

从零开始学习软件漏洞挖掘系列教程第一篇：工欲善其事必先利其器

1 实验简介

- 实验所属系列： 系统安全
- 实验对象： 本科/专科信息安全专业
- 相关课程及专业： 计算机网络
- 实验时数（学分）： 2 学时
- 实验类别： 实践实验类

2 实验目的

通过动手做一些实践，熟悉常用的软件漏洞挖掘工具，能在日后的软件漏洞挖掘做到游刃有余。

3 预备知识

1. 关于 Immunity Debugger 的一些基础知识

Immunity Debugger 是位于迈阿密的专业渗透测试技术公司发布的一种工具，这个工具能够加快编写利用安全漏洞代码，分析恶意软件和二进制文件逆向工程等的速度。Immunity 称这个调试工具能帮助渗透测试人员制作利用安全漏洞代码的时间减少一半。尤其是 Immunity Debugger 的插件 mona.py 更是软件漏洞挖掘的神器。

2. 关于 Windbg 的一些基础知识

Windbg 是在 windows 平台下，强大的用户态和内核态调试工具。虽然 windbg 也提供图形界面操作，但它最强大的地方还是有着强大的调试命令，一般会结合 GUI 和命令行进行操作，常用的视图有：局部变量、全局变量、调用栈、线程、命令、寄存器、白板等。其中“命令”视图是默认打开的。

3. 关于 Python 的一些基础知识

Python，是一种面向对象的解释性的计算机程序设计语言，也是一种功能强大而完善的通用型语言，已经具有十多年的发展历史，成熟且稳定。Python 具有脚本语言中最丰富和强大的类库，足以支持绝大多数日常应用。Python 在漏洞利用是理想的开始 Exploit 工具。

4 实验环境



服务器: Windows 7 SP1 , IP 地址: 随机分配

辅助工具: Windbg, ImmunityDebugger, python2.7, mona.py

mona.py 是由 corelan team 整合的一个可以自动构造 Rop Chain 而且集成了

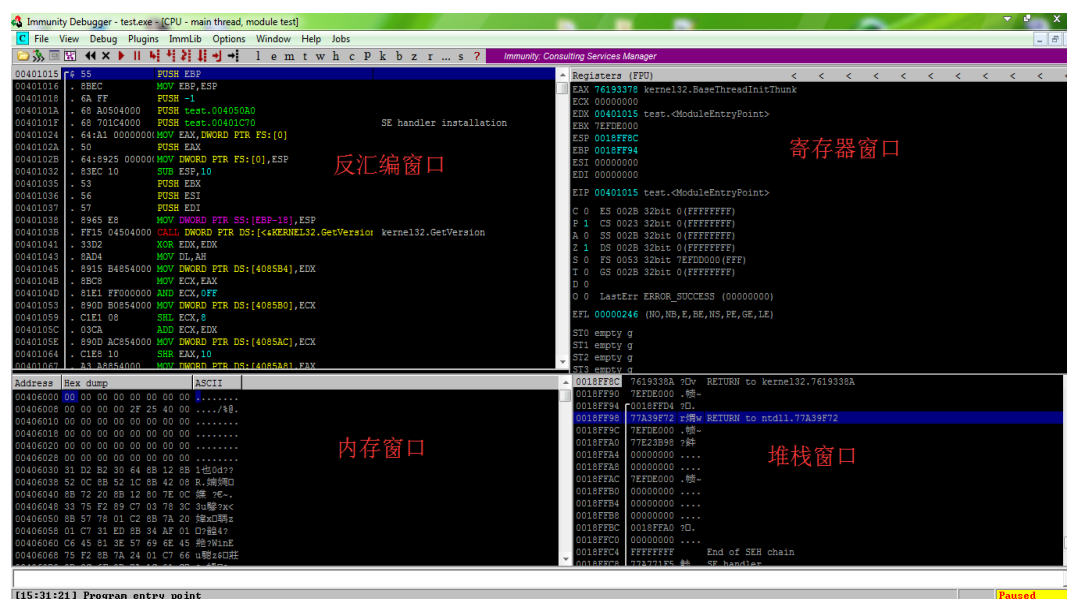
metasploit 计算偏移量功能的强大挖洞辅助插件'

5 实验步骤

5.1 实验任务一

任务描述: 熟悉 Immunity Debugger 的基本使用。

1. 我们用 Immunity Debugger 打开一个软件将会看到下面



Immunity Debugger 主界面有四个窗口，分别是

*反汇编窗口，反汇编窗口又分为四列：地址，机器码，机器码对应的汇编指令，注释。

*寄存器窗口，这里显示了某时刻 EAX(累加器),EBX(基址寄存器),ECX(计数器),EDX(数据寄存器),ESI(源变址寄存器),EDI(目的变址寄存器),EBP(基址指针),ESP(堆栈指针),EIP(指令指针)等寄存器的值。

*内存窗口，这个可以查看某个地址的内容比如我想看看 0x401000 这个地址有什么东西，那么只需要在内存窗口 Ctrl+g 然后输入 401000 回车

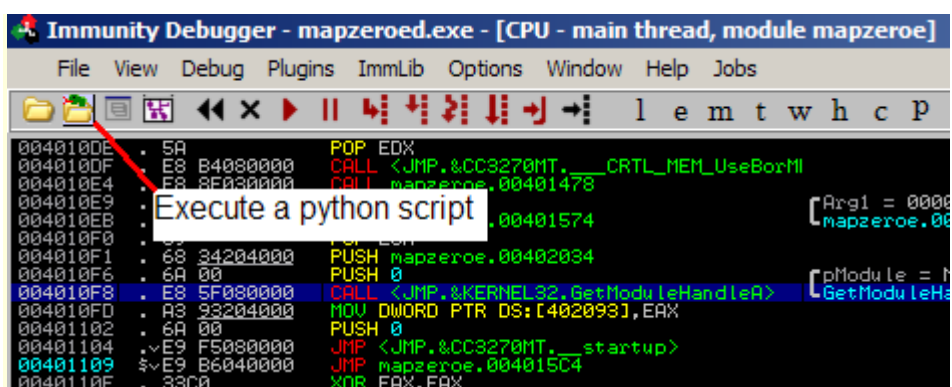
Address	Hex dump	ASCII
00401000	55 8B EC 53 56 57 8D 05	U 类SVW?
00401008	30 60 40 00 FF E0 33 C0	0`0. ??
00401010	5F 5E 5B 5D C3 55 8B EC	_^[] 脱类
00401018	6A FF 68 A0 50 40 00 68	j h 嫩0.1
00401020	70 1C 40 00 64 A1 00 00	p0.d?.
00401028	00 00 50 64 89 25 00 00	..Pd?..
00401030	00 00 83 EC 10 53 56 57	..浚0SVW
00401038	89 65 E8 FF 15 04 50 40	堤?00P0
00401040	00 33 D2 8A D4 89 15 B4	.3 见 鼓0?
00401048	85 40 00 8B C8 81 E1 FF	舞. 婢全
00401050	00 00 00 89 0D B0 85 40	...? 婢0
00401058	00 C1 E1 08 03 CA 89 0D	.玲00 黄.
00401060	AC 85 40 00 C1 E8 10 A3	琼0. 浚0?
00401068	A8 85 40 00 6A 00 E8 A8	0.j. 媛
00401070	00 00 00 00 00 00 00 00
00401078	00 00 00 00 00 00 00 00
00401080	00 00 00 00 00 00 00 00
00401088	00 00 00 00 00 00 00 00
00401090	00 00 00 00 00 00 00 00
00401098	00 00 00 00 00 00 00 00
004010A0	00 00 00 00 00 00 00 00
004010A8	00 00 00 00 00 00 00 00
004010B0	00 00 00 00 00 00 00 00
004010B8	00 00 00 00 00 00 00 00
004010C0	00 00 00 00 00 00 00 00
004010C8	00 00 00 00 00 00 00 00
004010D0	00 00 00 00 00 00 00 00
004010D8	00 00 00 00 00 00 00 00
004010E0	00 00 00 00 00 00 00 00
004010E8	00 00 00 00 00 00 00 00
004010F0	00 00 00 00 00 00 00 00
004010F8	00 00 00 00 00 00 00 00
00401100	00 00 00 00 00 00 00 00
00401108	00 00 00 00 00 00 00 00
00401110	00 00 00 00 00 00 00 00
00401118	00 00 00 00 00 00 00 00
00401120	00 00 00 00 00 00 00 00
00401128	00 00 00 00 00 00 00 00
00401130	00 00 00 00 00 00 00 00
00401138	00 00 00 00 00 00 00 00
00401140	00 00 00 00 00 00 00 00
00401148	00 00 00 00 00 00 00 00
00401150	00 00 00 00 00 00 00 00
00401158	00 00 00 00 00 00 00 00
00401160	00 00 00 00 00 00 00 00
00401168	00 00 00 00 00 00 00 00
00401170	00 00 00 00 00 00 00 00
00401178	00 00 00 00 00 00 00 00
00401180	00 00 00 00 00 00 00 00
00401188	00 00 00 00 00 00 00 00
00401190	00 00 00 00 00 00 00 00
00401198	00 00 00 00 00 00 00 00
004011A0	00 00 00 00 00 00 00 00
004011A8	00 00 00 00 00 00 00 00
004011B0	00 00 00 00 00 00 00 00
004011B8	00 00 00 00 00 00 00 00
004011C0	00 00 00 00 00 00 00 00
004011C8	00 00 00 00 00 00 00 00
004011D0	00 00 00 00 00 00 00 00
004011D8	00 00 00 00 00 00 00 00
004011E0	00 00 00 00 00 00 00 00
004011E8	00 00 00 00 00 00 00 00
004011F0	00 00 00 00 00 00 00 00
004011F8	00 00 00 00 00 00 00 00
00401200	00 00 00 00 00 00 00 00
00401208	00 00 00 00 00 00 00 00
00401210	00 00 00 00 00 00 00 00
00401218	00 00 00 00 00 00 00 00
00401220	00 00 00 00 00 00 00 00
00401228	00 00 00 00 00 00 00 00
00401230	00 00 00 00 00 00 00 00
00401238	00 00 00 00 00 00 00 00
00401240	00 00 00 00 00 00 00 00
00401248	00 00 00 00 00 00 00 00
00401250	00 00 00 00 00 00 00 00
00401258	00 00 00 00 00 00 00 00
00401260	00 00 00 00 00 00 00 00
00401268	00 00 00 00 00 00 00 00
00401270	00 00 00 00 00 00 00 00
00401278	00 00 00 00 00 00 00 00
00401280	00 00 00 00 00 00 00 00
00401288	00 00 00 00 00 00 00 00
00401290	00 00 00 00 00 00 00 00
00401298	00 00 00 00 00 00 00 00
004012A0	00 00 00 00 00 00 00 00
004012A8	00 00 00 00 00 00 00 00
004012B0	00 00 00 00 00 00 00 00
004012B8	00 00 00 00 00 00 00 00
004012C0	00 00 00 00 00 00 00 00
004012C8	00 00 00 00 00 00 00 00
004012D0	00 00 00 00 00 00 00 00
004012D8	00 00 00 00 00 00 00 00
004012E0	00 00 00 00 00 00 00 00
004012E8	00 00 00 00 00 00 00 00
004012F0	00 00 00 00 00 00 00 00
004012F8	00 00 00 00 00 00 00 00
00401300	00 00 00 00 00 00 00 00
00401308	00 00 00 00 00 00 00 00
00401310	00 00 00 00 00 00 00 00
00401318	00 00 00 00 00 00 00 00
00401320	00 00 00 00 00 00 00 00
00401328	00 00 00 00 00 00 00 00
00401330	00 00 00 00 00 00 00 00
00401338	00 00 00 00 00 00 00 00
00401340	00 00 00 00 00 00 00 00
00401348	00 00 00 00 00 00 00 00
00401350	00 00 00 00 00 00 00 00
00401358	00 00 00 00 00 00 00 00
00401360	00 00 00 00 00 00 00 00
00401368	00 00 00 00 00 00 00 00
00401370	00 00 00 00 00 00 00 00
00401378	00 00 00 00 00 00 00 00
00401380	00 00 00 00 00 00 00 00
00401388	00 00 00 00 00 00 00 00
00401390	00 00 00 00 00 00 00 00
00401398	00 00 00 00 00 00 00 00
004013A0	00 00 00 00 00 00 00 00
004013A8	00 00 00 00 00 00 00 00
004013B0	00 00 00 00 00 00 00 00
004013B8	00 00 00 00 00 00 00 00
004013C0	00 00 00 00 00 00 00 00
004013C8	00 00 00 00 00 00 00 00
004013D0	00 00 00 00 00 00 00 00
004013D8	00 00 00 00 00 00 00 00
004013E0	00 00 00 00 00 00 00 00
004013E8	00 00 00 00 00 00 00 00
004013F0	00 00 00 00 00 00 00 00
004013F8	00 00 00 00 00 00 00 00
00401400	00 00 00 00 00 00 00 00
00401408	00 00 00 00 00 00 00 00
00401410	00 00 00 00 00 00 00 00
00401418	00 00 00 00 00 00 00 00
00401420	00 00 00 00 00 00 00 00
00401428	00 00 00 00 00 00 00 00
00401430	00 00 00 00 00 00 00 00
00401438	00 00 00 00 00 00 00 00
00401440	00 00 00 00 00 00 00 00
00401448	00 00 00 00 00 00 00 00
00401450	00 00 00 00 00 00 00 00
00401458	00 00 00 00 00 00 00 00
00401460	00 00 00 00 00 00 00 00
00401468	00 00 00 00 00 00 00 00
00401470	00 00 00 00 00 00 00 00
00401478	00 00 00 00 00 00 00 00
00401480	00 00 00 00 00 00 00 00
00401488	00 00 00 00 00 00 00 00
00401490	00 00 00 00 00 00 00 00
00401498	00 00 00 00 00 00 00 00
004014A0	00 00 00 00 00 00 00 00
004014A8	00 00 00 00 00 00 00 00
004014B0	00 00 00 00 00 00 00 00
004014B8	00 00 00 00 00 00 00 00
004014C0	00 00 00 00 00 00 00 00
004014C8	00 00 00 00 00 00 00 00
004014D0	00 00 00 00 00 00 00 00
004014D8	00 00 00 00 00 00 00 00
004014E0	00 00 00 00 00 00 00 00
004014E8	00 00 00 00 00 00 00 00
004014F0	00 00 00 00 00 00 00 00
004014F8	00 00 00 00 00 00 00 00
00401500	00 00 00 00 00 00 00 00
00401508	00 00 00 00 00 00 00 00
00401510	00 00 00 00 00 00 00 00
00401518	00 00 00 00 00 00 00 00
00401520	00 00 00 00 00 00 00 00
00401528	00 00 00 00 00 00 00 00
00401530	00 00 00 00 00 00 00 00
00401538	00 00 00 00 00 00 00 00
00401540	00 00 00 00 00 00 00 00
00401548	00 00 00 00 00 00 00 00
00401550	00 00 00 00 00 00 00 00
00401558	00 00 00 00 00 00 00 00
00401560	00 00 00 00 00 00 00 00
00401568	00 00 00 00 00 00 00 00
00401570	00 00 00 00 00 00 00 00
00401578	00 00 00 00 00 00 00 00
00401580	00 00 00 00 00 00 00 00
00401588	00 00 00 00 00 00 00 00
00401590	00 00 00 00 00 00 00 00
00401598	00 00 00 00 00 00 00 00
004015A0	00 00 00 00 00 00 00 00
004015A8	00 00 00 00 00 00 00 00
004015B0	00 00 00 00 00 00 00 00
004015B8	00 00 00 00 00 00 00 00
004015C0	00 00 00 00 00 00 00 00
004015C8	00 00 00 00 00 00 00 00
004015D0	00 00 00 00 00 00 00 00
004015D8	00 00 00 00 00 00 00 00
004015E0	00 00 00 00 00 00 00 00
004015E8	00 00 00 00 00 00 00 00
004015F0	00 00 00 00 00 00 00 00
004015F8	00 00 00 00 00 00 00 00
00401600	00 00 00 00 00 00 00 00
00401608	00 00 00 00 00 00 00 00
00401610	00 00 00 00 00 00 00 00
00401618	00 00 00 00 00 00 00 00
00401620	00 00 00 00 00 00 00 00
00401628	00 00 00 00 00 00 00 00
00401630	00 00 00 00 00 00 00 00
00401638	00 00 00 00 00 00 00 00
00401640	00 00 00 00 00 00 00 00
00401648	00 00 00 00 00 00 00 00
00401650	00 00 00 00 00 00 00 00
00401658	00 00 00 00 00 00 00 00
00401660	00 00 00 00 00 00 00 00
00401668	00 00 00 00 00 00 00 00
00401670	00 00 00 00 00 00 00 00
00401678	00 00 00 00 00 00 00 00
00401680	00 00 00 00 00 00 00 00
00401688	00 00 00 00 00 00 00 00
00401690	00 00 00 00 00 00 00 00
00401698	00 00 00 00 00 00 00 00
004016A0	00 00 00 00 00 00 00 00
004016A8	00 00 00 00 00 00 00 00
004016B0	00 00 00 00 00 00 00 00
004016B8	00 00 00 00 00 00 00 00
004016C0	00 00 00 00 00 00 00 00
004016C8	00 00 00 00 00 00 00 00
004016D0	00 00 00 00 00 00 00 00
004016D8	00 00 00 00 00 00 00 00
004016E0	00 00 00 00 00 00 00 00
004016E8	00 00 00 00 00 00 00 00
004016F0	00 00 00 00 00 00 00 00
004016F8	00 00 00 00 00 00 00 00
00401700	00 00 00 00 00 00 00 00
00401708	00 00 00 00 00 00 00 00
00401710	00 00 00 00 00 00 00 00
00401718	00 00 00 00 00 00 00 00

```

7709020C . 78410100 DD 00014178
EBX=7EFDE000
Stack SS: (0018FFFF)-00000000
Address Hex dump ASCII
00040123 00 58 09 00 00 6C 09 00 .X...1..
0004012B 00 3A 00 00 00 A8 09 00 .....?.
00040133 00 BC 09 00 00 2C 00 00 .?...?.
0004013B 00 E8 09 00 00 FC 09 00 .?...?.
00040143 00 2E 00 00 00 2C 0A 00 .....?.
0004014B 00 50 0A 00 00 32 00 00 .P...2..
00040153 00 84 0A 00 00 98 0A 00 .?...?.
0004015B 00 36 00 00 00 D0 0A 00 .6...?.
00040163 00 E4 0A 00 00 40 00 00 .?...@..
0004016B 00 24 0B 00 00 50 0B 00 .$.D..P.
dd 0x40123

```

关于 Python 脚本

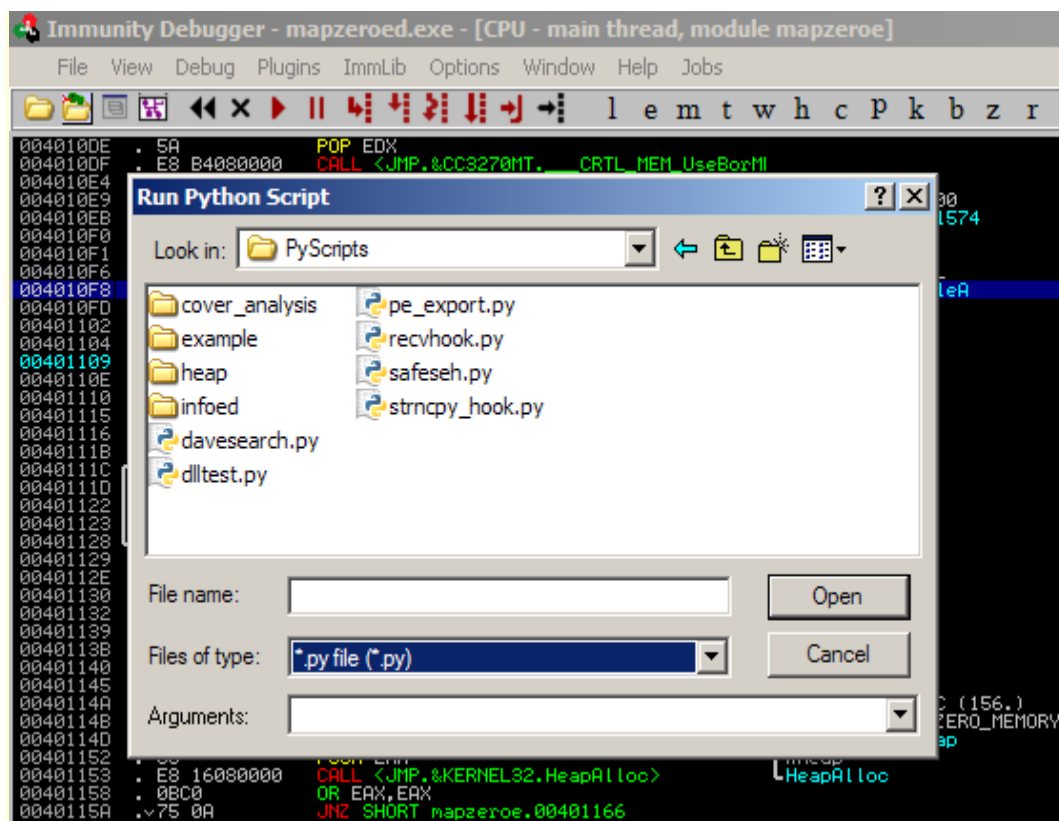


```

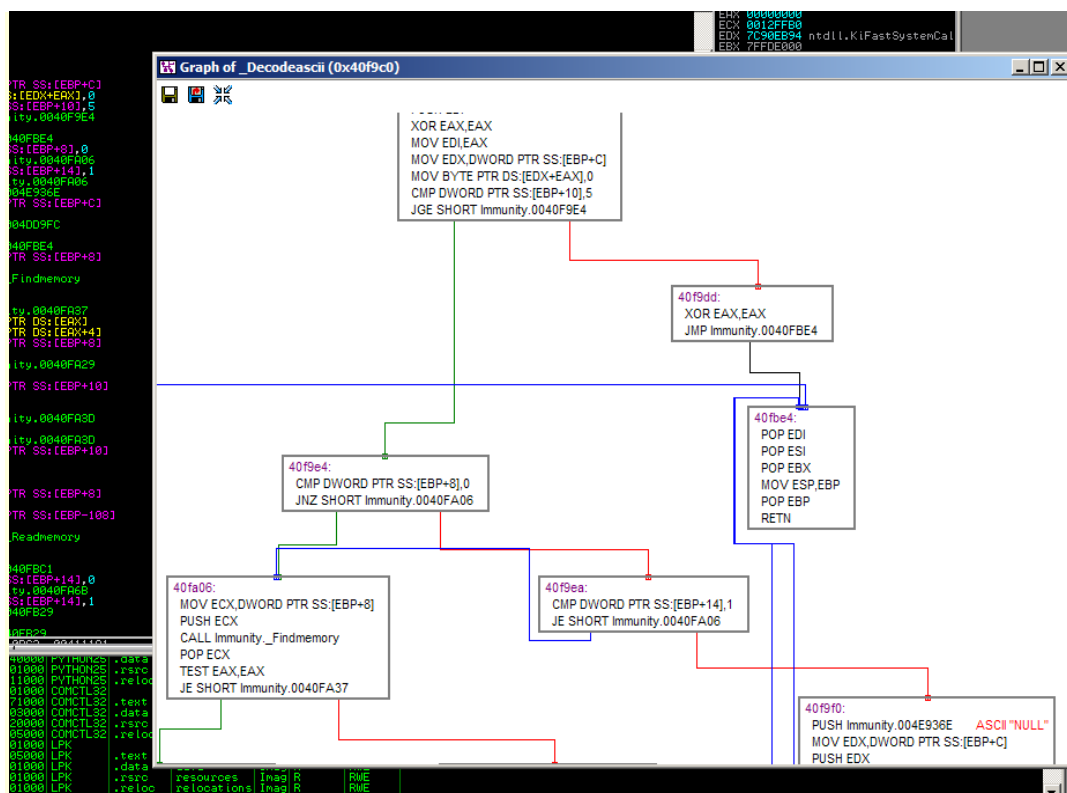
Immunity Debugger - mapzeroe.exe - [CPU - main thread, module mapzeroe]
File View Debug Plugins ImmLib Options Window Help Jobs
l e m t w h c p
004010DE . 5A POP EDX
004010DF . E8 B4080000 CALL <JMP.&CC3270MT.__CRTL_MEM_UseBorM
004010E4 . E8 8F030000 CALL mapzeroe.00401478 [Arg1 = 0008
mapzeroe.00
004010E8 . Execute a python script 00401574
004010F0 . 5A POP EDI
004010F1 . 68 34204000 PUSH mapzeroe.00402034
004010F6 . 6A 00 PUSH 0 [pModule = h
004010F8 . E8 5F080000 CALL <JMP.&KERNEL32.GetModuleHandleA> [GetModuleHa
004010FD . A3 93204000 MOV DWORD PTR DS:[402093],EAX
00401102 . 6A 00 PUSH 0
00401104 . E9 F5080000 JMP <JMP.&CC3270MT.__startup>
00401109 . E9 B6040000 JMP mapzeroe.004015C4
0040110E . 33C0 XOR EAX,EAX

```

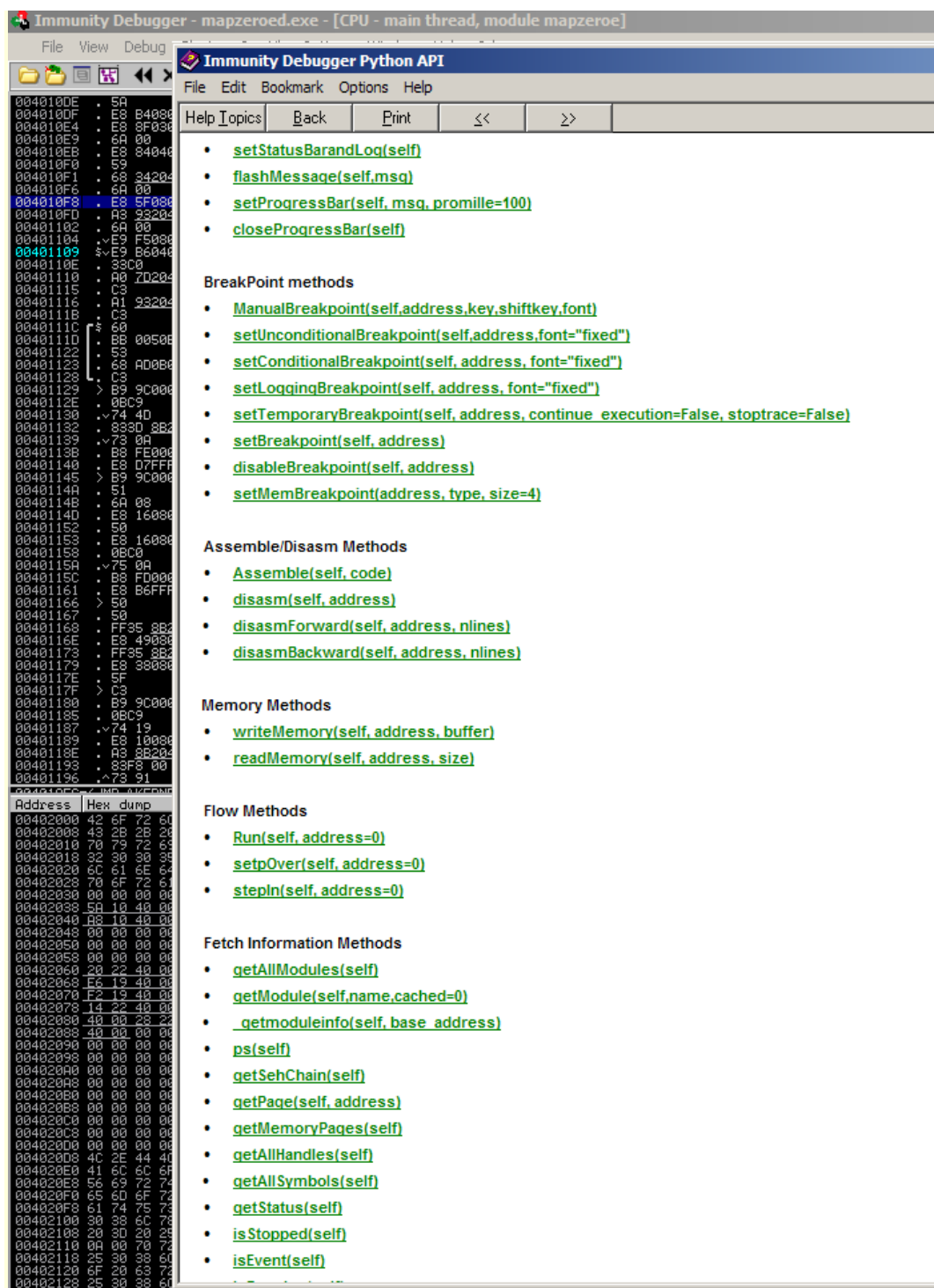
Python 脚本可以在运行时加载和修改。包括 Python 解释器将加载任何更改您的自定义脚本。包括示例脚本,如何创建自己的完整文档



Python GraphingBuilt 绘图另一个 Immunity 调试器的功能是创建函数图形的能力。我们 Python 向量库将创建一个窗口内 Immunity 调试器按一个按钮图您所选择的功能。不需要第三方软件。



Immunity 调试器的 Python API 包含许多有用的实用程序和功能。脚本可以像本机代码集成到调试器，这意味着您的代码可以创建自定义表、图表以及各种接口，仍在 Immunity 调试器内。例如，当 Immunity SafeSEH 脚本运行时，它的输出结果到表内 Immunity 调试器窗口。



2. 接下来重点介绍 Immunity Debugger 的一个插件 mona.py。在 Immunity Debugger 下方的命令行输入!mona 即可查看插件所有信息

The image is a screenshot of the Immunity Debugger application. The title bar reads "Immunity Debugger - test.exe - [Log data]". The menu bar includes "File", "View", "Debug", "Plugins", "ImmLib", "Options", "Window", "Help", and "Jobs". The toolbar contains various icons for file operations, debugging, and navigation. The main window displays the "mona" plugin interface, which includes a "new" button and a "mona" button. The "mona" button is highlighted. The main window also displays the "mona" plugin version 2.0 r427, written by Corelan, and a link to the project page: https://redmine.corelan.be/projects/mona. The command window shows the output of the "mona" plugin, including a list of modules and a search for pointers. The status bar at the bottom indicates the current module is "Immona".

我们这个系列教程用到的命令有!mona pc N (产生 N 个随机字符串), !mona poststr (计算 str 在 N 个字符中出现的位置), !mona seh (找出没有开启 safeseh 模块中的 pop pop retn 序列)。如果你不懂某个命令怎么用请输入 !mona help 某个命令。如图

```
OBADF00D Usage of command 'pc' :
OBADF00D -----
OBADF00D Create a cyclic pattern of a given size. Output will be written to pattern.txt
OBADF00D Mandatory argument : size (numeric value)
OBADF00D Optional arguments :
OBADF00D     -js : output pattern in unicode escaped javascript format
OBADF00D     -extended : extend the 3rd character set (numbers) with punctuation marks etc
OBADF00D     -c1 <chars> : set the first charset to this string of characters
OBADF00D     -c2 <chars> : set the second charset to this string of characters
OBADF00D     -c3 <chars> : set the third charset to this string of characters
OBADF00D
OBADF00D [+] This mona.py action took 0:00:00.002000
```

5.1.2. 练习



关于 mona 插件，以下说法错误的是？【单选题】

- 【A】!mona pattern_create 3000 可以创建 3000 个随机字符。
- 【B】某个命令的用法可以 !mona help 命令。
- 【C】!mona bytearray -b '\x00' 产生一系列除\x00 外的随机字节数组
- 【D】!mona 是一个自动化挖掘漏洞工具

答案: D

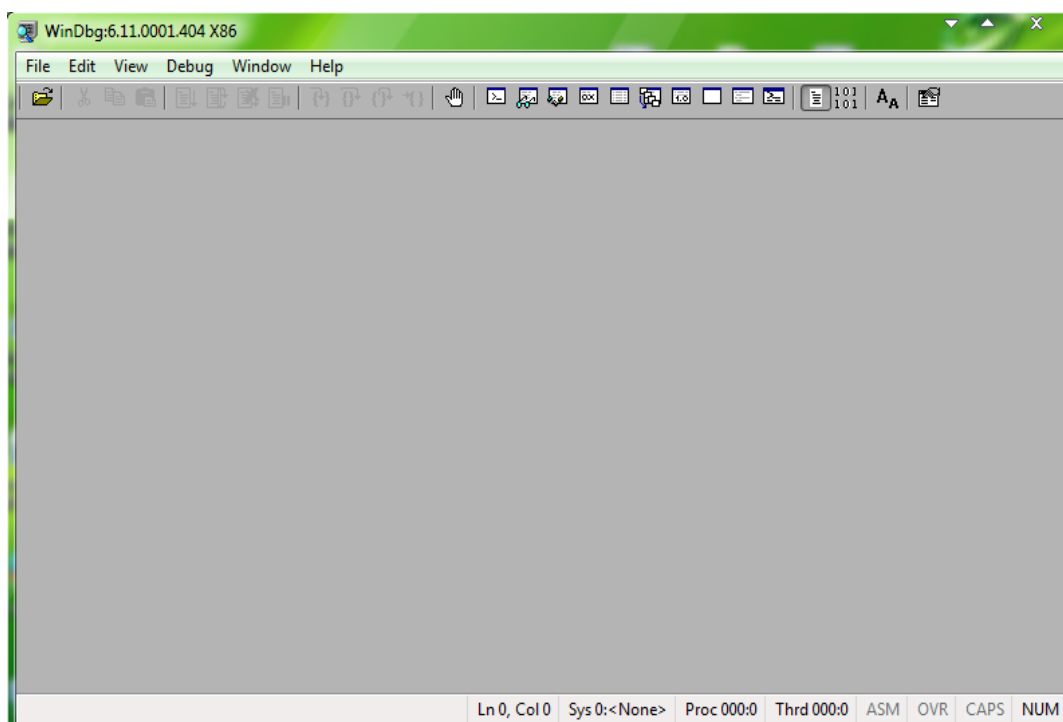
5.2 实验任务二

任务描述：熟悉 WinDbg 和 python 的基本使用

1. 我们打开 windbg 后，默认是这个界面，清新，简洁。

WinDbg 支持以下三种类型的命令：

- 常规命令，用来调试进程
- 点命令，用来控制调试器
- 扩展命令，可以添加叫 WinDbg 的自定义命令，一般由扩展 dll 提供这些命令



下面列举一些常用的 windbg 命令：

1. 启动 WinDbg

要用 WinDbg (x86) 调试 32 位程序，用 WinDbg (x64) 调试 64 位程序。

2. 使用帮助

任何时候都可以使用 !help 命令来获取帮助，查看命令的使用方法。

3. 设置 SymbolFile Path，指定了符号库，我们才能看到详细的类型信息

SRV*c:\symbols*http://msdl.microsoft.com/download/symbols

WinDbg 会将微软的符号库下载指定的本地目录中

4. 重新加载符号

如果进入调试之后才指定的符号路径，需要使用命令来重新加载符号
.reload

5.Ctrl+Break 终止一个很长时间没有完成的命令， **Ctrl+Break** 也可以让正在运行的程序暂停

6.保存 dump 文件

```
.dump /ma c:\test.dmp 保存 full-dump  
.dump /m c:\test.dmp 保存 mini-dump
```

7. 分析 Dump

一般先 !analyze -v Windbg 会根据上面命令自动分析，然后 ~* kv 打印所有线程的堆栈

8. 察看模块信息

```
lm 显示所有模块信息  
lmf 显示所有模块及其路径  
lmD 显示所有模块详细信息
```

9. 单步调试

```
g 继续运行(go), 热键 F5  
t 单步越过(step over), 热键 F10  
p 单步进入(step into), 热键 F11
```

10. 设置断点(break point)

bp [address] [“command”] 设置软件断点。

比如 bp kernel32!CreateProcessW 表示在调用这个 CreateProcess 时设置断点。

如 bp kernel32!CreateFileW "du poi(esp+4); g" 表示在调用 CreateFile 时打印出文件路径(第一个参数)，然后继续执行

针对某线程设置断点，只要在命令前加~线程号：

比如 ~0 bp 0x441242, 表示 0 号线程执行到地址 0x441242 时中断

ba [access size] [command]设置硬断点。

其中，access 指定访问方式(e 执行指令, r 读取数据, w 写入数据)

size 表示监视数据的大小(1, 2, 4)

比如 ba r4 0x414422, 表示在地址 0x414422 写入 4 字节数据是触发断点

11.管理断点

bl 列出所有当前断点的状态
 bc 清除断点, bc * 清除所有断点, bc 0 清除 0 号断点
 bd 禁用某个断点(disable)
 be 打开某个断点(enable)

12.察看堆栈

kn [frame count]察看当前堆栈及其索引, frame count 指定要显示多少帧
 kb 显示堆栈帧地址, 返回地址, 参数, 函数名等
 kv 在 kb 的基础上增加了函数调用约定等信息, 所以推荐用 kv 命令察看堆栈.
 .frame [frame index] 将当前堆栈切换到某个堆栈帧, 比如.frame 1 切换到第 1 帧
 dv 命令察看当前堆栈帧的局部变量

13.察看和修改寄存器

r 显示所有寄存器的值
 r eax=0x100 将 eax 寄存器的改成 0x100

14.搜索内存(search memory)

s -[type] range pattern

其中 type, b 表示 byte, w 表示 word, d 表示 dword, a 表示 ASCII string, u 表示 unicode string

Range 表示地址范围, 可以用 2 种表示: 一是起始地址加终止地址, 二是起始地址加 L 长度(不是字节长度, 是单位长度)。如果搜索空间长度超过 256M, 用 L?length。

Pattern 指定要搜索的内容。

比如 s -u 522e0000 527d1000 "web"表示在 522e0000 和 527d1000 之间搜索 Unicode 字符串" web"

比如 s -w 522e0000 L0x100 0x1212 0x2212 0x1234 表示在起始地址 522e0000 之后的 0x100 个单位内搜索 0x1212 0x2212 0x1234 系列的起始地址

15.反汇编某一地址

u address, 比如 u 0x410040 表示反汇编地址 0x410040 的代码
 uf 反汇编某个函数, 比如 uf test!main
 ub 反汇编某地址之前的代码, 比如 ub 0x 0x410040 L20
 !lmi [module name] 显示某一模块的详细信息

以上只是一部分命令，不用死记硬背，需要用的时候现查就行了。

下面使用 Windbg 实际分析一个程序：

```
//by www.netfairy.net
```

```
#include<stdio.h>
```

```
#include<windows.h>
```

```
//主函数
```

```
int main()
```

```
{
```

```
    char buffer[8];
```

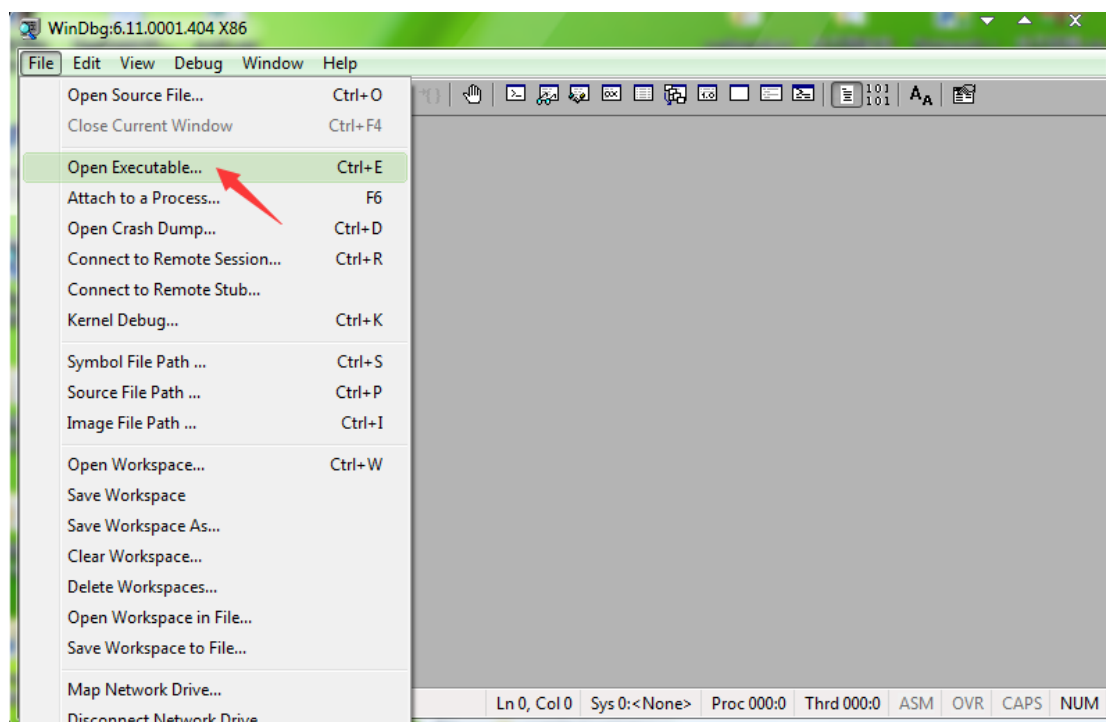
```
    MessageBox(NULL,"Hello This is a test","Netfairy",NULL);
```

```
    strcpy(buffer,"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA");
```

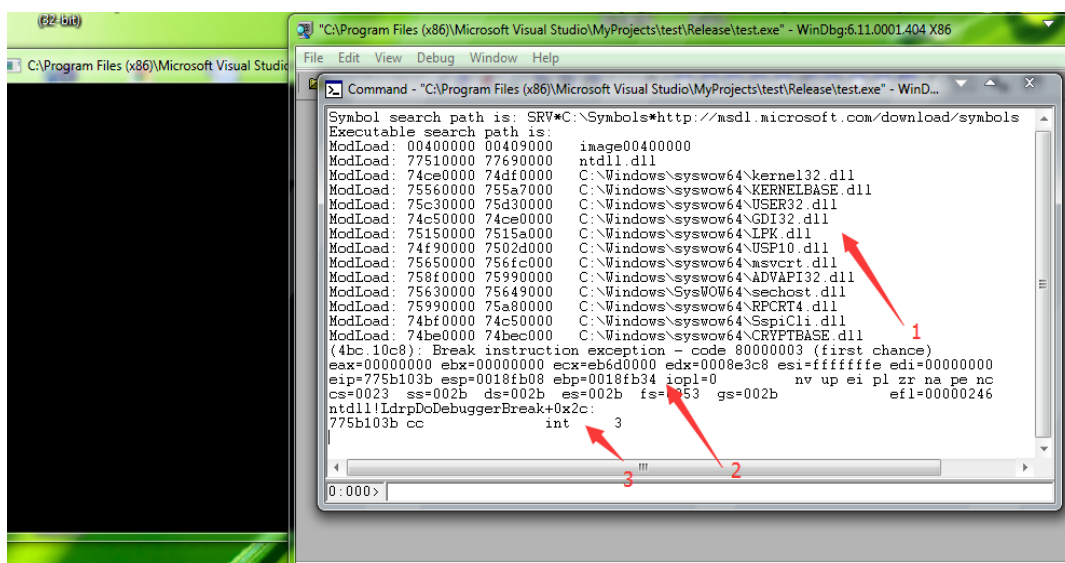
```
    return 0;
```

```
}
```

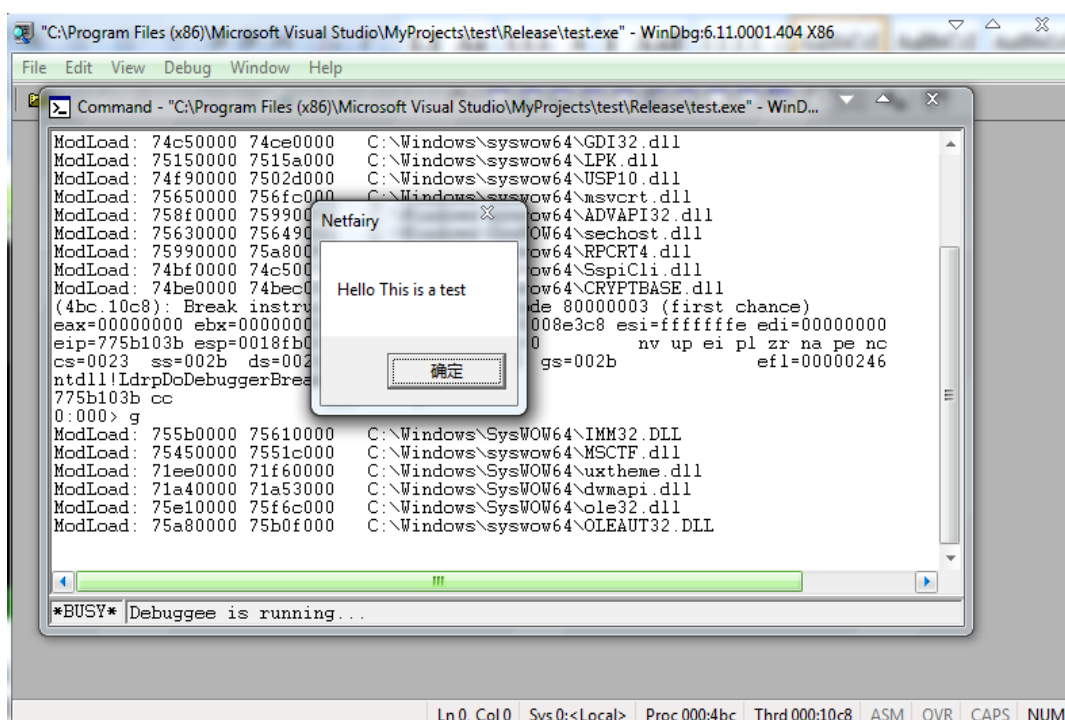
strcpy(buffer,"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA");会造成典型的缓冲区溢出，程序将不能正常返回，我们看看 Windbg 捕获并分析这个异常。编译这个程序，你可以在 C 盘找到这个 test1.exe。用 Windbg 打开



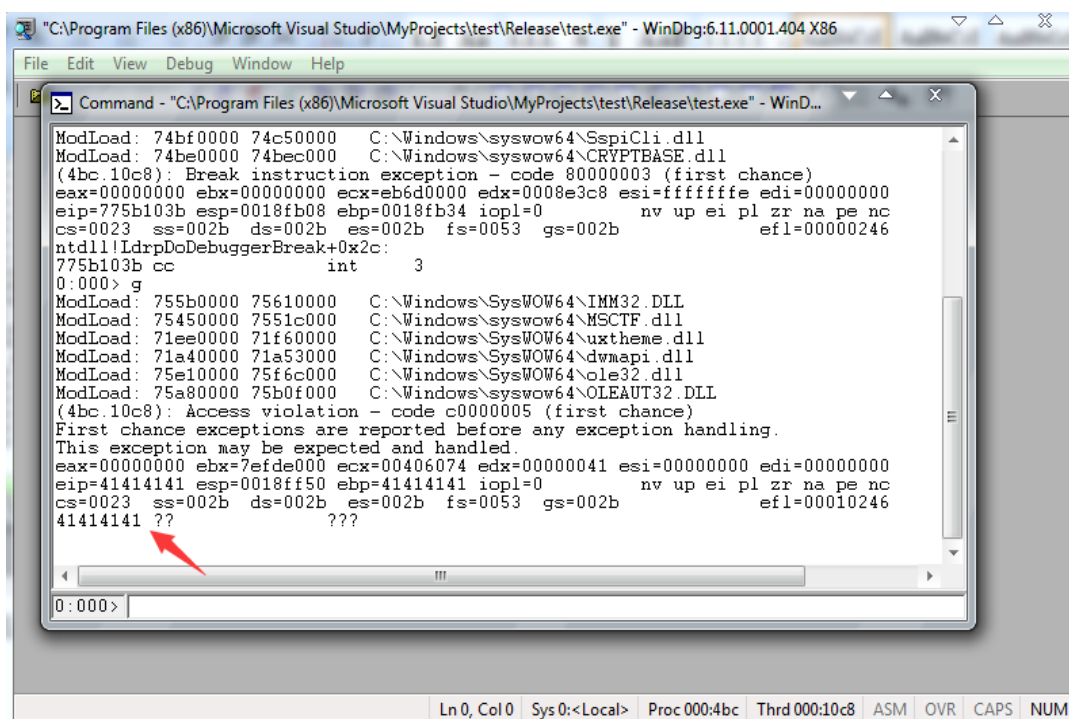
程序中断



这里提供了三个重要信息：1：程序加载的模块列表，其中有模块的加载基址和介绍地址。2：当前各寄存器的值 3：当前执行的指令。我们输入 g 命令让程序跑起来



到这里还没有出现任何异常，但是当我们按下确定之后



Boom!!!程序出错了,程序不知道接下来执行什么。此时的 eip 为 0x41414141, 由模块加载列表可知 0x41414141 不属于任何模块。

```

ModLoad: 00400000 00409000 image00400000
ModLoad: 77510000 77690000 ntdll.dll
ModLoad: 74ce0000 74df0000 C:\Windows\syswow64\kernel32.dll
ModLoad: 75560000 755a7000 C:\Windows\syswow64\KERNELBASE.dll
ModLoad: 75c30000 75d30000 C:\Windows\syswow64\USER32.dll
ModLoad: 74c50000 74ce0000 C:\Windows\syswow64\GDI32.dll
ModLoad: 75150000 7515a000 C:\Windows\syswow64\LPK.dll
ModLoad: 74f90000 7502d000 C:\Windows\syswow64\USP10.dll
ModLoad: 75650000 756fc000 C:\Windows\syswow64\msvcrt.dll
ModLoad: 758f0000 75990000 C:\Windows\syswow64\ADVAPI32.dll
ModLoad: 75630000 75649000 C:\Windows\SysWOW64\sechost.dll
ModLoad: 75990000 75a80000 C:\Windows\syswow64\RPCRT4.dll
ModLoad: 74bf0000 74c50000 C:\Windows\syswow64\SspiCli.dll
ModLoad: 74be0000 74bec000 C:\Windows\syswow64\CRYPTBASE.dll

```

下面用 !analyze -v 分析程序出错原因

```

invalid exception stack at 00000000
0:000> !analyze -v
*****
*
*                               Exception Analysis
*
*****

***                               ***
***                               ***
*** Your debugger is not using the correct symbols                    ***
***                               ***
*** In order for this command to work properly, your symbol path      ***
*** must point to .pdb files that have full type information.         ***
***                               ***
*** Certain .pdb files (such as the public OS symbols) do not        ***
*** contain the required information. Contact the group that          ***
*** provided you with these symbols if you need this command to      ***
*** work.                                                              ***
***                               ***
*** Type referenced: kernel32!pNlsUserInfo                            ***
***                               ***
*****
*****                               *****
*****                               *****
***                               ***
*** Your debugger is not using the correct symbols                    ***
***                               ***
*** In order for this command to work properly, your symbol path      ***
*** must point to .pdb files that have full type information.         ***
***                               ***
*** Certain .pdb files (such as the public OS symbols) do not        ***
*** contain the required information. Contact the group that          ***
*** provided you with these symbols if you need this command to      ***
*** work.                                                              ***
***                               ***
*** Type referenced: kernel32!pNlsUserInfo                            ***
***                               ***
*****

FAULTING_IP:
+e
41414141 ??          ???

EXCEPTION_RECORD: ffffffff -- (.exr 0xffffffffffffffff)
ExceptionAddress: 41414141
ExceptionCode: c0000005 (Access violation)
ExceptionFlags: 00000000
NumberParameters: 2
Parameter[0]: 00000008

```

```
Parameter[1]: 41414141
Attempt to execute non-executable address 41414141

FAULTING_THREAD: 000010c8
PROCESS_NAME: image00400000
ERROR_CODE: (NTSTATUS) 0xc0000005 - 0x%08lx
EXCEPTION_CODE: (NTSTATUS) 0xc0000005 - 0x%08lx
EXCEPTION_PARAMETER1: 00000008
EXCEPTION_PARAMETER2: 41414141
WRITE_ADDRESS: 41414141
FOLLOWUP_IP:
kernel32!BaseThreadInitThunk+e
74cf338a 50          push     eax
FAILED_INSTRUCTION_ADDRESS:
+5ad7952f04e7de04
41414141 ??          ???
NTGLOBALFLAG: 70
APPLICATION_VERIFIER_FLAGS: 0
IP_ON_HEAP: 41414141
The fault address is not in any loaded module, please check your build's re
log at <releasedir>\bin\build_logs\timebuild\ntrebase.log for module which
contain the address if it were loaded.
IP_IN_FREE_BLOCK: 41414141
BUGCHECK_STR: APPLICATION_FAULT_SOFTWARE_NX_FAULT_FILL_PATTERN_41414141
PRIMARY_PROBLEM_CLASS: SOFTWARE_NX_FAULT_FILL_PATTERN_41414141
DEFAULT_BUCKET_ID: SOFTWARE_NX_FAULT_FILL_PATTERN_41414141
FRAME_ONE_INVALID: 1
LAST_CONTROL_TRANSFER: from 41414141 to 41414141
STACK_TEXT:
WARNING: Frame IP not in any known module. Following frames may be wrong.
0018ff4c 41414141 41414141 41414141 00000041 0x41414141
0018ff88 74cf338a 7efde000 0018ffd4 77549f72 0x41414141
0018ff94 77549f72 7efde000 7763a520 00000000 kernel32!BaseThreadInitThunk+0
0018ffd4 77549f45 00401130 7efde000 00000000 ntdll! RtlUserThreadStart+0x7
```

```

0018ffec 00000000 00401130 7efde000 00000000 ntdll!_RtlUserThreadStart+0x1b

SYMBOL_STACK_INDEX: 2
SYMBOL_NAME: kernel32!BaseThreadInitThunk+e
FOLLOWUP_NAME: MachineOwner
MODULE_NAME: kernel32
IMAGE_NAME: kernel32.dll
DEBUG_FLR_IMAGE_TIMESTAMP: 53159a85
STACK_COMMAND: ~0s ; kb
FAILURE_BUCKET_ID: SOFTWARE_NX_FAULT_FILL_PATTERN_41414141_c0000005_kernel:
BUCKET_ID: APPLICATION_FAULT_SOFTWARE_NX_FAULT_FILL_PATTERN_41414141_BAD_I
Followup: MachineOwner
-----

```

ExceptionAddress: 41414141 指明出错地址为 0x41414141。

ExceptionCode: c0000005 (Access violation) 异常代码为 c0000005，这是一个访问异常，因为 0x41414141 不是一个合法的地址。

STACK_TEXT:

WARNING: Frame IP not in any known module. Following frames may be wrong.

```

0018ff4c 41414141 41414141 41414141 00000041
0018ff88 74cf338a 7efde000 0018ffd4 77549f72
0018ff94 77549f72 7efde000 7763a520 00000000
0018ffd4 77549f45 00401130 7efde000 00000000
0018ffec 00000000 00401130 7efde000 00000000

```

显示异常时刻堆栈信息。还有其它很多无关信息，我们无须理会。可以看到 Windbg 捕获到了缓冲区溢出异常。

2. 下面简单介绍一下 python。

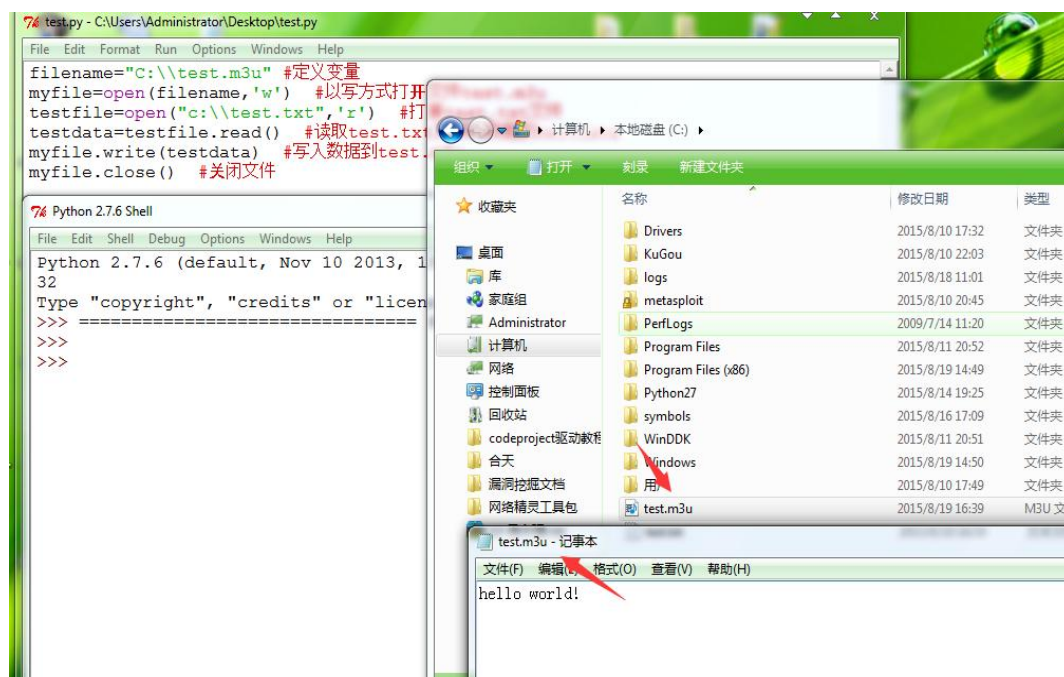
Python 是一种强大的脚本语言，适合用来做漏洞挖掘，它简单，快速，使很多人爱不释手。这里是 python 官网 <https://www.python.org/>。目前 python 有 [python2.x](#) 和 [python3.x](#) 版本有些语法不一样，如输出 hello world 在 python2.7 是 print “hello world”，而在 python3.4 中是 print (“hello world”)，在本系列教程中，我始终用 python2.7。在 windows 下安装 python2.7 十分简单，只需要到官方网站下载 python2.7，然后一路 Next 就行了。在我们这个系列教程中，用的的一个 python 脚本是

```

filename="C:\\test.m3u" #定义变量
myfile=open(filename,'w') #以写方式打开文件 test.m3u
testfile=open("c:\\test.txt",'r') #打开 test.txt 文件
testdata=testfile.read() #读取 test.txt 文件的数据到 testdata
myfile.write(testdata) #写入数据到 test.m3u
myfile.close() #关闭文件

```


我们在桌面新建一个 test.py 文件，复制这段代码进去，然后在 c 盘下新建 test.txt 文件，内容为 Hello world!。然后运行下 test.py 代码看下



C 盘下多了 test.m3u 文件，打开发现里面确实是 hello world!

Python 的强大之处当然不止这里，但是我们这个教程主要用到这段代码，当然，还有别的。

5.2.2. 练习



以下说法不正确的是：【单选题】

- 【A】 Windbg 不能调试驱动程序
- 【B】 python 写的代码不需要编译可以运行
- 【C】 python 中 i=1 这样写不会报错。
- 【D】 windbg 包含普通，元，扩展命令。

答案：A

6 布置一个任务

使用 python 完成一个 socket 通信的实验任务，并对实验结果进行分析，完成思考题目，总结实验的心得体会，并提出实验的改进意见。

7 提示

- 1) python 实现 socket 通信需要用到 socket 这个库
- 2) 既然通信，那么需要有客户端和服务端，需要分开写。

8 配套学习资源

1. Python 实现 socket 通信

<http://www.netfairy.net/?post=157>