



北京邮电大学

Beijing University of Posts and Telecommunications



汇编语言与逆向工程

Assembly Language and Software Reverse Engineering

付俊松
网络空间安全学院

2024年3月



第四章 Hello, world of reverse!

- 一. HelloWorld程序逆向分析
- 二. 快速定位关键函数
- 三. 逆向牛刀小试



第四章 Hello, world of reverse!

(1) HelloWorld程序逆向分析

- 1. HelloWorld程序
- 2. IDA载入分析
- 3. 动态调试



第四章 Hello, world of reverse!

(1) HelloWorld程序逆向分析 — HelloWorld程序

- 用VC++ 6.0编译Hello World程序
 - (VC++ 6.0 Debug版本)

4-1

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

```
Int main() {
```

```
    MessageBox(NULL, "Welcome to the world of reverse!", "Helloworld!", MB_OK);
```

```
    return 0;
```

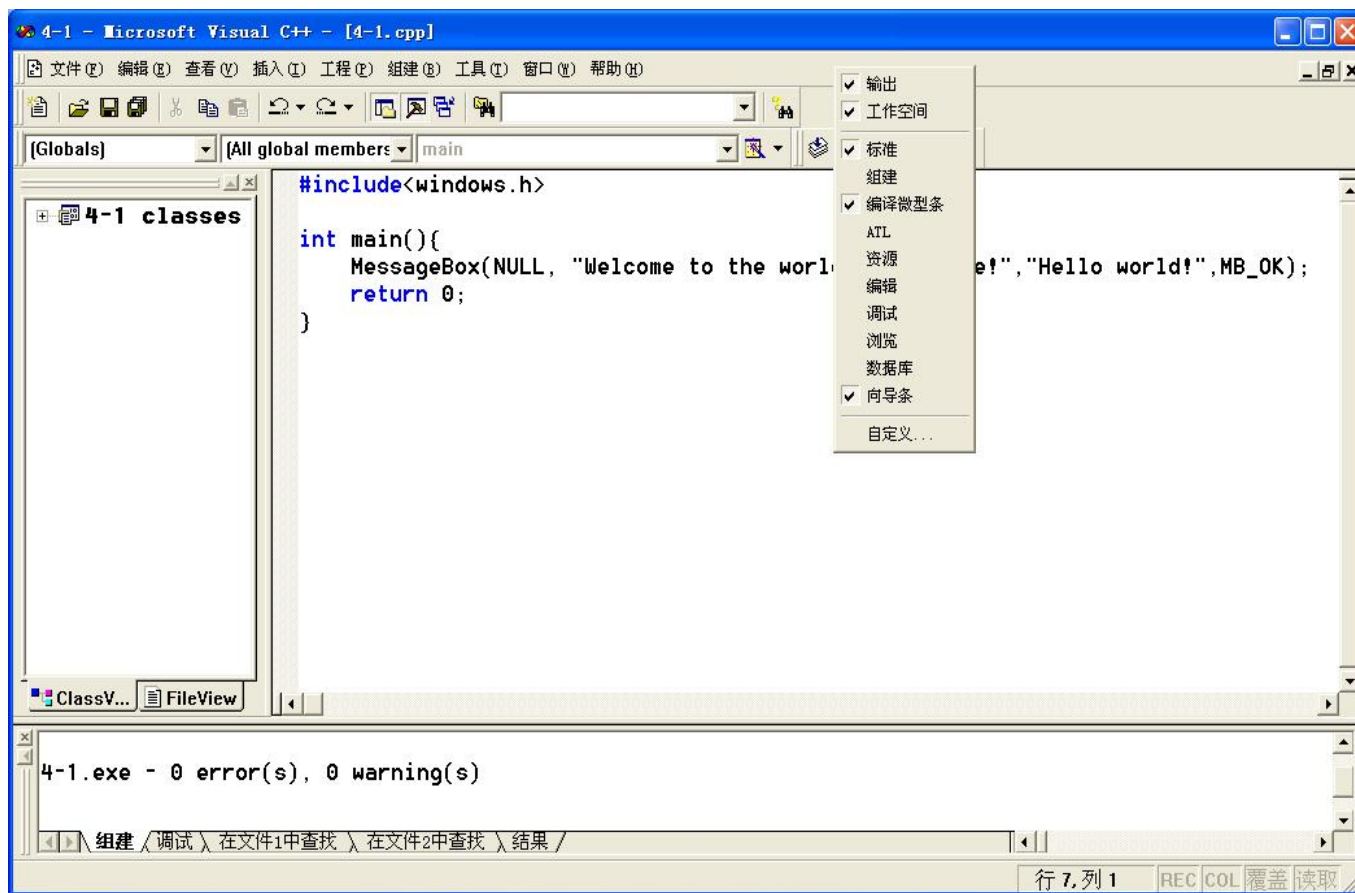
```
}
```



第四章 Hello, world of reverse!

(1) HelloWorld程序逆向分析 — HelloWorld程序

– 在菜单栏中鼠标右击选中“组建”一栏





第四章 Hello, world of reverse!

(1) HelloWorld程序逆向分析 — HelloWorld程序

– 选择编译Release版本的程序

- 相对于Debug版本，Release版本的程序更加简洁，方便调试



- 运行一下程序，会弹出如下图的框





第四章 Hello, world of reverse!

(1) HelloWorld程序逆向分析

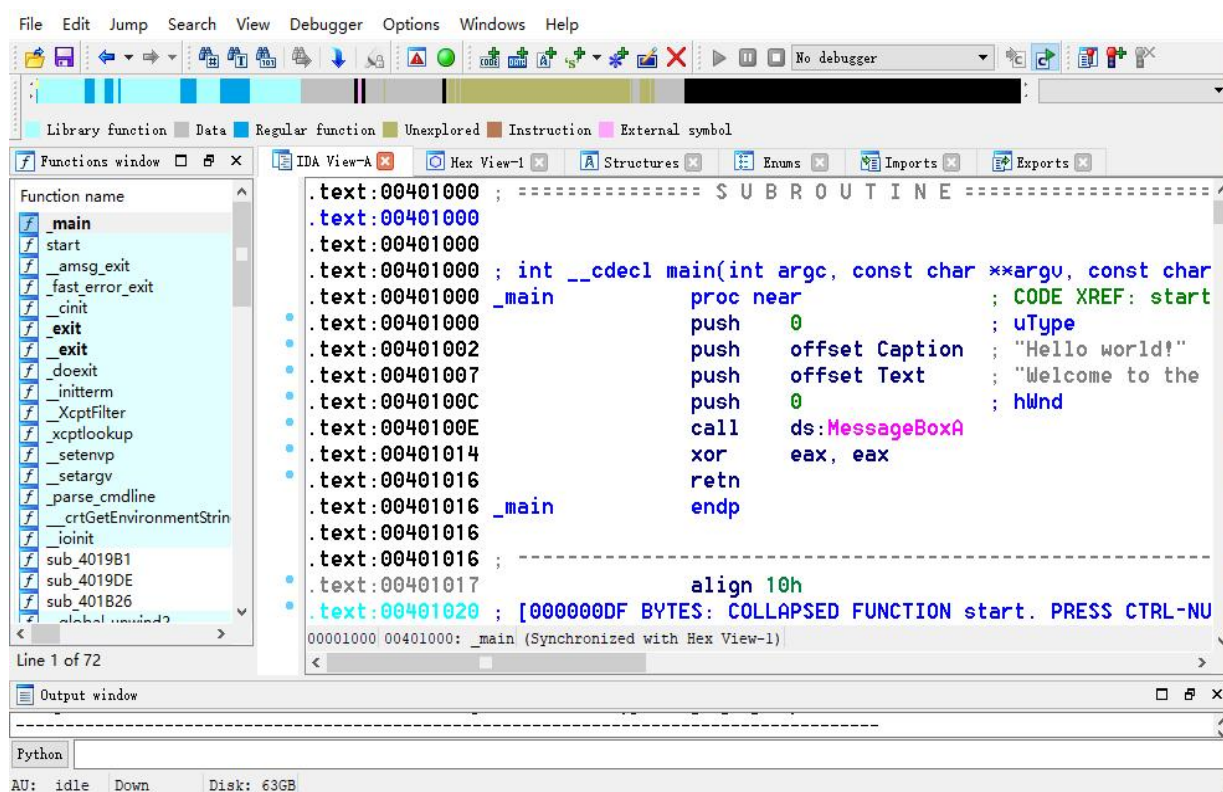
- 1. HelloWorld程序
- 2. IDA载入分析
- 3. 动态调试



第四章 Hello, world of reverse!

(1) HelloWorld程序逆向分析 — IDA载入分析

□ 由于是VC++ 6.0编译的程序，所以用IDA加载该程序后，能识别出main函数





第四章 Hello, world of reverse!

(1) HelloWorld程序逆向分析 — IDA载入分析

- 鼠标选中main函数里的任一区域，F5反编译生成伪代码

The screenshot shows the IDA Pro interface with the main function selected in the Functions list. The disassembly window displays the following code:

```
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     MessageBoxA(0, Text, Caption, 0);
4     return 0;
5 }
```

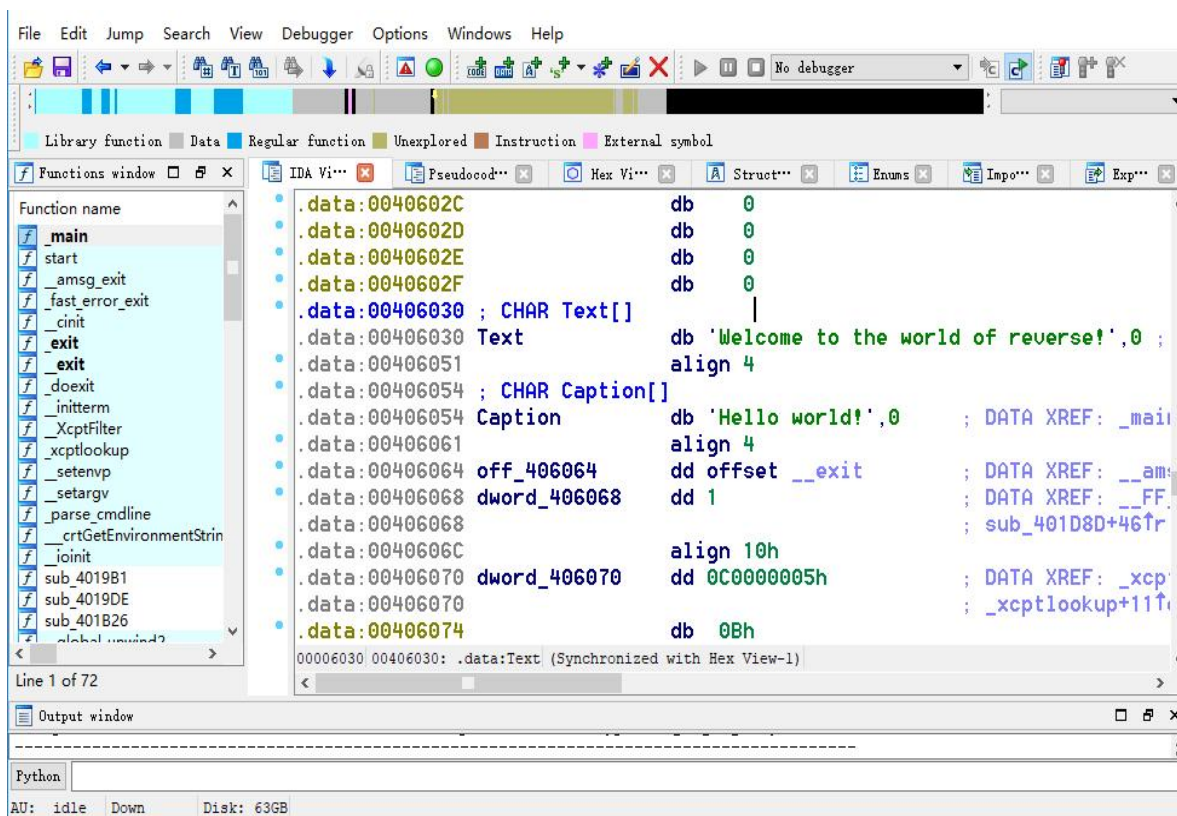
The status bar at the bottom indicates the current position is at 0000100E main:3.



第四章 Hello, world of reverse!

(1) HelloWorld程序逆向分析 — IDA载入分析

- 程序的功能是调用MessageBoxA函数输出消息框，消息框的内容和标题通过双击Text和Caption变量即可查看





第四章 Hello, world of reverse!

(1) HelloWorld程序逆向分析

- 1. HelloWorld程序
- 2. IDA载入分析
- 3. 动态调试



第四章 Hello, world of reverse!

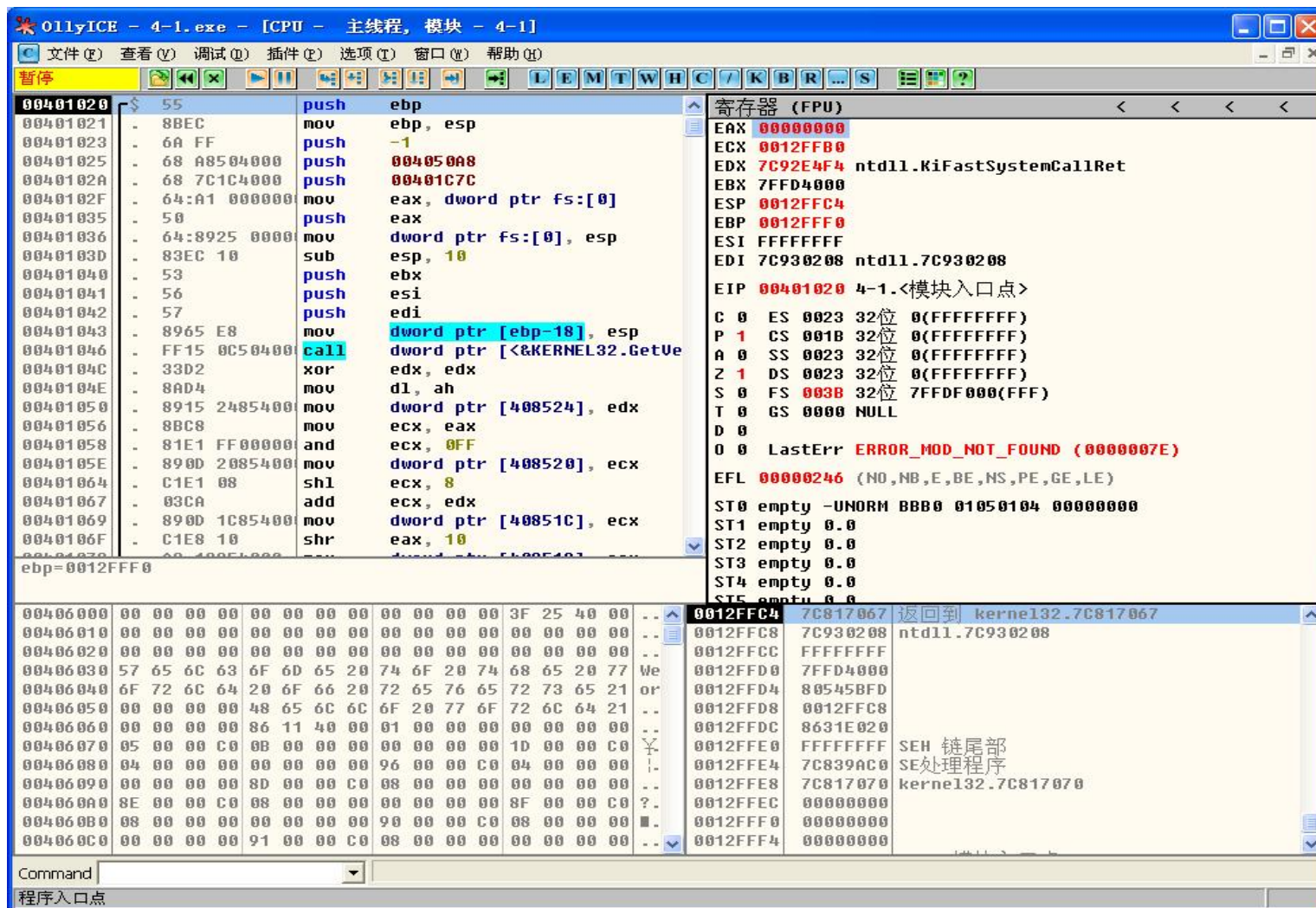
(1) HelloWorld程序逆向分析 — 动态调试

- 动态调试的目的，是利用逆向分析方法，找到主要功能函数，并分析出其功能
- 从程序入口点开始
 - 使用OlllyICE动态调试Hello World程序，来确定main函数的位置
 - 用调试器载入Hello World程序，程序停在地址0x401020处
 - 这便是Hello World程序的代码入口点（EP, Entry Point），即该程序最先执行的代码的起始位置



第四章 Hello, world of reverse!

(1) HelloWorld程序逆向分析 — 动态调试



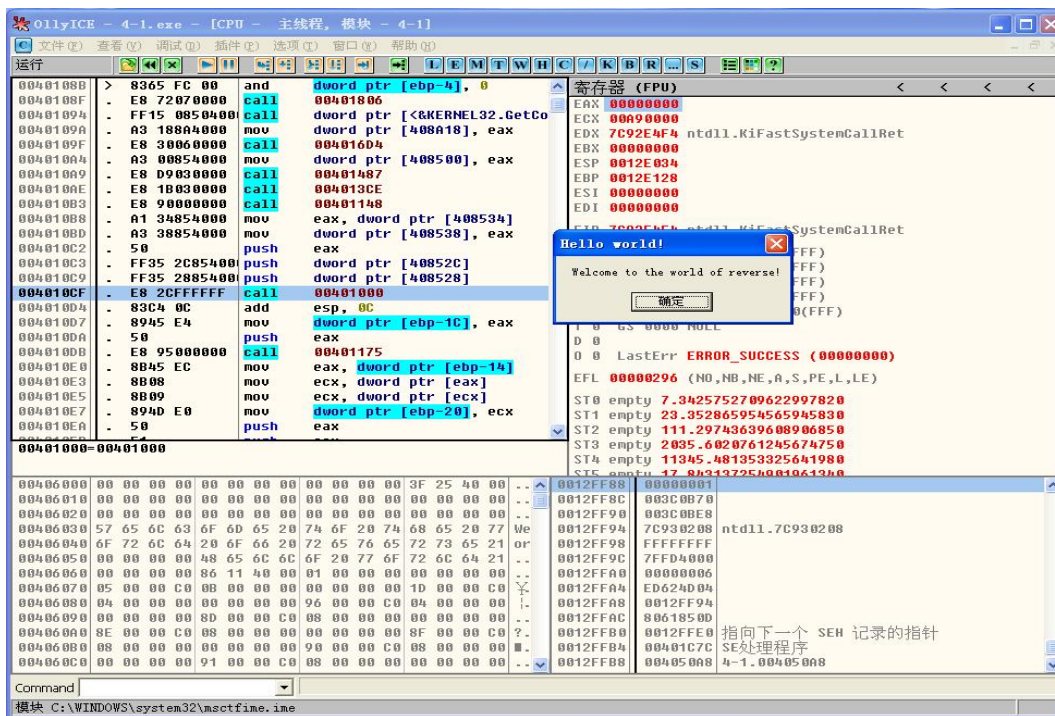


第四章 Hello, world of reverse!

(1) HelloWorld程序逆向分析 — 动态调试

— 单步跟踪

- 一直用F8（单步步过）命令调试Hello World程序
- 直到运行到地址0x4010CF处，程序弹了一个消息框





第四章 Hello, world of reverse!

(1) HelloWorld程序逆向分析 — 动态调试

- 可以猜想，该地址处调用的**0x401000**函数可能就是我们要找的**main**函数
- 为了验证猜想，在**0x4010CF**地址用**F2**命令下断点，重新开始调试程序
- **F9**命令执行到断点处，然后**F7**单步步入，就进入到**0x401000**函数中



第四章 Hello, world of reverse!

(1) HelloWorld程序逆向分析 — 动态调试

```
00401000 6A 00 push 0
00401002 68 54604000 push 00406054
00401007 68 30604000 push 00406030
0040100C 6A 00 push 0
0040100E FF15 9C504000 call dword ptr [<USER32.MessageBoxA>]
00401014 33C0 xor eax, eax
00401016 C3 ret
00401017 90 nop
00401018 90 nop
00401019 90 nop
```

可以看到，0x401000函数里调用了MessageBoxA函数，因此该函数即为main函数

```
00401023 6A FF push -1
00401025 68 A8504000 push 004050A8
0040102A 68 7C1C4000 push 00401C7C
0040102F 64:A1 00000000 mov eax, dword ptr fs:[0]
00401035 50 push eax
00401036 64:8925 00000000 mov dword ptr fs:[0], esp
00401038 6A 10 push 10
0040103A 6A 10 push 10
```

本地调用来自 <模块入口点>+0AF

```
00406000 00 00 00 00 00 00 00 00 00 00 00 00 3F 25 40 00 .....?%
00406010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00406020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00406030 57 65 6C 63 6F 6D 65 20 74 6F 20 74 68 65 20 77 Welcome to the
00406040 6F 72 6C 64 20 6F 66 20 72 65 76 65 72 73 65 21 world of reverse
00406050 00 00 00 00 48 65 6C 6C 6F 20 77 6F 72 6C 64 21 ....Hello world
00406060 00 00 00 00 86 11 40 00 01 00 00 00 00 00 00 00 ....?@.
00406070 05 00 00 C0 00 00 00 00 00 00 00 00 10 00 00 C0 Y.?.?.?.
00406080 04 00 00 00 00 00 00 00 96 00 00 C0 04 00 00 00 |.?.?.?.
00406090 00 00 00 00 80 00 00 C0 08 00 00 00 00 00 00 00 ....?.?.
004060A0 8E 00 00 C0 00 00 00 00 00 00 00 00 8F 00 00 C0 ??.?.?.
004060B0 08 00 00 00 00 00 00 00 90 00 00 C0 08 00 00 00 ■.?.?.?.
004060C0 00 00 00 00 91 00 00 C0 08 00 00 00 00 00 00 00 ....?.?.
```

0012FF84 004010D4 返回到 4-1.<模块入口点>+0AF

0012FF88 00000001

0012FF8C 003C0B70

0012FF90 003C0BE8

0012FF94 7C930208 ntdll.7C930208

0012FF98 FFFFFFFF

0012FF9C 7FFD6000

0012FFA0 00000006

0012FFA4 ED7BED04

0012FFA8 0012FF94

0012FFAC 8061850D

0012FFB0 0012FFE0 指向下一个 SEH 记录的指针

0012FFB4 00401C7C SE处理程序



第四章 Hello, world of reverse!

(1) HelloWorld程序逆向分析 — 动态调试

- 学C和C++的时候，总说main函数是程序的入口函数，但我们在调试的时候发现，从程序的代码入口点到main函数执行之前，程序明明还做了其他操作，这些操作又是什么呢？
- 分析start函数



第四章 Hello, world of reverse!

(1) HelloWorld程序逆向分析 — 动态调试

在IDA中跳转到代码入口地址0x401020处，看到了如图所示的start函数

Assembly code for the start function:

```
t:00401020 public start
t:00401020 start proc near
t:00401020
t:00401020 var_20 = dword ptr -20h
t:00401020 var_1C = dword ptr -1Ch
t:00401020 ms_exc = CPPEH_RECORD ptr -18h

t:00401033 push eax
t:00401036 mov large fs:0, esp
t:0040103D sub esp, 10h
t:00401040 push ebx
t:00401041 push esi
t:00401042 push edi
t:00401043 mov [ebp+ms_exc.old_esp], esp

00001020 00401020: start (Synchronized with Hex View-1)
```

Output window:

```
408A18: using guessed type int dword_408A18;
```

Bottom status bar:

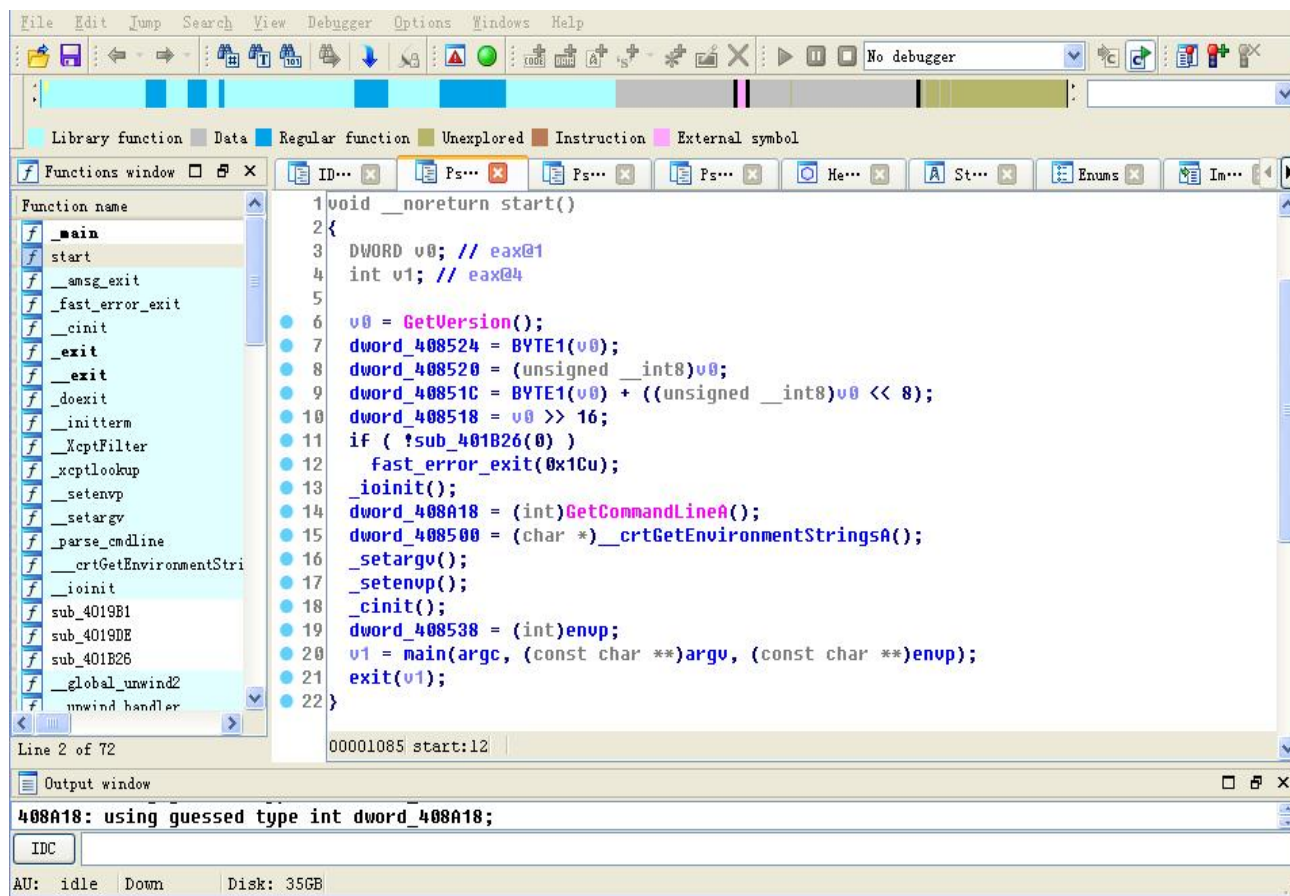
```
AU: idle Down Disk: 35GB
```



第四章 Hello, world of reverse!

(1) HelloWorld程序逆向分析 — 动态调试

– F5生成的伪代码



```
File Edit Jump Search View Debugger Options Windows Help
Library function Data Regular function Unexplored Instruction External symbol
Function name
main
start
_amsi_exit
_fast_error_exit
_cinit
_exit
_doexit
_initterm
_xcptFilter
_xcptlookup
_setenvp
_setargv
_parse_cmdline
__crtGetEnvironmentStri
_ioinit
sub_4019B1
sub_4019DE
sub_401B26
_global_unwind2
unwind handler
1 void __noreturn start()
2 {
3     DWORD v0; // eax@1
4     int v1; // eax@4
5
6     v0 = GetVersion();
7     dword_408524 = BYTE1(v0);
8     dword_408528 = (unsigned __int8)v0;
9     dword_40851C = BYTE1(v0) + ((unsigned __int8)v0 << 8);
10    dword_408518 = v0 >> 16;
11    if ( !sub_401B26(0) )
12        fast_error_exit(0x1Cu);
13    _ioinit();
14    dword_408A18 = (int)GetCommandLineA();
15    dword_408508 = (char *)__crtGetEnvironmentStringsA();
16    _setargv();
17    _setenvp();
18    _cinit();
19    dword_408538 = (int)envp;
20    v1 = main(argc, (const char **)argv, (const char **)envp);
21    exit(v1);
22 }
```

Line 2 of 72 00001085 start:12

Output window

408A18: using guessed type int dword_408A18;

IDC

AU: idle Down Disk: 35GB



第四章 Hello, world of reverse!

(1) HelloWorld程序逆向分析 — 动态调试

- 分析start函数的伪代码可以得知，它在执行一些初始化操作
 - 如获取命令行参数、获取环境变量值、初始化全局变量等，一切准备工作完成之后，再调用main函数



第四章 Hello, world of reverse!

- 一. HelloWorld程序逆向分析
- 二. 快速定位关键函数
- 三. 逆向牛刀小试



第四章 Hello, world of reverse!

(2) 快速定位关键函数

- 1. 长驱直入法
- 2. 字符串查找法
- 3. API引用法
- 4. API断点法



第四章 Hello, world of reverse!

(2) 快速定位关键函数 — 长驱直入法

□长驱直入法的原理

- 当程序功能非常明确时，从程序入口处一步一步分析，逐条执行指令，直到找到关键函数
- 不过长驱直入法仅适用于被调试的代码量不大、且程序功能明确的情况
 - 倘若被调试的程序比较复杂时，这种方法就不适合了



第四章 Hello, world of reverse!

(2) 快速定位关键函数 — 长驱直入法

- 运行程序的时候会弹出消息对话框，因此通过确定 **MessageBox** 函数的调用位置,即可确定 **main** 函数的位置
- 可以通过单步调试 **HelloWorld** 这个程序（**F8**, **Step Over**），当执行到某条指令时，程序弹出消息对话框，就能锁定 **main** 函数的位置



第四章 Hello, world of reverse!

(2) 快速定位关键函数

- 1. 长驱直入法
- 2. 字符串查找法
- 3. API引用法
- 4. API断点法



第四章 Hello, world of reverse!

(2) 快速定位关键函数 — 字符串查找法

- 程序运行时弹出的消息对话框上显示了两个字符串，一个是标题“Hello world!”，一个是内容“Welcome to the world of reverse!”



- 可以通过查找这两个字符串来确定main函数的位置



第四章 Hello, world of reverse!

(2) 快速定位关键函数 — 字符串查找法

□ 字符串查找法

- IDA PRO的字符串查找法
- OllyDbg的字符串查找法



第四章 逆向初体验

(2) 快速定位关键函数 — 字符串查找法

□ IDA PRO的字符串查找法

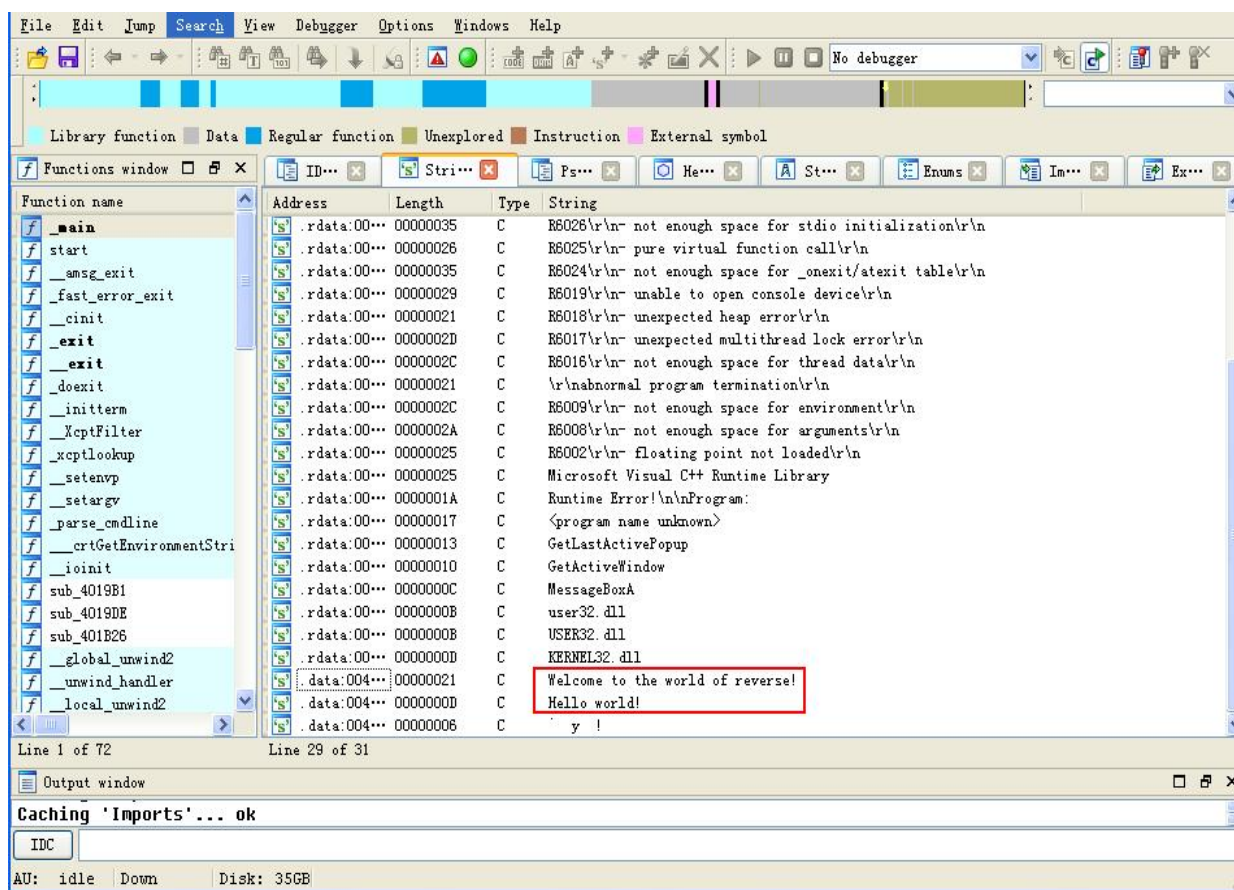
- 用IDA PRO加载目标程序后，通过三连击迅速确定关键函数位置
- 三连击：“Shift F12 + x + F5”



第四章 Hello, world of reverse!

(2) 快速定位关键函数 — 字符串查找法

— (1) Shift + F12搜索字符串



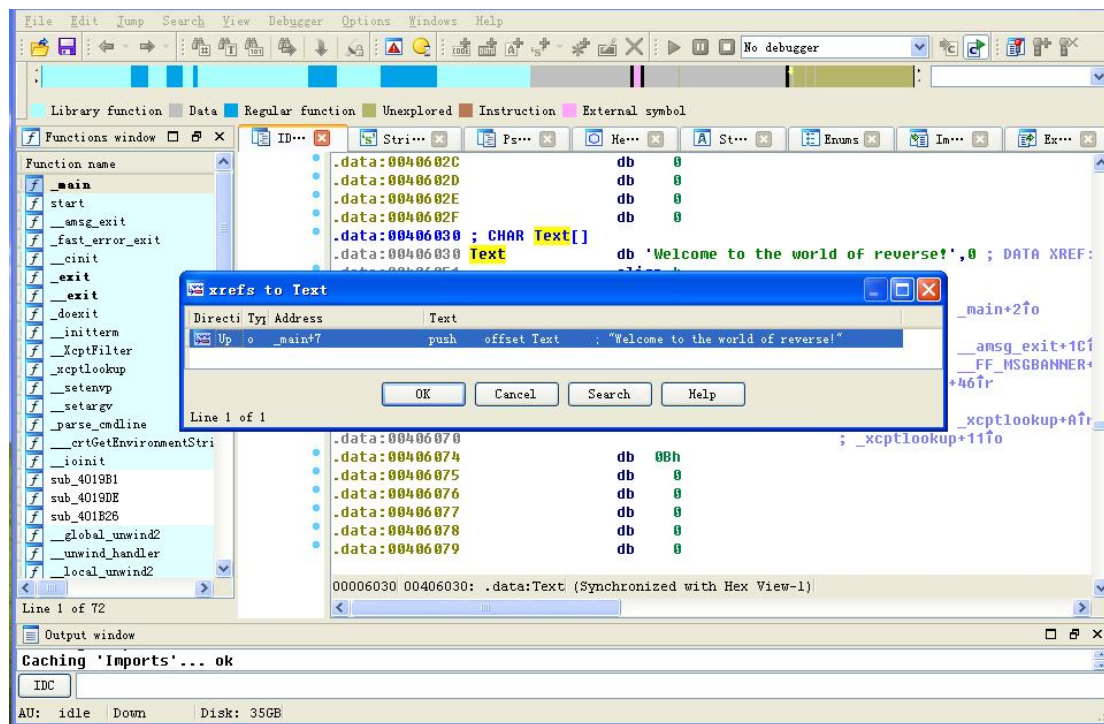


第四章 Hello, world of reverse!

(2) 快速定位关键函数 — 字符串查找法

— (2) x查看交叉引用

- 选中一个字符串，双击跳转到汇编窗口，然后按快捷键x查看调用该字符串的地方





第四章 Hello, world of reverse!

(2) 快速定位关键函数 — 字符串查找法

➤ 点击OK即可来到关键函数的汇编代码段

```
File Edit Jump Search View Debugger Options Windows Help
Library function Data Regular function Unexplored Instruction External symbol
Functions window
Function name
f _main
f start
f _amsg_exit
f _fast_error_exit
f _cinit
f _exit
f _exit
f _doexit
f _initterm
f _XcptFilter
f _xcptlookup
f _setenvp
f _setargv
f _parse_cmdline
f _crtGetEnvironmentStri
f _ioinit
f sub_4019B1
f sub_4019DE
f sub_401B26
f _global_unwind2
f _unwind_handler
f _local_unwind2
Line 1 of 72
Output window
Caching 'Imports'... ok
IDC
AU: idle Down Disk: 35GB
```

```
.text:00401000 ; int __cdecl main(int argc, const char **argv, const char **envp)
.text:00401000 _main
.text:00401000 proc near
.text:00401002 push 0 ; CODE XREF: start+AF↓p
.text:00401007 push offset Caption ; uType
.text:0040100C push offset Text ; "Hello world!"
.text:0040100E push 0 ; "Welcome to the world of r
.text:00401014 call ds:MessageBoxA ; hWnd
.text:00401016 xor eax, eax
.text:00401016 retn
.text:00401016 _main
.text:00401016 endp
.text:00401016 ; -----
.text:00401017 align 10h
.text:00401020 ; ===== SUBROUTINE =====
.text:00401020 ; Attributes: library function noreturn bp-based frame
.text:00401020 ;
.text:00401020 public start
.text:00401020 proc near
.text:00401020 start
.text:00401020 var_20 = dword ptr -20h
0000100C 0040100C: _main+C (Synchronized with Hex View-1)
```

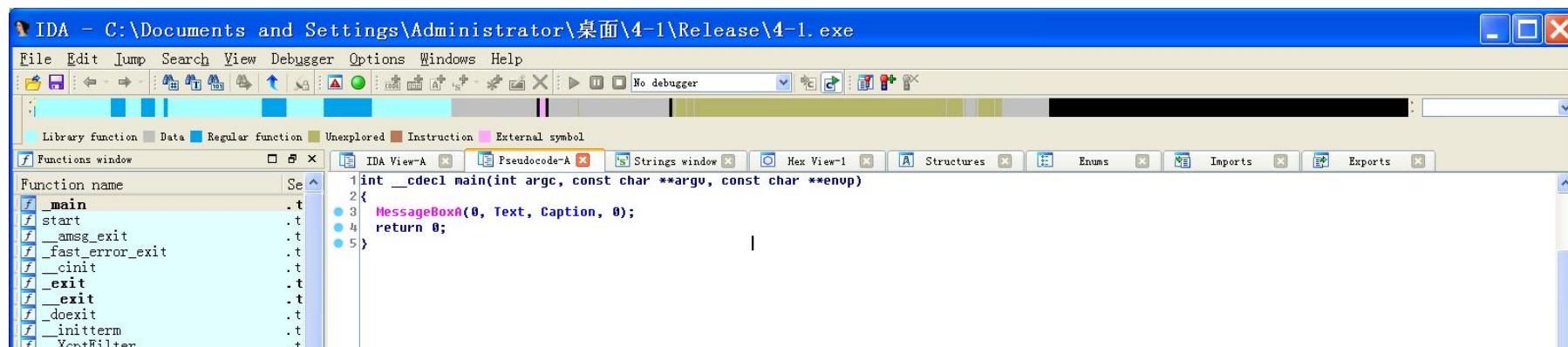


第四章 Hello, world of reverse!

(2) 快速定位关键函数 — 字符串查找法

— (3) F5生成伪代码

- 在关键函数处，按快捷键F5生成伪代码，然后分析关键函数的代码逻辑





第四章 Hello, world of reverse!

(2) 快速定位关键函数 — 字符串查找法

□ 字符串查找法

- IDA PRO的字符串查找法
- OllyDbg的字符串查找法

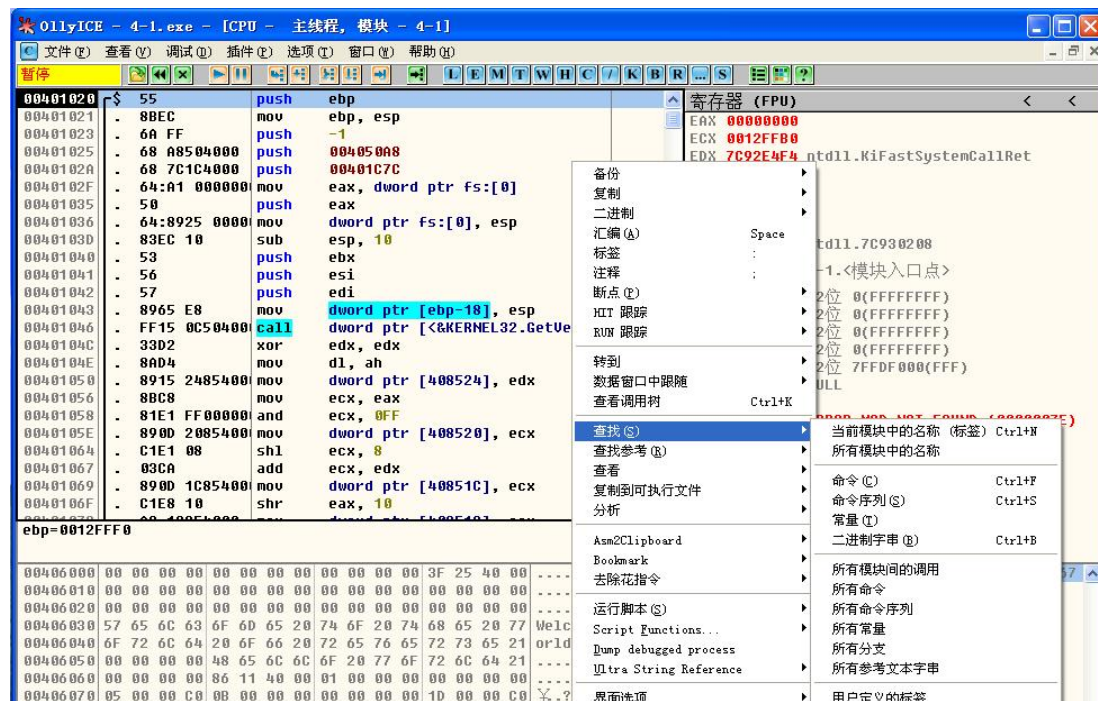


第四章 Hello, world of reverse!

(2) 快速定位关键函数 — 字符串查找法

□ OllyDbg的字符串查找法

- OllyDbg加载目标程序后，鼠标右键菜单—>查找—>所有参考文本字符串



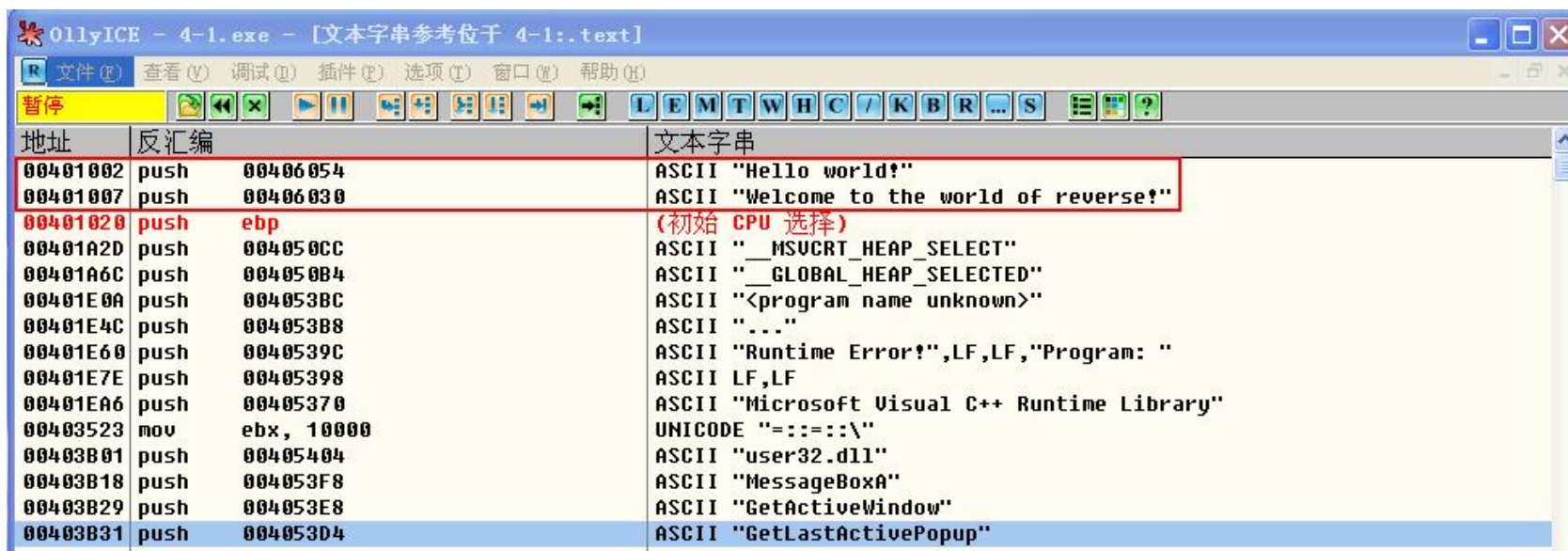


第四章 Hello, world of reverse!

(2) 快速定位关键函数 — 字符串查找法

□ OllyDbg的字符串查找法

– 字符串查找结果



地址	反汇编	文本字符串
00401002	push 00406054	ASCII "Hello world!"
00401007	push 00406030	ASCII "Welcome to the world of reverse!"
00401020	push ebp	(初始 CPU 选择)
00401A2D	push 004050CC	ASCII "__MSVCRT_HEAP_SELECT"
00401A6C	push 004050B4	ASCII "__GLOBAL_HEAP_SELECTED"
00401E0A	push 004053BC	ASCII "<program name unknown>"
00401E4C	push 004053B8	ASCII "..."
00401E60	push 0040539C	ASCII "Runtime Error!",LF,LF,"Program: "
00401E7E	push 00405398	ASCII LF,LF
00401EA6	push 00405370	ASCII "Microsoft Visual C++ Runtime Library"
00403523	mov ebx, 10000	UNICODE "=::::\\"
00403B01	push 00405404	ASCII "user32.dll"
00403B18	push 004053F8	ASCII "MessageBoxA"
00403B29	push 004053E8	ASCII "GetActiveWindow"
00403B31	push 004053D4	ASCII "GetLastActivePopup"



第四章 Hello, world of reverse!

(2) 快速定位关键函数 — 字符串查找法

□ OllyDbg的字符串查找法

- 双击 “Hello world!” 或者 “Welcome to the world of reverse!” 即可来到main函数位置

OllyICE - 4-1.exe - [CPU - 主线程, 模块 - 4-1]

文件(F) 查看(V) 调试(D) 插件(P) 选项(O) 窗口(W) 帮助(H)

暂停

地址	汇编	注释
00401000	6A 00	push 0
00401002	68 54604000	push 00406054
00401007	68 30604000	push 00406030
0040100C	6A 00	push 0
0040100E	FF15 9C504000	call dword ptr [&USER32.MessageBoxA]
00401014	33C0	xor eax, eax
00401016	C3	ret
00401017	90	nop
00401018	90	nop
00401019	90	nop
0040101A	90	nop
0040101B	90	nop
0040101C	90	nop
0040101D	90	nop
0040101E	90	nop
0040101F	90	nop
00401020	55	push ebp
00401021	8BEC	mov ebp, esp
00401023	6A FF	push -1
00401025	68 A8504000	push 004050A8
0040102A	68 7C1C4000	push 00401C7C
0040102F	64:A1 00000000	mov eax, dword ptr fs:[0]
00401035	50	push eax
00401036	64:8925 0000	mov dword ptr fs:[0], esp
0040103D	83EC 10	sub esp, 10
00401040	53	push ebx
00401041	56	push esi
00401042	57	push edi
00401043	8065 F8	mov dword ptr [ebp-18], esp

Style = MB_OK|MB_APPLMODAL
Title = "Hello world!"
Text = "Welcome to the world of reverse!"
hOwner = NULL
MessageBoxA

SE 处理程序安装

00406054=00406054 (ASCII "Hello world!")



第四章 Hello, world of reverse!

(2) 快速定位关键函数

- 1. 长驱直入法
- 2. 字符串查找法
- 3. API引用法
- 4. API断点法



第四章 Hello, world of reverse!

(2) 快速定位关键函数 — API引用法

- Windows编程中，有些功能需要通过调用Win32 API来实现，认真观察一个程序的功能后，能够大致推测出它在运行时调用了哪些Win32 API
- 以HelloWorld程序为例，它在运行时会弹出一个消息窗口，因此推断出该程序调用了MessageBox函数，通过查找哪里调用了该API即可确定关键函数的位置。



第四章 Hello, world of reverse!

(2) 快速定位关键函数 — API引用法

- (1) IDA PRO
- (2) OllyDbg



第四章 Hello, world of reverse!

(2) 快速定位关键函数 — API引用法

□ (1) IDA PRO

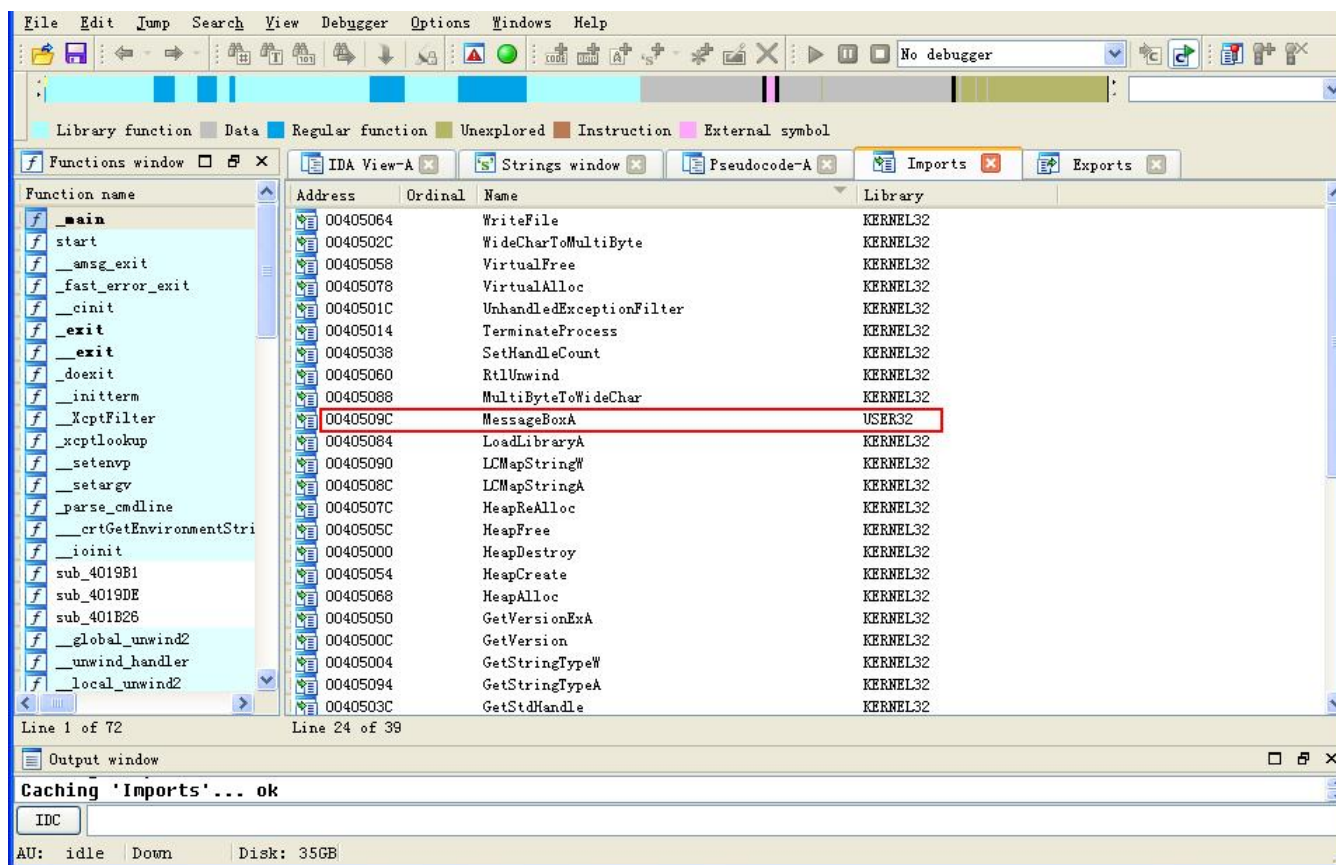
- IDA中通过导入函数窗口（Imports）可以迅速找到MessageBoxA函数（因为HelloWorld程序编译时用的是ASCII字符集，所以调用的是MessageBoxA函数，若是Unicode字符串，调用的就是MessageBoxW函数）



第四章 Hello, world of reverse!

(2) 快速定位关键函数 — API引用法

□ (1) IDA PRO



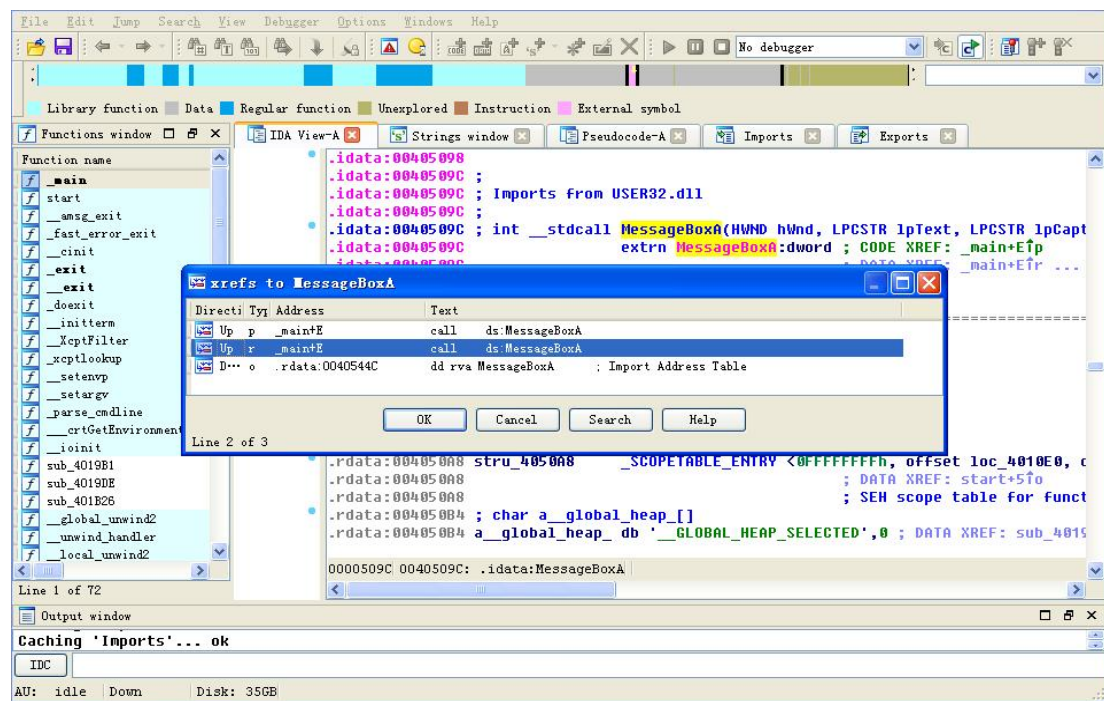


第四章 Hello, world of reverse!

(2) 快速定位关键函数 — API引用法

□ (1) IDA PRO

- IDA PRO双击MessageBoxA函数来到汇编代码窗口，通过查看交叉引用即可确定关键函数位置





第四章 Hello, world of reverse!

(2) 快速定位关键函数 — API引用法

□ (2) OllyDbg

- OllyDbg加载目标程序后，鼠标右键菜单—>查找—>所有模块间的调用

地址	反汇编	目标文件
0040100E	call dword ptr [<&USER32.MessageBoxA]	USER32.MessageBoxA
0040103D	sub esp, 10	(初始 CPU 选择)
00401046	call dword ptr [<&KERNEL32.GetVersion]	kernel32.GetVersion
00401094	call dword ptr [<&KERNEL32.GetCommandLineA]	kernel32.GetCommandLineA
00401141	call dword ptr [<&KERNEL32.ExitProcess]	kernel32.ExitProcess
004011A7	call dword ptr [<&KERNEL32.GetCurrentProcess]	kernel32.GetCurrentProcess
004011AE	call dword ptr [<&KERNEL32.TerminateProcess]	kernel32.TerminateProcess
00401228	call dword ptr [<&KERNEL32.UnhandledExceptionFilter]	kernel32.UnhandledExceptionFilter
00401382	call dword ptr [<&KERNEL32.GetModuleFileNameA]	kernel32.GetModuleFileNameA

- 双击MessageBoxA即可来到调用它的地址处（0x0040100E），即可确定关键函数位置



第四章 Hello, world of reverse!

(2) 快速定位关键函数

- 1. 长驱直入法
- 2. 字符串查找法
- 3. API引用法
- 4. API断点法

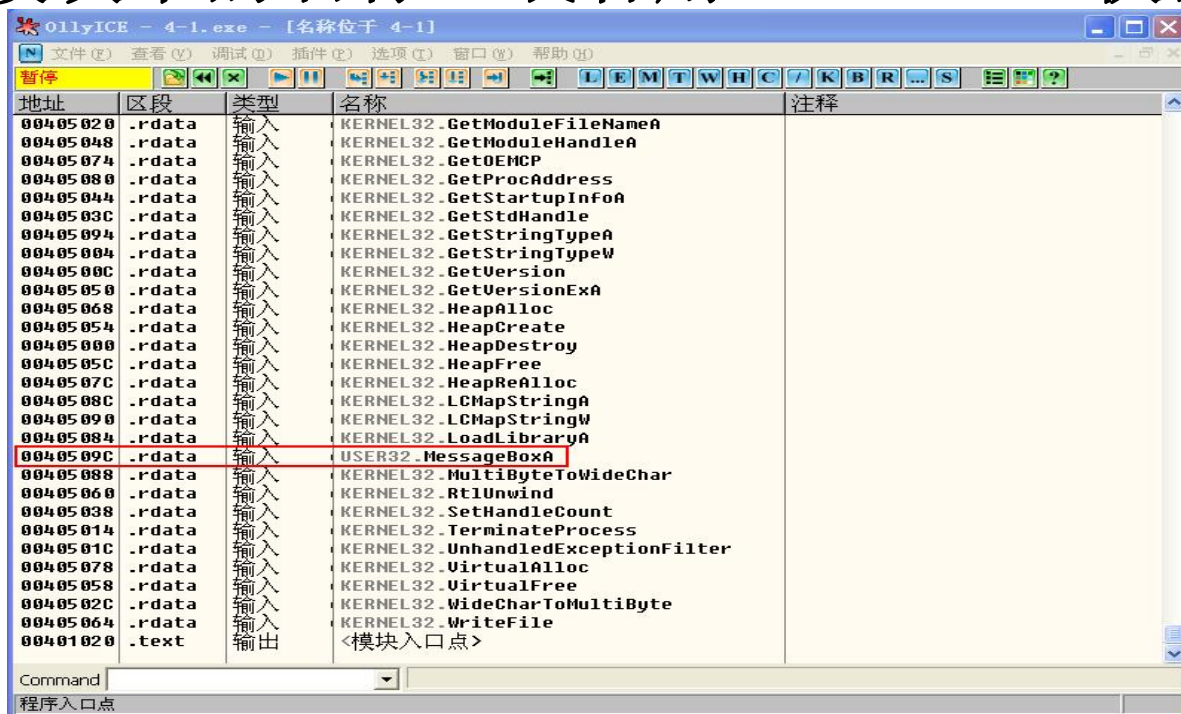


第四章 Hello, world of reverse!

(2) 快速定位关键函数 — API断点法

□ 通过在API上设置断点来确定关键函数位置

- 用OllyDbg加载目标程序，鼠标右键菜单—>查找—>当前模块中的名称，或者用“ctrl + n”快捷键

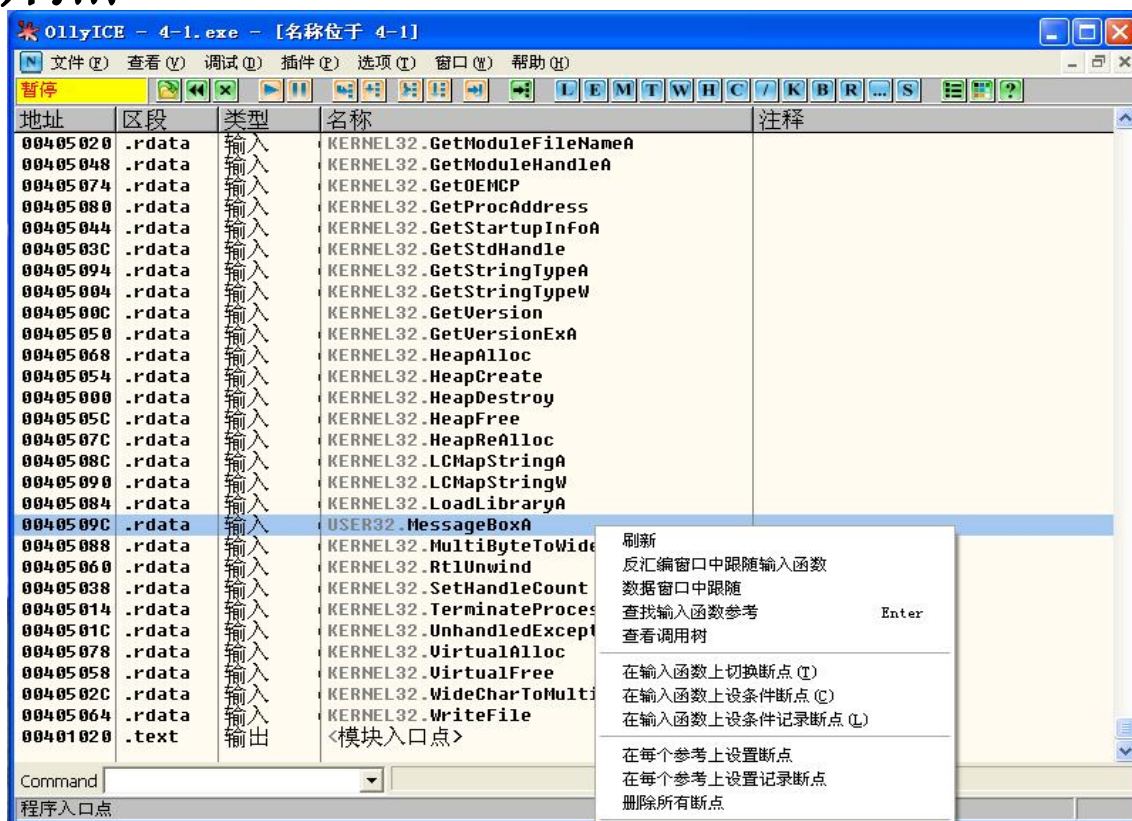




第四章 Hello, world of reverse!

(2) 快速定位关键函数 — API断点法

- 鼠标右键MessageBoxA函数，选择“在每个参考上设置断点”





第四章 Hello, world of reverse!

(2) 快速定位关键函数 — API断点法

– F9运行程序，便会在关键函数处停下来

The screenshot shows the OllyICE debugger interface. The title bar reads "OllyICE - 4-1.exe - [CPU - 主线程, 模块 - 4-1]". The menu bar includes "文件(F)", "查看(V)", "调试(D)", "插件(P)", "选项(O)", "窗口(W)", and "帮助(H)". The toolbar contains various icons for debugging, including "暂停" (Pause), "单步" (Step), "断点" (Breakpoint), and "窗口" (Window). The assembly window displays the following code:

```
00401000 $ 6A 00 push 0
00401002 . 68 54604000 push 00406054
00401007 . 68 30604000 push 00406030
0040100C . 6A 00 push 0
0040100E . FF15 9C504000 call dword ptr [<&USER32.MessageBoxA]
00401014 . 33C0 xor eax, eax
00401016 . C3 retn
00401017 . 90 nop
00401018 . 90 nop
00401019 . 90 nop
0040101A . 90 nop
0040101B . 90 nop
0040101C . 90 nop
0040101D . 90 nop
0040101E . 90 nop
0040101F . 90 nop
00401020 $ 55 push ebp
00401021 . 8BEC mov ebp, esp
00401023 . 6A FF push -1
00401025 . 68 A8504000 push 004050A8
0040102A . 68 7C1C4000 push 00401C7C
0040102F . 64:A1 00000000 mov eax, dword ptr fs:[0]
00401035 . 50 push eax
00401036 . 64:8925 00000000 mov dword ptr fs:[0], esp
0040103D . 83EC 10 sub esp, 10
00401040 . 53 push ebx
00401041 . 56 push esi
00401042 . 57 push edi
00401043 . 8965 E8 mov dword ptr [ebp-18], esp
```

The right-hand pane shows the parameters for the selected `MessageBoxA` function:

```
Style = MB_OK|MB_APPLMODAL
Title = "Hello world!"
Text = "Welcome to the world of reverse!"
hOwner = NULL
```

Below the assembly window, the status bar displays: `ds:[0040509C]=77D507EA (USER32.MessageBoxA)`.



第四章 Hello, world of reverse!

- 一. HelloWorld程序逆向分析
- 二. 快速定位关键函数
- 三. 逆向牛刀小试



第四章 Hello, world of reverse!

(3) 逆向牛刀小试

- 以**firstRe.exe**作为本章的练习题目，更好的练习如何定位关键函数代码段



第四章 Hello, world of reverse!

(3) 逆向牛刀小试

- 1. 定位关键函数
- 2. 程序逻辑分析



第四章 Hello, world of reverse!

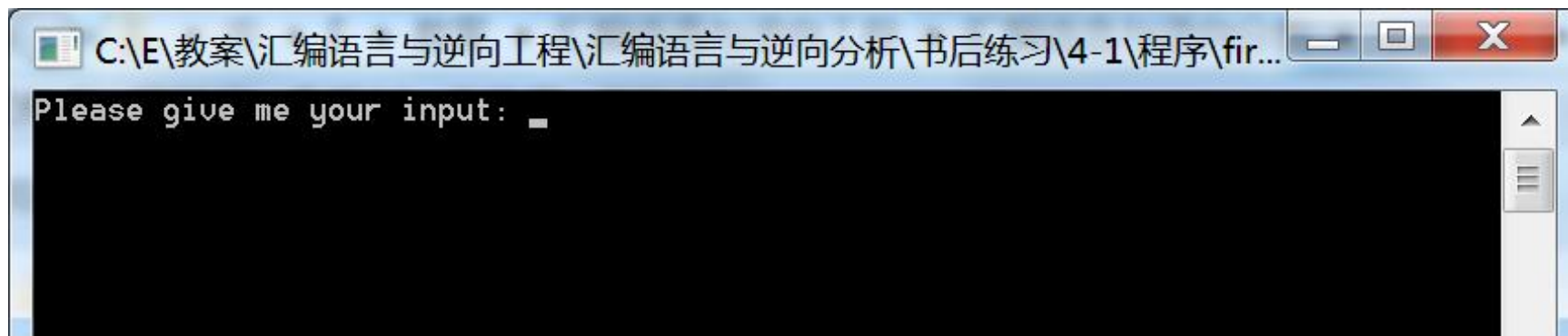
(3) 逆向牛刀小试 — 定位关键函数

- 运行程序，会让我们输入一串字符串，然后会提示我们输入错误

D:\>firstRe.exe

Please give me your input: aaaaaaaaaa

Sorry! You are wrong!!!

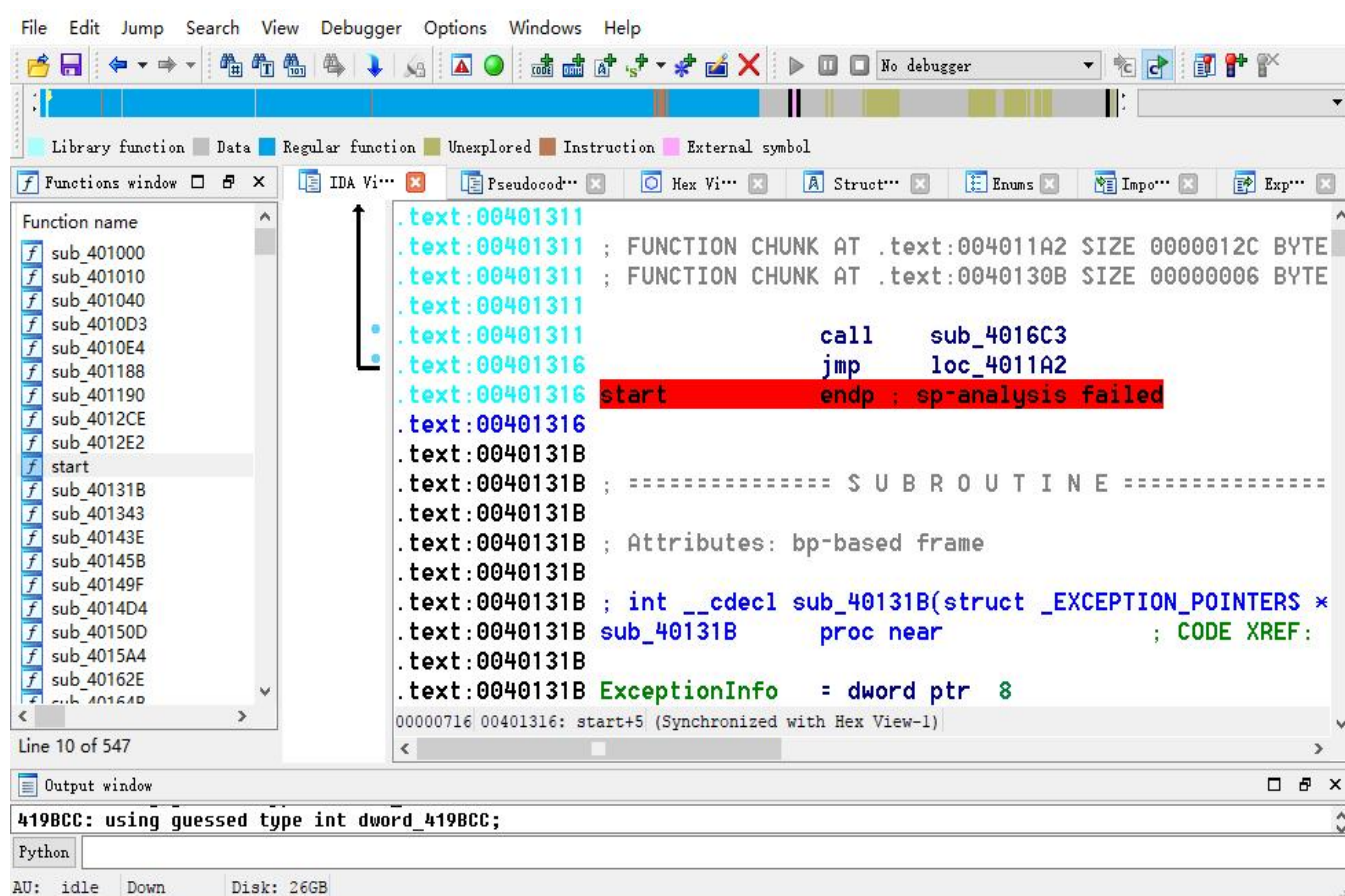




第四章 Hello, world of reverse!

(3) 逆向牛刀小试 — 定位关键函数

□ 用IDA加载目标程序，在函数名称一栏并没能找到main函数

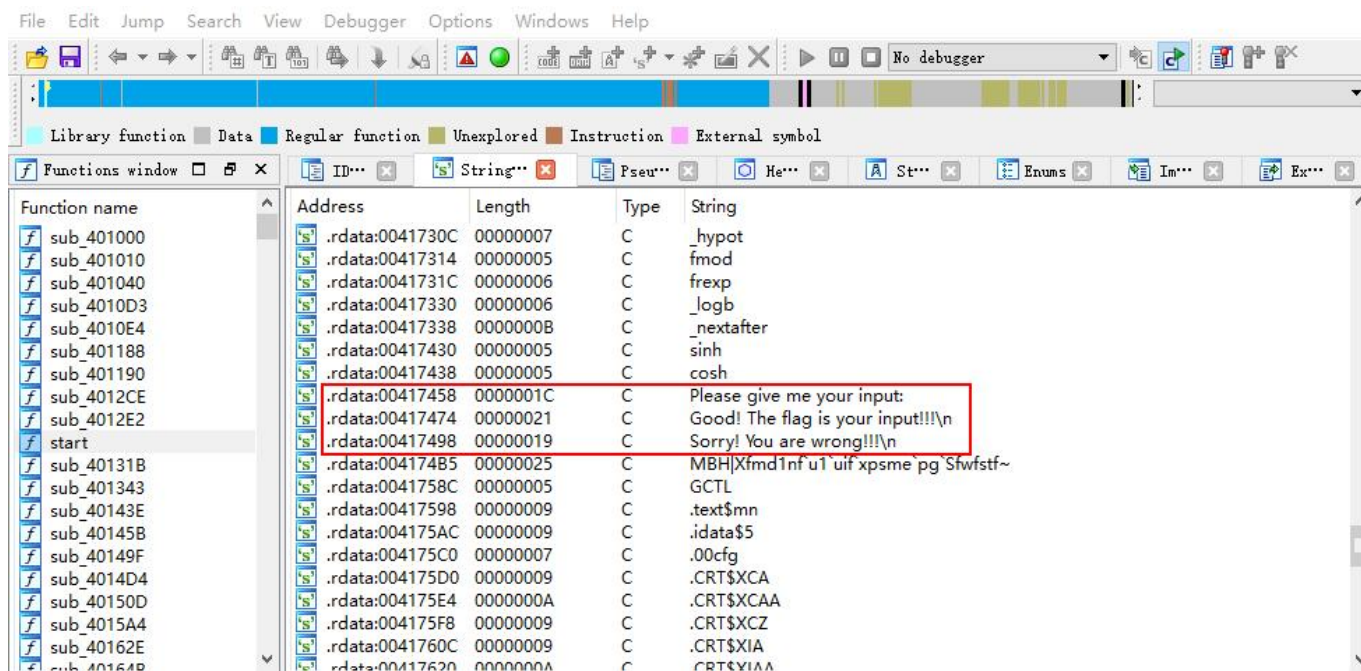




第四章 Hello, world of reverse!

(3) 逆向牛刀小试 — 定位关键函数

□ 通过字符串查找法来确定关键函数代码段，按快捷键“**Shift + F12**”搜索字符串





第四章 Hello, world of reverse!

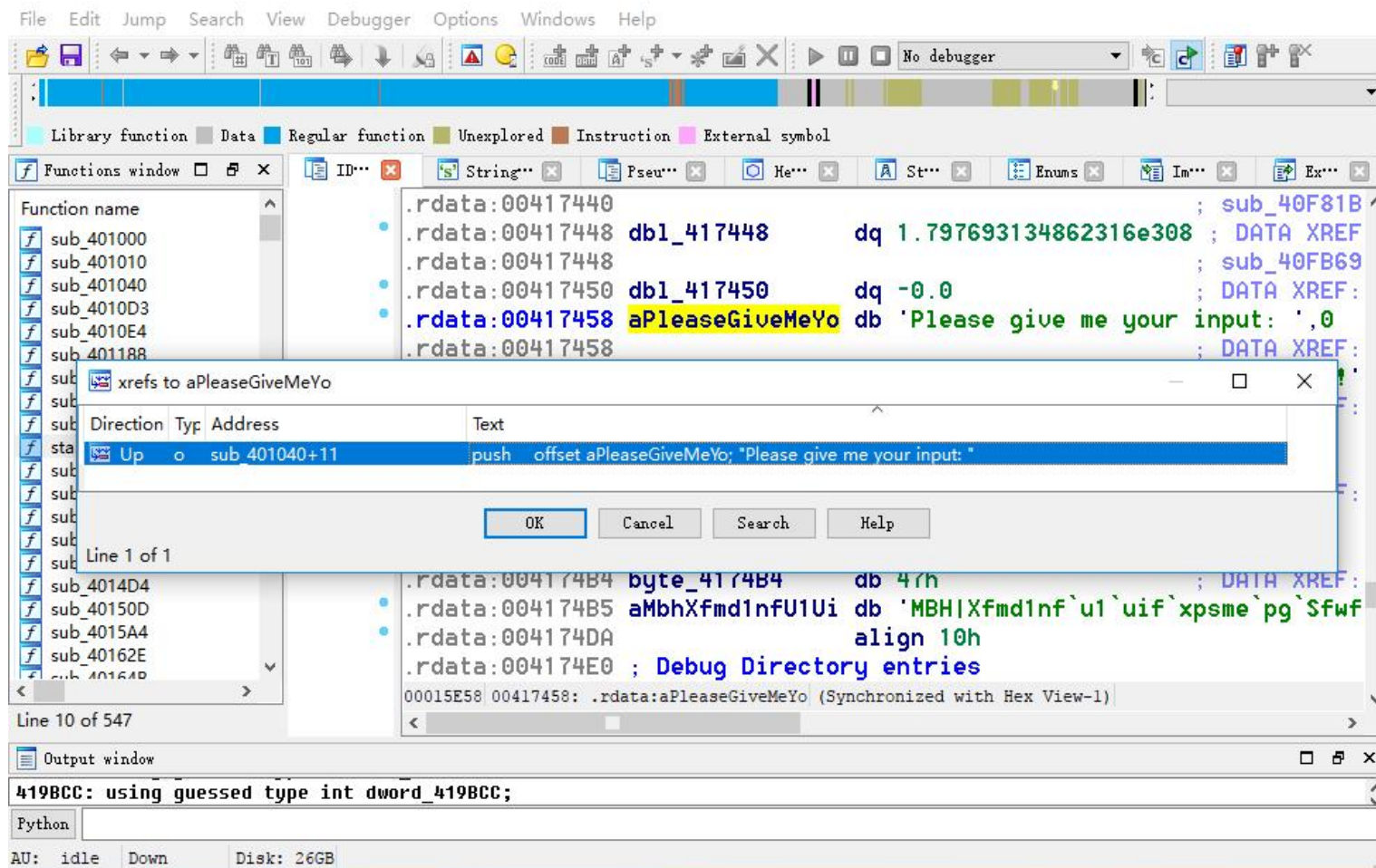
(3) 逆向牛刀小试 — 定位关键函数

- 可以看到跟程序功能有关的字符串，选中“**Please give me your input:**”并双击，来到该字符串的汇编代码窗口，并按快捷键“**x**”查看其交叉引用



第四章 Hello, world of reverse!

(3) 逆向牛刀小试 — 定位关键函数





第四章 Hello, world of reverse!

(3) 逆向牛刀小试 — 定位关键函数

□ 来到函数sub_401040的汇编代码

```
IDA View-A  Pseudocode-A  Strings window  Hex View-1  Structures
.text:00401040 sub_401040      proc near          ; CODE XREF: start-7B↓p
.text:00401040
.text:00401040 var_3C      = dword ptr -3Ch
.text:00401040 var_38      = byte ptr -38h
.text:00401040 var_4       = dword ptr -4
.text:00401040
.text:00401040 push      ebp
.text:00401041 mov       ebp, esp
.text:00401043 sub       esp, 3Ch
.text:00401046 mov       eax, __security_cookie
.text:00401048 xor       eax, ebp
.text:0040104D mov       [ebp+var_4], eax
.text:00401050 push      esi
.text:00401051 push      offset aPleaseGiveMeYo ; "Please give me your input: "
.text:00401056 call     sub_401010
.text:0040105B add       esp, 4
.text:0040105E xor       esi, esi
.text:00401060
```



第四章 Hello, world of reverse!

(3) 逆向牛刀小试 — 定位关键函数

□ 按快捷键“F5”
查看其伪代码，
便成功定位到
关键函数部分。

```
1 int sub_401040()  
2 {  
3     signed int v0; // esi@1  
4     char v1; // al@2  
5     int v2; // eax@5  
6     int v4; // [sp+4h] [bp-3Ch]@0  
7     char v5[52]; // [sp+8h] [bp-38h]@3  
8  
9     sub_401010("Please give me your input: ");  
10    v0 = 0;  
11    do  
12    {  
13        v1 = sub_403B40();  
14        if ( v1 == 10 )  
15            break;  
16        v5[v0++] = v1;  
17    }  
18    while ( v0 < 40 );  
19    v5[v0] = 0;  
20    if ( strlen(v5) == 37 )  
21    {  
22        v2 = 0;  
23        while ( v5[v2] + 1 == byte_4174B4[v2] )  
24        {  
25            if ( ++v2 >= 37 )  
26            {  
27                if ( v2 == 37 || v4 )  
28                {  
29                    sub_401010("Good! The flag is your input!!!\n");  
30                    return 0;  
31                }  
32                break;  
33            }  
34        }  
35    }  
36    sub_401010("Sorry! You are wrong!!!\n");  
37    return 0;  
38 }
```



第四章 Hello, world of reverse!

(3) 逆向牛刀小试

- 1. 定位关键函数
- 2. 程序逻辑分析



第四章 Hello, world of reverse!

(3) 逆向牛刀小试 — 程序逻辑分析

- 结合OllyDbg动态调试和代码，可以确定函数sub_401010为输出函数（printf），函数sub_403B4D为读取字符函数（getchar）。
- 程序一开始会获取用户的输入，当输入超过40个字符或者遇到换行（v1=10）就停止读取。
 - ‘\n’的ASCII码值为10：用鼠标选中10这个数字，然后按快捷键‘r’转换为字符\n

```
10  v0 = 0;  
11  do  
12  {  
13      v1 = sub_403B4D();  
14      if ( v1 == 10 )  
15          break;  
16      v5[v0++] = v1;  
17  }  
18  while ( v0 < 40 );
```

```
11  do  
12  {  
13      v1 = sub_403B4D();  
14      if ( v1 == '\n' )  
15          break;  
16      v5[v0++] = v1;  
17  }  
18  while ( v0 < 40 );
```




第四章 Hello, world of reverse!

(3) 逆向牛刀小试 — 程序逻辑分析

□ 如果需要程序输出 “Good! The flag is your input!!!”，就需要我们的输入满足两个条件：

– 1、输入字符串长度为37

– 2、将输入的每个字符ASCII码值加1后，与内存中的字符串byte_4174B4相等，双击byte_4174B4

```
19 v5[v0] = 0;
20 if ( strlen(v5) == 37 )
21 {
22     v2 = 0;
23     while ( v5[v2] + 1 == byte_4174B4[v2] )
24     {
25         if ( ++v2 >= 37 )
26         {
27             if ( v2 == 37 || v4 )
28             {
29                 sub_401010("Good! The flag is your input!!!\n");
30                 return 0;
31             }
32             break;
33         }
34     }
35 }
```



第四章 Hello, world of reverse!

(3) 逆向牛刀小试 — 程序逻辑分析

□ 通过关键代码分析，很容易找到比较的字符串地址为0x4174B4，其内容为

```
.rdata:004174B4 ; char byte_4174B4[] |  
.rdata:004174B4 byte_4174B4 db 47h ; DATA XREF: sub_401040+55↑r  
.rdata:004174B5 aMbH|Xfmd1nf`u1`uif`xpsme`pg`Sfwfstf~,0  
.rdata:004174B6
```

- 0x4174B4为47h ； 即G
- 0x4174B5为
“MBH|Xfmd1nf`u1`uif`xpsme`pg`Sfwfstf~”
- 故应该为：
“GMBH|Xfmd1nf`u1`uif`xpsme`pg`Sfwfstf~”



第四章 Hello, world of reverse!

(3) 逆向牛刀小试 — 程序逻辑分析

□ 将该字符串的每个字符ASCII码值减1即可得到正确的flag。

```
#include<stdio.h>
#include<string.h>
Int main() {
    char check[]="GMBH|Xfmdlnf`u1`uif`xpsme`pg`Sfwfstf~";
    char flag[40]={0};
    for(int i=0;i<strlen(check);i++){
        flag[i]= check[i]-1;
    }
    printf("%s\n", flag);
    return 0;
}
```



第四章 Hello, world of reverse!

(3) 逆向牛刀小试 — 程序逻辑分析

A screenshot of a Windows command prompt window. The title bar is blue and contains the text "C:\Documents and Settings\Administrator\桌面..." followed by standard window control buttons (minimize, maximize, close). The command prompt itself has a black background with white text. The first line of text is "FLAG>Welcome_t0_the_world_of_Reverse>". The second line of text is "Press any key to continue". The cursor is positioned at the end of the second line.

```
C:\Documents and Settings\Administrator\桌面...  
FLAG>Welcome_t0_the_world_of_Reverse>  
Press any key to continue
```



谢 谢!