实验一 壳保护实验

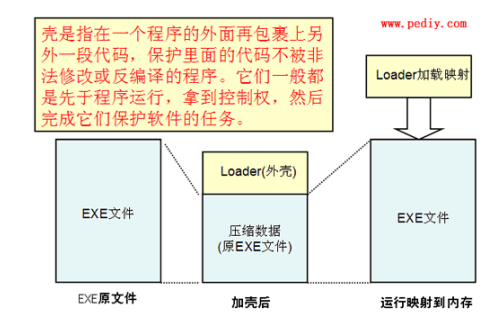
1. 实验背景

壳是在一些计算机软件里也有一段专门负责保护软件不被非法修改或反编译的程序。它们一般都是先于程序运行、拿到控制权，然后完成它们保护软件的任务。

UPX 是一个开源的，免费的可执行程序压缩壳。

IDA 是一款逆向必用工具，强大的反编译能力。

1. 实验目的
2. 理解壳保护技术的原理；
3. 熟悉常用加脱壳工具upx的使用方法；
4. 熟悉加壳、脱壳流程，学会手工脱壳的基本流程以及注意事项。
5. 实验原理



**脱壳方法**

1. 使用脱壳工具

识别壳： exeinfoPE、PEiD、Stud\_PE；

脱壳器：专用：UnPecomp2、untelock；

通用：Quick Unpack、linxerUnpacker。

1. 手动脱壳

1）找原程序入口点 OEP（Original Entry Point）

**一步到位法**：查找popad指令，找到匹配处，确认是壳解压完毕即将跳转到OEP部分，只适用于极少数压缩壳。

**内存镜像法**：内存镜像法的原理在于对于程序资源段和代码段下断点，一般程序自解压或者自解密时，会首先访问资源段获取所需资源，然后在自动脱壳完成后，转回程序代码段。这时候下内存一次性断点，程序就会停在OEP处。

方法步骤：

a.选项->调试选项->异常->勾选所有的忽略异常

b. Alt+M快捷键查看内存映像，对 .rsrc/.text段设置内存访问断点，执行至断点

c. 对.text段第一字节设置硬件执行断点，Shift+F9运行至断点最后异常法：最后一次异常法的原理是，程序在自解压或自解密过程中，可能会触发无数次的异常。如果能定位到最后一次程序异常的位置，可能就会很接近自动脱壳完成位置。

2）DUMP（内存映像转储）

应该在壳加载器执行完毕后，程序在内存中完整还原，且原程序指令未被执行，程序状态未受影响的情况下DUMP。

3）重建PE文件（IAT）

压缩壳一般不破坏TAT（Import Address Table），加密壳中原始程序IAT被劫持到壳加载器相应函数。

1. 实验步骤及主要截图
2. **逆向分析原程序**

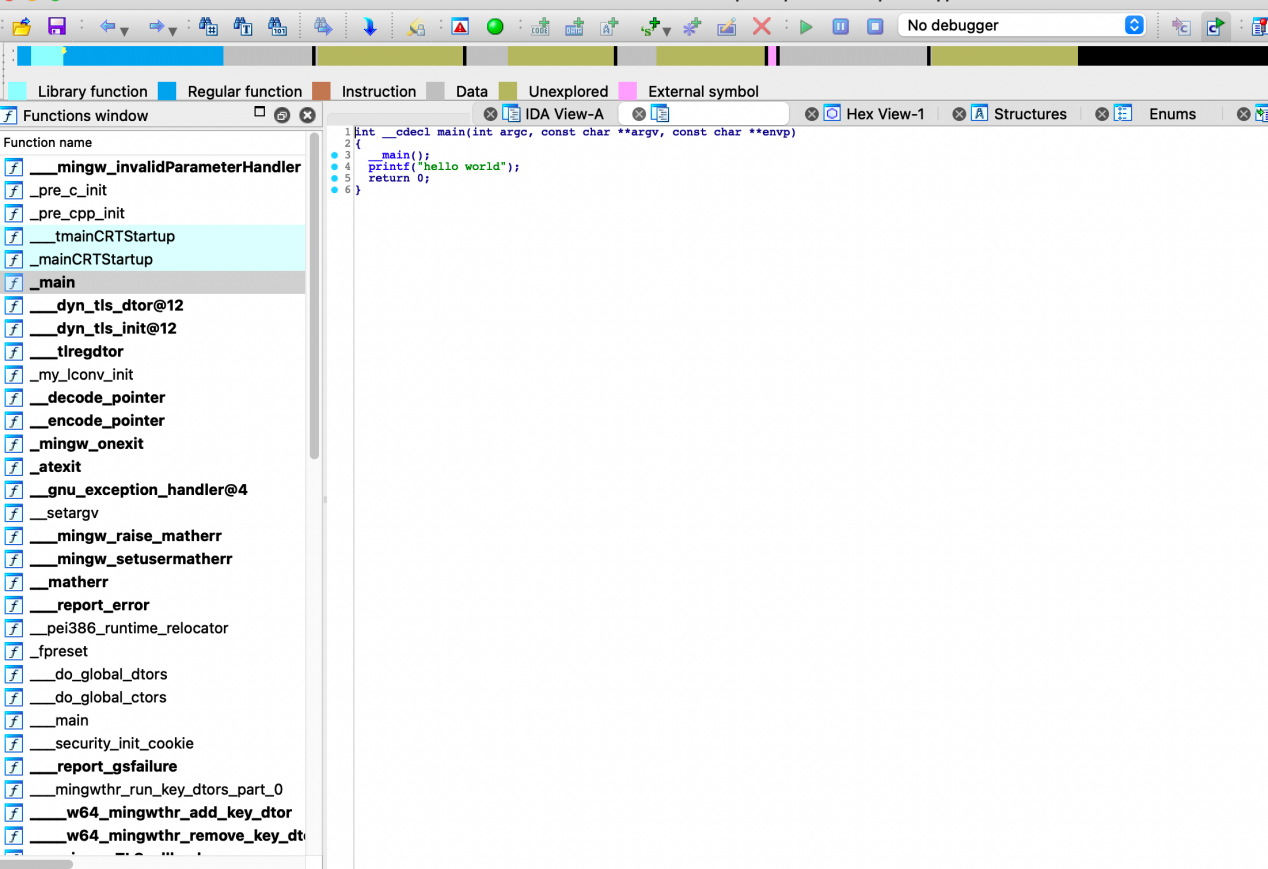


图 1 IDA打开界面

1. 加壳

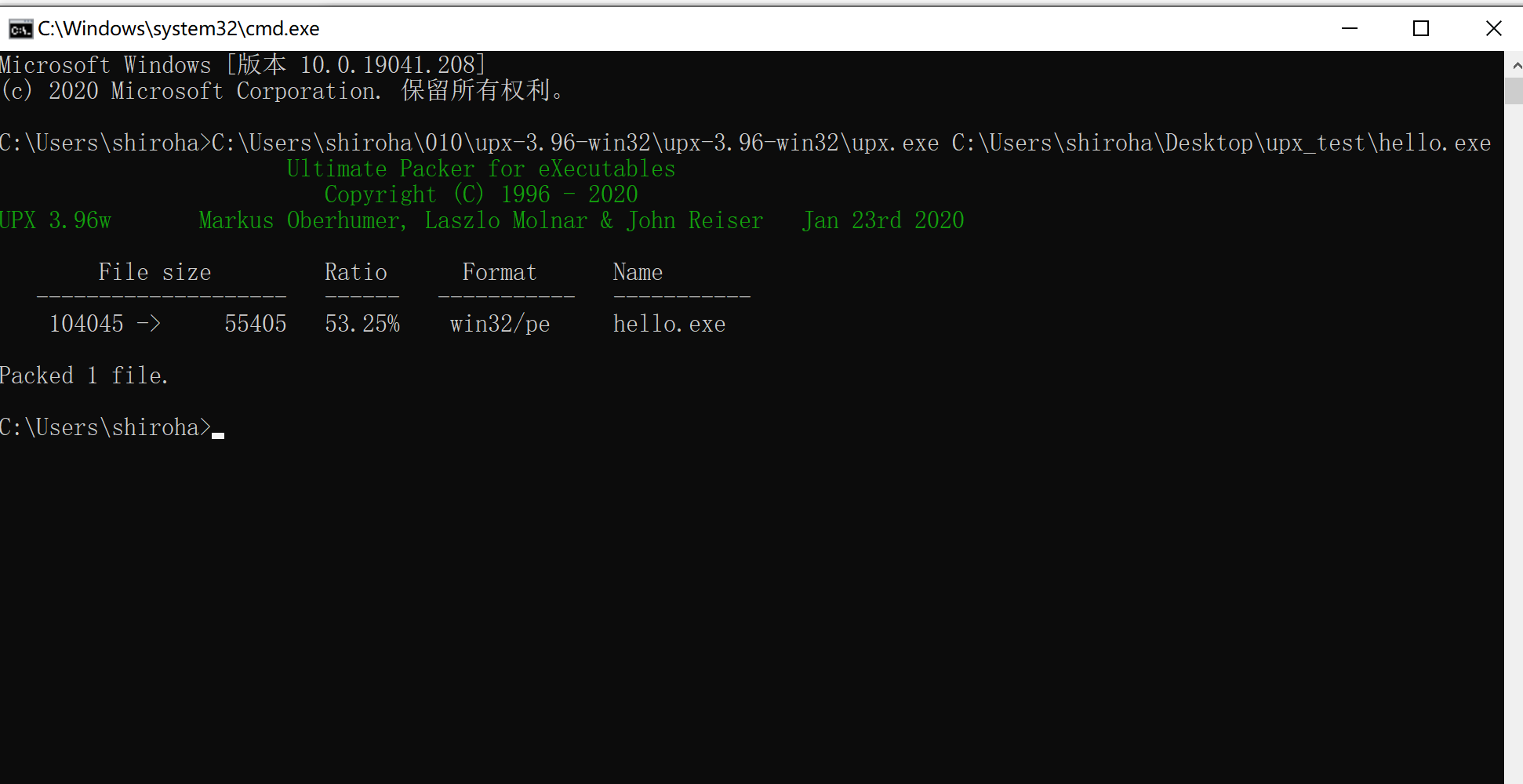


图2加壳

1. ESP平衡定律

运行到pushad指令（x32dbg会下一次性断点）

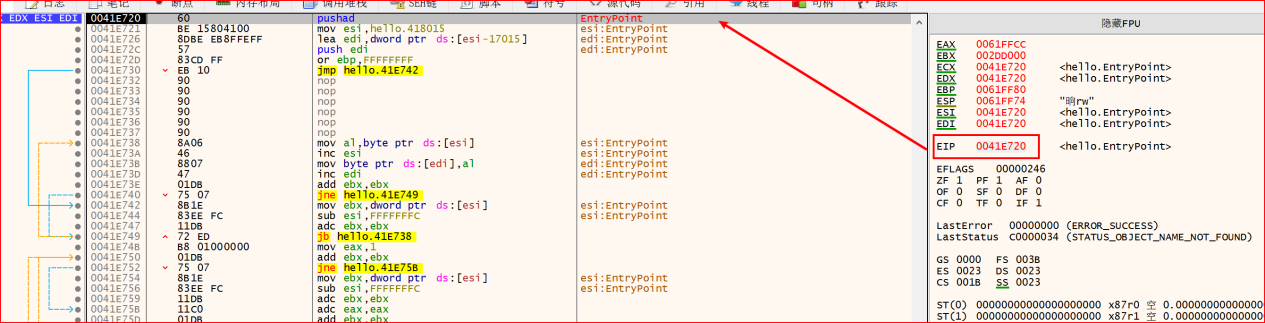
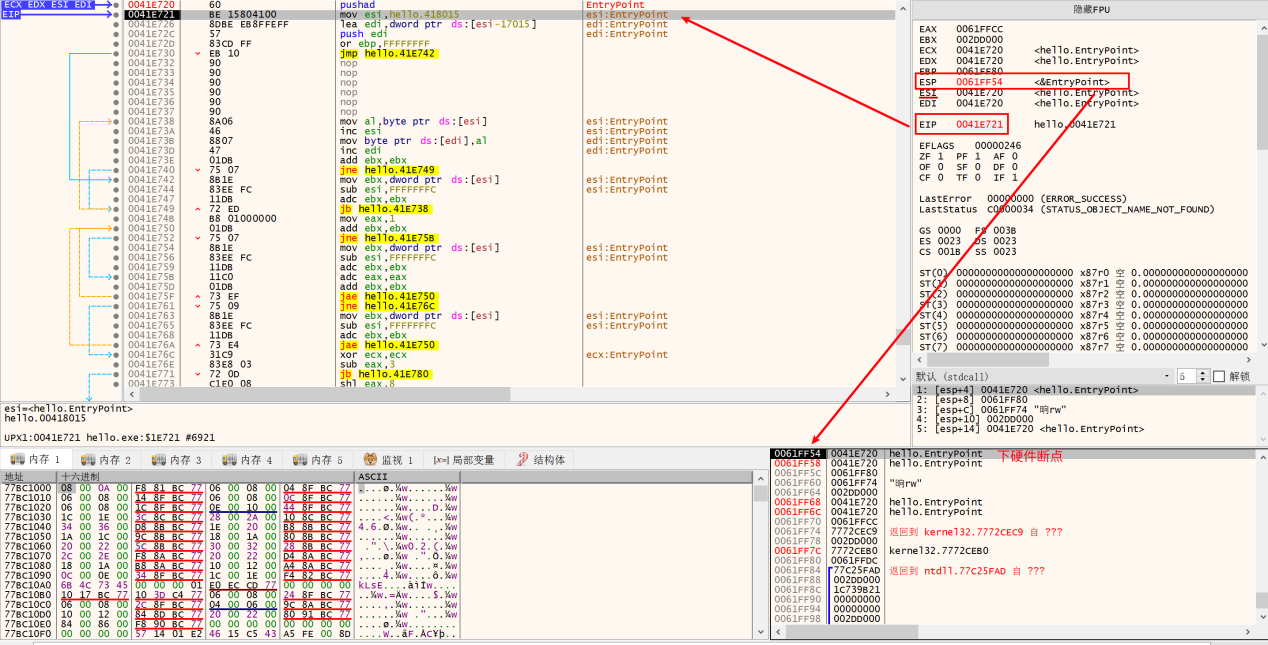


图3 pushad

执行pushad指令后，在esp指向的内存出下硬件断点



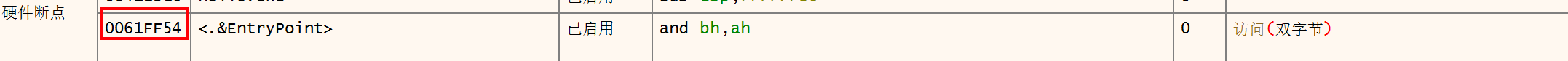


图4 硬件断点

点击继续运行，触发硬件断点，发现在popad之后，并且在下方不远处有一条跳转指令，跳转到离目前eip很远的地方

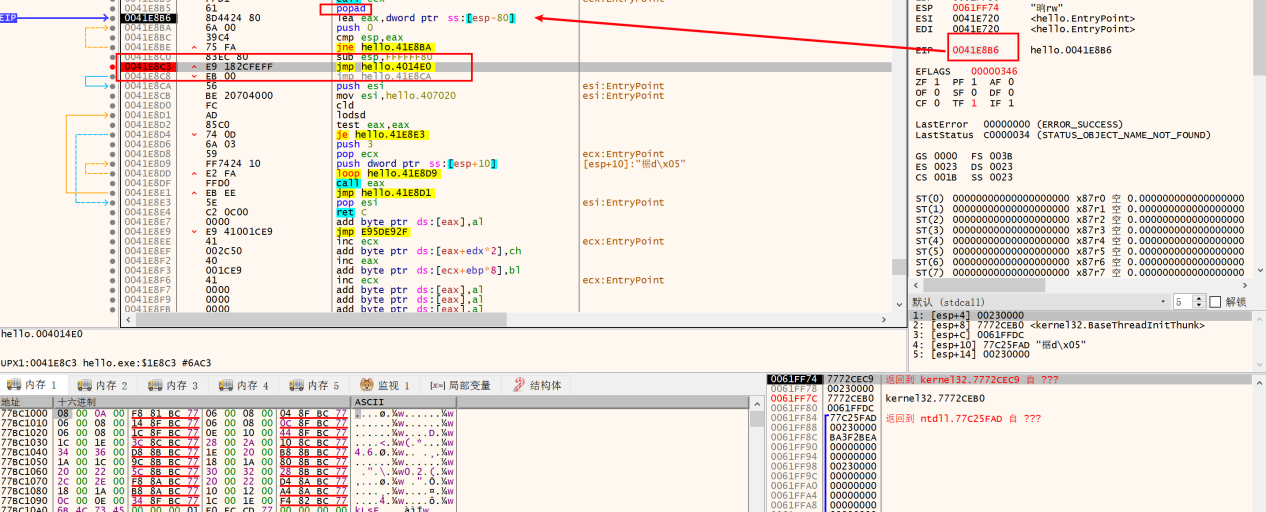


图5 触发硬件断点

跳转完成之后，可以直接看到程序，虽然还需要一点流程。



图6 源程序

然后dump此时的程序即可

1. 思考题
2. 为什么需要加壳

（1）防止软件被破解，保护软件版权；

（2）增加程序运行速度；

（3）病毒制作，绕过一些杀毒软件的扫描，从而实现作为病毒的一些入侵或破坏一些特征；

（4）减少程序空间占用。

2、 脱壳原理

脱壳的基本原则就是单步跟踪，只能往前，不能往后。脱壳的一般流程是：查 壳->寻找OEP->Dump->修复 找OEP的一般思路如下： 先看壳是加密壳还是压缩壳，压缩壳相对来说容易些，一般是没有异常，找到对应的popad后就能到入口，跳到入口的方式一般为。 我们知道文件被一些压缩加壳软件加密，下一步我们就要分析加密软件的名称、版本。因为不同软件甚至不同版本加的壳，脱壳处理的方法都不相同。

3、了解常用软件所加壳的种类

加密壳、压缩壳、伪装壳、多层壳等