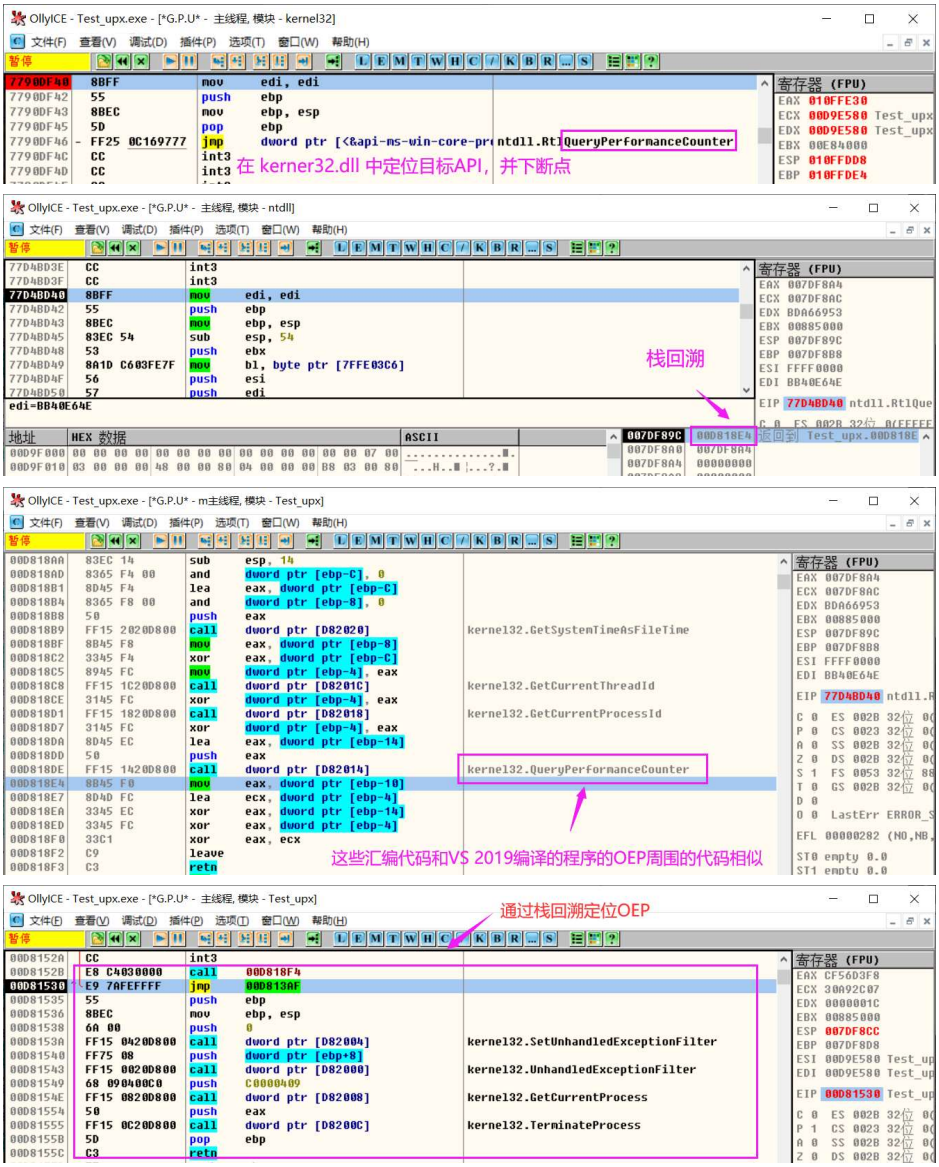
   识别OEP：靠经验（分析市面上流行的编译器生成的可执行文件，分析其入口点的特征（对应的汇编代码））

   手法：

   - ESP定律：为了保证程序执行壳的代码前的寄存器环境要和壳执行完毕后跳转到OEP处的寄存器环境一致，所以在执行加壳代码前就需要保存寄存器环境，执行到OEP代码前需要恢复寄存器环境
   - API： 根据入口点的特征定位OEP。示例：VS2019编译的程序在OEP处回调用一些API，在加壳的程序中通过定位模块中的对应的API来找到程序的OEP，如下图所示：



   在加壳的程序中，找到对应模块中的API，并下一个断点，运行程序，等待断点断下后，通过栈回溯找到程序的OEP:



- 单步跟踪（步过循环，只向下跳转）：对抗中遇到反调试，上面的两个方法不能使用，就需要单步过掉反调试代码。遇到大的地址跳转需要注意目标地址是否是OEP。

2. Dump：dump 后的可执行程序运行时如果出现一些类似 "无法定位程序输出点 xxx 函数" 于动态链接库等问题，最好的解决办法就是 32位程序在32位系统中进行Dump，64位程序放在64位系统中进行Dump 。

3. 修复PE：修复导入表可使用 "ImportREC" 工具。

IAT 加密：

查看其加密方式，观察其怎样获取API地址的。