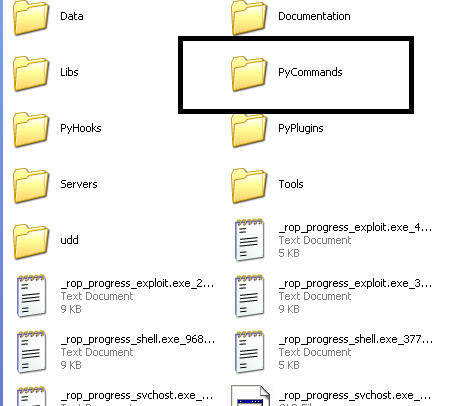
DEP绕过分析：

关于DEP的概念以及方法参见《0day安全：软件安全漏洞分析技术》，由于此书只给出了绕过的方法，调用链即ROP链的构造并未涉及，这里主要涉及ROP链的构造。

ROP链中的指令如果进行手动搜索的话有点太过于麻烦，为了自动实现ROP链的构造，为了实现rop链的构造，这里使用一个新的调试器，Immunity Debugger，此调试器由python编写所以需要安装python环境。之后安装Immunity Debugger调试器。然后在github上搜索mona，下载后将mona.py复制到Immunity Debugger安装目录下的PyCommand问价夹中。（此处使用的操作系统为windows xp sp2英文版，编译器VC6.0）



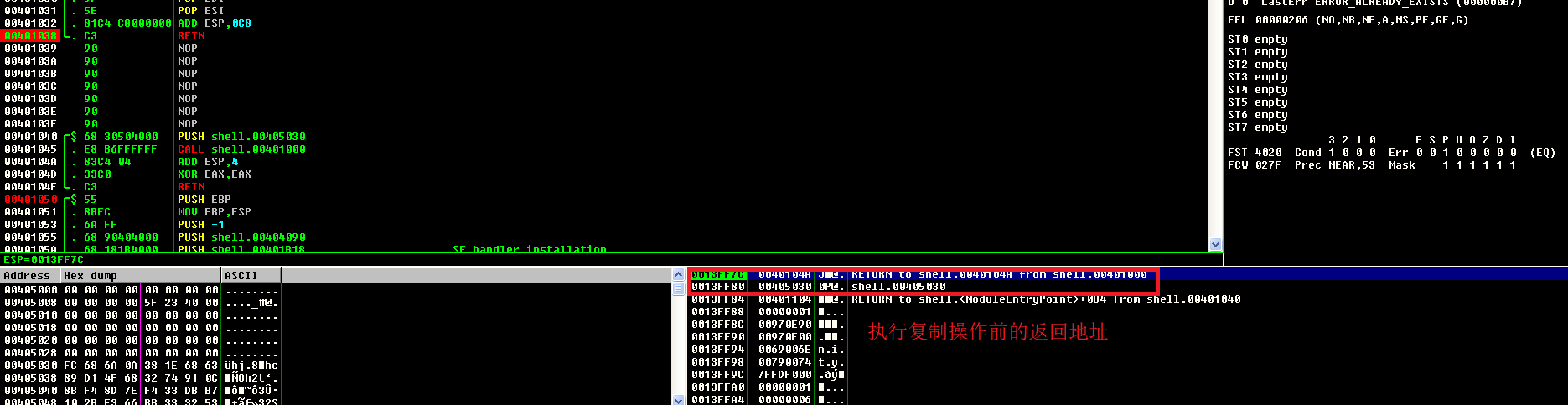
**图1 复制的目录**

使用closeDEP下中的exploit.c用VC6.0进行编译，编译为release版本，关闭优化。此处使用的代码由strcpy构造了一个栈溢出漏洞。编译运行后会弹出kingtest的对话框。此时的运行的程序是关闭DEP运行的。

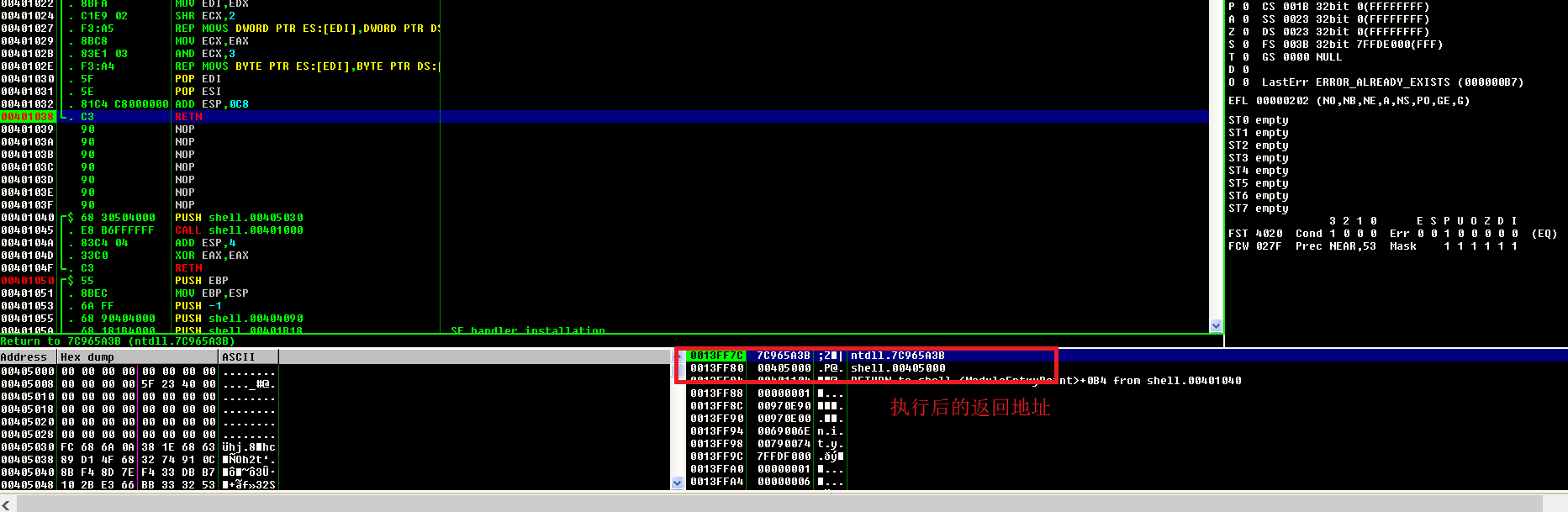


**图2 运行图**

使用Immunity Debugger进行动态分析，定位到strcpy函数中

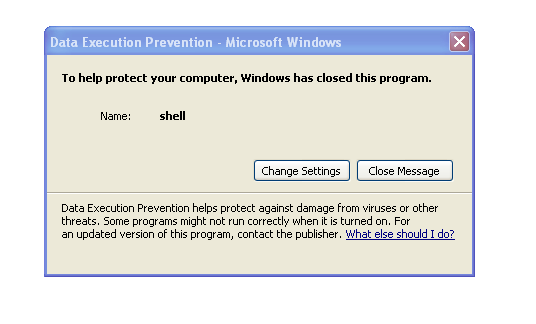
****

**图3执行复制前的返回地址**

****

**图3 执行后的返回地址**

此时在执行ret后就会跳转到恶意代码中执行。假如开启DEP后再运行将出现以下情况。

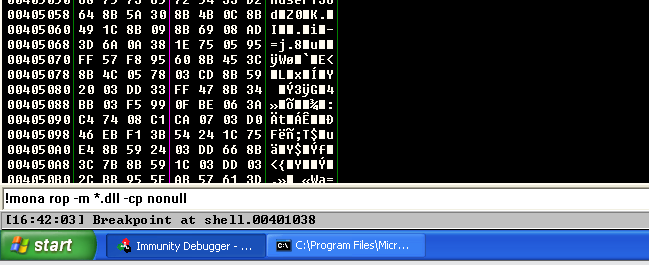
****

**图4 开启DEP后的运行效果图**

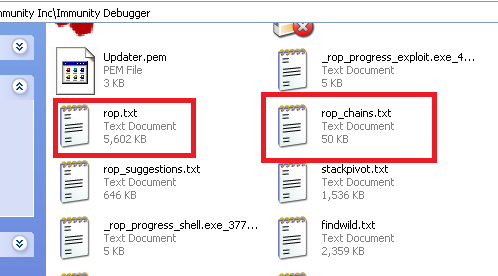
这时就不能跳转到栈区中执行了，为了能顺利跳转到栈区执行需要构造ROP链，利用执行区的代码片段完成绕过DEP的操作。

这时就要使用刚刚下载的mona.py进行ROP链的构造，（也可以自己手动搜索构造）。

输入!mona rop -m \*.dll -cp nonull后在安装目录就会出现rop\_chains.txt。此文件中是由此命令自动生成的ROP链，也可以使用同目录下的rop.txt自己选取指令构造（注意当所执行程序调用过多Dll文件时如果计算机性能不足，运行时间将过长或卡死，所以可以使用！mona modules 确定使用的dll文件然后选取dll文件替代\*.dll进行rop链的构造）

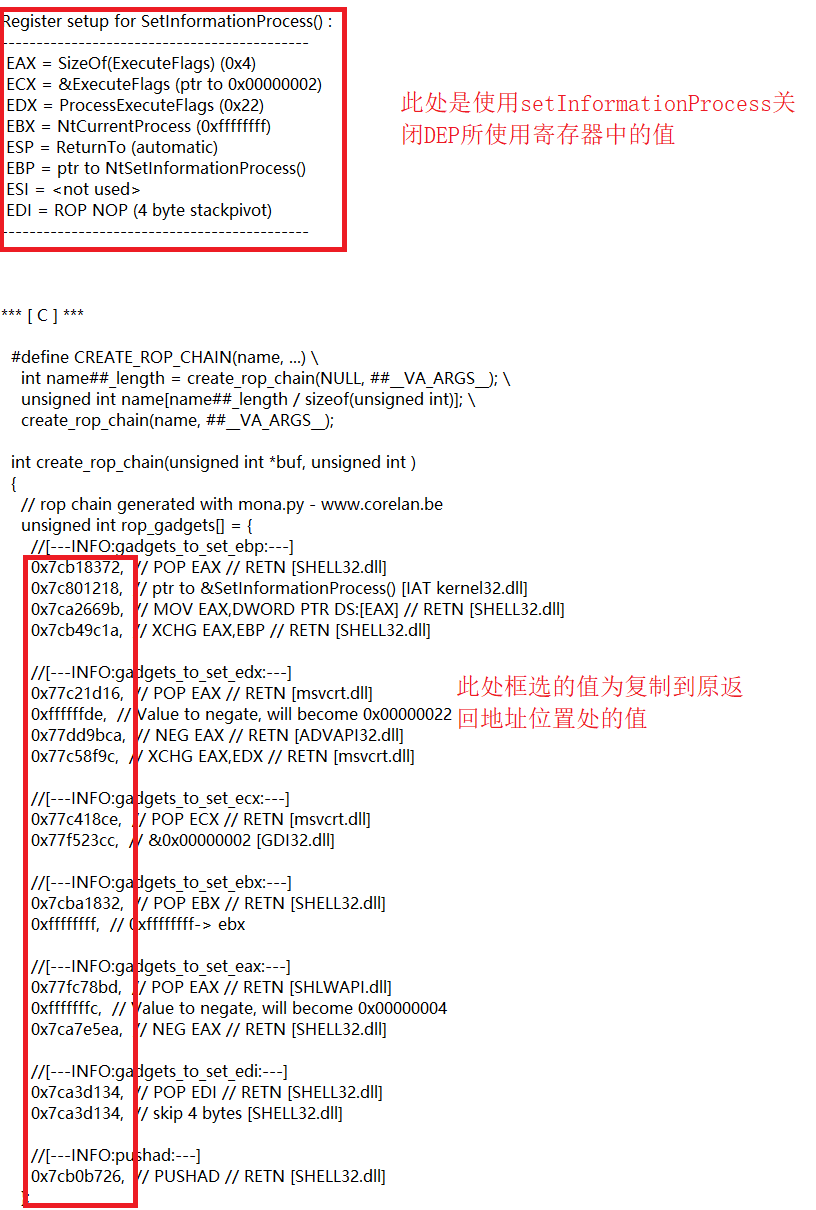
****

**图5 输入命令的位置**

****

**图6 生成的文件**

进入rop\_chain.txt后选取一个绕过的方法将生成的rop地址覆盖返回地址。图7中框选的值覆盖，如果还是不知道可以对比openDEP和closeDEP中的exploit.c的中值，覆盖后当返回时就会执行ROP链中的指令，进而关闭DEP。

****