



南開大學

NANKAI UNIVERSITY, P.R.CHINA 1919

允公允能 日新月异

汇编语言与逆向技术

第10章 动态逆向分析技术

王志

zwang@nankai.edu.cn

南开大学 网络空间安全学院

2024-2025学年



允公允能 日新月异

本章知识点

- 动态逆向分析
- x64dbg加载程序
- x64dbg的界面
- Memory Map
 - 难点：重定位
- 执行指令
 - 难点：执行到返回、执行到用户代码
- 断点 Breakpoint
 - 难点：软件断点、硬件断点、内存断点、条件断点、消息断点



南开大学
Nankai University



南開大學

NANKAI UNIVERSITY, P.R.CHINA 1919

允公允能 日新月异

动态逆向分析



允公允能 日新月异

动态逆向分析

- 动态分析技术中，最重要的是调试器
 - 用户模式的调试器
 - 内核模式的调试器



南开大学
Nankai University



允公允能 日新月异

用户模式调试器

- 调试用户模式的应用程序
- CPU **Ring3** 级
- x64dbg



南开大学
Nankai University



允公允能 日新月异

内核模式调试器

- 调试操作系统内核程序
- CPU **Ring0** 级
- WinDbg



南开大学
Nankai University



x64dbg

- x64dbg是一款具有可视化界面的用户模式调试器
- x64dbg结合了静态反汇编和动态调试
- x64dbg界面简洁优雅，显示的信息丰富而不杂乱
- x64dbg是最受欢迎的用户模式调适器之一
- x64dbg的外观与使用方法与经典的OllyDbg调试器基本一致
 - OllyDbg已经停止开发
 - x64dbg基本功能已经十分稳定，但仍时常有新功能加入



南开大学
Nankai University



x64dbg

- 可识别C和Windows编程中常用的函数
- 能将参数注释出来。
- 自动分析函数过程、循环语句、代码中的字符串
- 提供开放的调试接口
- 大量的插件可以选择



南开大学
Nankai University



x64dbg可以调试以下哪种级别的程序？

- A Ring0
- B Ring1
- C Ring2
- D Ring3

提交



南开大学
Nankai University



南開大學

NANKAI UNIVERSITY, P.R.CHINA 1919

允公允能 日新月异

x64dbg的下载和安装



下载与安装x64dbg

- 下载地址<https://snapshots.x64dbg.com/>
 - 通常下载最新版即可
 - 亦可根据课程安排统一下载指定版本
- 下载后得到一个压缩包，将其解压到任意位置
- 运行snapshot_xxxxxxxxxx\release文件夹内的x96dbg.exe
 - 在弹出的对话框中选择“是”
 - 该操作将x64dbg调试器注册为shell扩展
 - 这样在文件浏览器中右键单击一个.exe文件即可开始调试，无需预先打开x64dbg

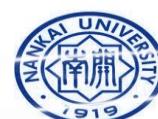


南开大学
Nankai University



x32dbg与x64dbg

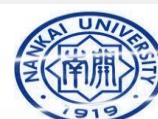
- 目前最新的Windows操作系统一般位64位系统，既可以运行64位程序，也可以运行32位程序
- x64dbg其实包含两个可执行文件
 - x32dbg.exe用于调试32位程序，它位于release\x32文件夹
 - x64dbg.exe用于调试64位程序，它位于release\x64文件夹
- 本课程主要学习32位程序，所以用到的是x32dbg.exe
 - 但习惯上一般还是称其为x64dbg
- 在文件管理器内右键单击一个.exe文件，然后选择“Debug with x64dbg”，x64dbg会自动根据该文件是32位还是64位选择恰当的x32dbg/x64dbg





初次配置x64dbg

- 打开x64dbg，在Options-Languages菜单中将语言设置为中文，并重新打开x64dbg
- 菜单-选项-选项-引擎，然后在弹出的对话框中勾选“禁用ASLR”
 - 禁用ASLR可以保证每次EXE被加载的地址一致





南開大學

NANKAI UNIVERSITY, P.R.CHINA 1919

允公允能 日新月异

x64dbg加载程序



x64dbg加载程序

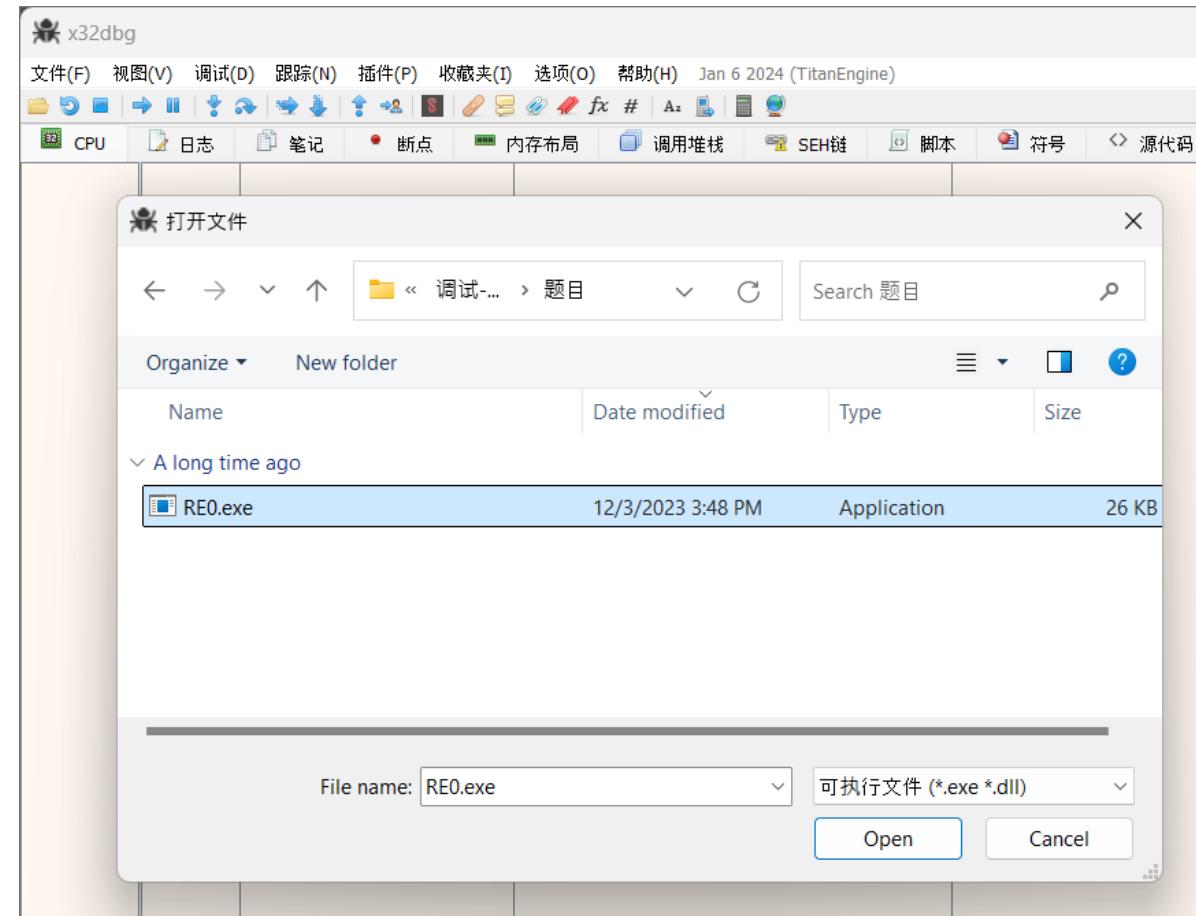
- 直接加载
 - EXE可执行文件
 - DLL动态链接库程序
- 如果程序已经在系统上运行，可以通过附加进程的方式调试
(Attach)



南开大学
Nankai University



直接加载程序



南开大学
Nankai University



允公允能 日新月异

打开一个可执行文件

- x64dbg会在软件PE头部提供的入口点（Entry Point）处中断。
- 如果想要调试main函数的代码，需要在main函数所在的地址添加一个断点并恢复执行



南开大学
Nankai University



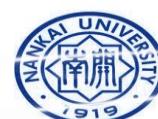
附加进程

附加

PID	名称	标题	路径
10392	smartscreen		C:\windows\System32\smartscre
6744	RuntimeBroker		C:\windows\System32\RuntimeB
9276	msedge		C:\Program Files (x86)\Micro
9448	msedge		C:\Program Files (x86)\Micro
3104	backgroundTaskHos		C:\Windows\System32\backgrou
20516	msedge		C:\Program Files (x86)\Micro
1176	msedge		C:\Program Files (x86)\Micro
17224	msedge		C:\Program Files (x86)\Micro
19168	msedge		C:\Program Files (x86)\Micro
7876	msedge	Chrome_widgetwin_0	C:\Program Files (x86)\Micro
18640	msedge	crashpad_sessionEndwatcher	C:\Program Files (x86)\Micro
14412	msedge	Copilot (preview)	C:\Program Files (x86)\Micro
4824	backgroundTaskHos		C:\Windows\System32\backgrou
13424	ai		C:\Program Files\Microsoft o
1560	UserOOBEBroker		C:\Windows\System32\oobe\Use
11304	svchost		C:\Windows\System32\svchost.
17852	SystemSettings	Settings	C:\Windows\ImmersiveControlP
14304	ShellExperienceHo		C:\Windows\SystemApps\ShellE
16656	WidgetService		C:\Program Files\Windows Apps
9348	ApplicationFrame	Settings	C:\Windows\System32\Appliac
5860	ai		C:\Program Files\Microsoft o
4636	WINWORD	x64dbg调试器入门.docx - Word	C:\Program Files\Microsoft o
10472	svchost		C:\Windows\System32\svchost

搜索: 在此输入可过滤结果...

[为什么进程 X 未显示于此?](#) 刷新 (F5) 查找窗口... 附加(A) 取消(C)



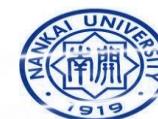
南开大学
Nankai University



允公允能 日新月异

附加进程

- x64dbg会立即暂停程序以及它所有的线程；
- 被附加的程序会暂停在Ntdll.dll的DbgBreakPoint处；



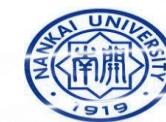
南开大学
Nankai University



允公允能 日新月异

附加进程

- 不希望调试Windows库，回到主代码最简单的方法就是在整个代码段中设置一个内存访问断点。
- 打开内存窗口，查看程序的内存空间，设置断点。



南开大学
Nankai University



允公允能 日新月异

内存访问断点

The screenshot shows the Immunity Debugger interface with a memory dump open. A context menu is displayed over a specific memory location. The menu path '内存断点(B)' is highlighted in blue. The menu options include:

- 访问 (Access)
- 读取 (Read)
- 写入 (Write)
- 执行 (Execute)
 - 一次性(S) F2
 - 重复设置(R)

The memory dump table has columns for Address, Size, Type, and Permissions.

地址	大小	方	页面信息	内容	类型	页面保护	初始保护
00400000	00001000	用户模块	debug2.exe		IMG	-R---	ERWC-
00401000	00003000	用户模块	".text"		IMG	ER---	ERWC-
00404000	00002000	用户模块	".rdata"		IMG	-R---	ERWC-
00406000	00001000	用户模块	".data"		IMG	-RW--	ERWC-
00407000	00001000	用户模块	".idata"		IMG	-R---	ERWC-
00408000	00001000	用户模块	".00cfg"		IMG	-R---	ERWC-
00409000	00001000	用户模块	".rsrc"		IMG	-R---	ERWC-
0040A000	00001000	用户模块	".reloc"		IMG	-R---	ERWC-
00770000	00011000	用户模块	\Device\Hardc		MAP	-R---	-R---
00790000	00010000	用户模块	保留		MAP	-RW--	-RW--
007A0000	0001F000	用户模块	保留		MAP	-R---	-R---
007C0000	00035000	用户模块	保留		PRV	-RW--	-RW--
007F5000	0000B000	用户模块	保留		PRV	-RW-G	-RW--
00800000	00017000	用户模块	保留		PRV	-RW--	-RW--
00817000	00012000	用户模块	PEB, TEB (14:		PRV	-RW--	-RW--
00829000	001D7000	用户模块	保留 (00800000)		PRV	-RW--	-RW--
00A00000	000FB000	用户模块	保留		PRV	-RW--	-RW--
00AFB000	00005000	用户模块	堆栈 (15440)		PRV	-RW-G	-RW--
00B00000	00004000	用户模块			MAP	-R---	-R---
00B10000	00001000	用户模块			MAP	-R---	-R---
00B20000	00002000	用户模块			PRV	-RW--	-RW--
00B30000	00011000	用户模块	\Device\Hardc		MAP	-R---	-R---
00B50000	00011000	用户模块	\Device\Hardc		MAP	-R---	-R---
00B70000	00003000	用户模块	\Device\Hardc		MAP	-R---	-R---
00B80000	00011000	用户模块	\Device\Hardc		MAP	-R---	-R---
00BA0000	00003000	用户模块	\Device\Hardc		MAP	-R---	-R---
00BB0000	00003000	用户模块	\Device\Hardc		MAP	-R---	-R---
00BC0000	00011000	用户模块	\Device\Hardc		MAP	-R---	-R---
00BE0000	00011000	用户模块	\Device\Hardc		MAP	-R---	-R---
00C00000	00002000	用户模块	保留		MAP	-R---	-R---
00C10000	00002000	用户模块	保留		MAP	-R---	-R---
00C20000	00001000	用户模块	保留		MAP	-R---	-R---
00C30000	00035000	用户模块	保留		MAP	-R---	-R---
00C65000	0000B000	用户模块	保留		MAP	-R---	-R---
00C70000	00008000	用户模块	保留 (00C70000)		MAP	-R---	-R---



南开大学
Nankai University



允公允能 日新月异

隐藏进程

- 系统的**隐藏进程**不能使用上述方法进行附加；
 - PC Hunter、GMER等工具获得隐藏进程的pid；
 - x64dbg的-p参数，通过隐藏进程的pid进行附加。



南开大学
Nankai University



允公允能 日新月异

重新装载

- Ctrl+F2可以重新装载当前调试的进程
- F2断点，记录之前的调试位置



南开大学
Nankai University



x64dbg可以调试以下哪种类型的程序？

- A exe程序
- B dll程序
- C 驱动程序
- D 正在执行的程序

提交



南开大学
Nankai University



南開大學

NANKAI UNIVERSITY, P.R.CHINA 1919

允公允能 日新月异

x64dbg的界面



x64dbg界面

反汇编
高亮显示的是下一条要执行的指令

寄存器

内存转储

栈

The screenshot displays the x64dbg debugger interface with the following sections:

- 反汇编 (Assembly):** Shows assembly code for the current thread (3312). A green box highlights the instruction at address 77A188B8, which is the next one to be executed.
- 寄存器 (Registers):** Displays the CPU register state. A green box highlights the EIP register containing the address 77A188B8.
- 内存转储 (Memory Dump):** Shows a dump of memory starting at address 77961000. A green box highlights the memory location at 77961000, which contains the value 80 E7 99 77.
- 栈 (Stack):** Shows the stack dump starting at address 009EF6CC. A green box highlights the stack entry at 009EF6CC, which contains the value 77A12FC1.

File menu: 文件(F) 被图(V) 调试(D) 跟踪(N) 插件(P) 收藏夹(I) 选项(O) 帮助(H) Jan 6 2024 (TitanEngine)

菜单栏: CPU 日志 签记 断点 内存布局 使用堆栈 SEH堆 脚本 符号 源代码 引用 线程 句柄 跟踪

状态栏: 已暂停 已到达系统点! 已调试时间: 0:12:59:57





反汇编窗口

- 反汇编面板窗口(Disassembler window)显示被调试程序的代码
 - 地址(Address)
 - 十六进制的机器码(Hex dump)
 - 反汇编代码(Disassembly)
 - 注释(Comment), 相关API参数



南开大学
Nankai University



反汇编窗口的双击操作

- Address列:显示相对地址，再次双击返回标准地址模式。
- Hex dump列:设置或取消无条件断点，对应的快捷键是“F2”键。

The screenshot shows two assembly windows from a debugger. In the left window, the EIP register is at address 77A188B8, indicated by a red arrow. The assembly code includes instructions like jmp ntdll.77A188C1, xor eax, eax, inc eax, and ret. In the right window, the EIP register is at address \$-9, indicated by a blue arrow. The assembly code is identical to the left window. Both windows show the same assembly code and memory dump.

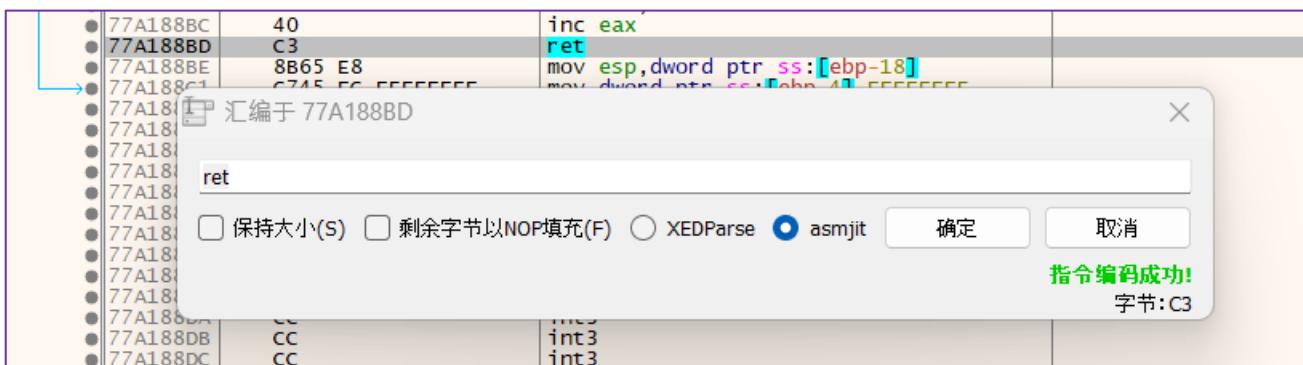
The screenshot shows a single assembly window from a debugger. The EIP register is at address //A188D3, indicated by a blue arrow. The assembly code includes instructions like jmp ntdll.//A188C1, xor eax, eax, inc eax, and ret. The memory dump column shows the instruction at address //A188D3 is currently being edited, as indicated by the red background.





反汇编窗口的双击操作

- Disassembly列:修改汇编代码，对应的快捷键是空格键。
- Comment列:增加或编辑注释，对应的快捷键是“;”键。





信息窗口

- 在进行动态跟踪时，信息窗口(Information window)显示与指令相关的各寄存器的值、API函数调用提示和跳转提示等信息。

The screenshot shows the OllyDbg debugger's Information window. The assembly pane displays the following code:

```
004014FE 8845 DC mov byte ptr ss:[ebp-24],al  
00401501 A1 30614000 mov eax,dword ptr ds:[<__scrt_current_>]  
00401506 33C9 xor ecx,ecx  
00401508 41 inc ecx  
00401509 3BC1 cmp eax,ecx  
0040150B 0F84 2F010000 je debug2.401640  
00401511 85C0 test eax,eax  
00401513 75 49 jne debug2.40155E  
00401515 89D0 30614000 mov dword ptr ds:<__scrt_current_native_startup_state>=0
```

The registers pane shows:

```
eax=0  
dword ptr ds:[00406130 <debug2.__scrt_current_native_startup_state>]=0
```

The stack pane shows:

```
.text:00401501 debug2.exe:$1501 #901
```

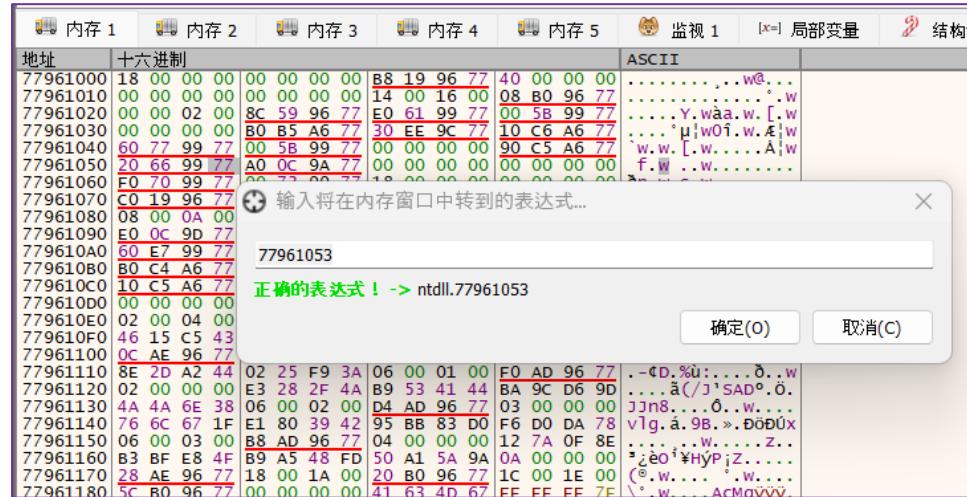




允公允能 日新月异

数据窗口

- 数据窗口 (Dump window)以十六进制和字符方式显示可执行文件内存中的数据
- 要显示指定内存地址的数据，可单击右键快捷菜单中的“Go to expression”命令或按“Ctrl+G”快捷键，打开地址窗口，输入地址



南开大学
Nankai University



允公允能 日新月异

数据窗口

地址	十六进制	ASCII
77961000	18 00 00 00	B8 19 96 77 40 00 00 00
77961010	00 00 00 00	00 00 00 00 14 00 16 00 08 B0 96 77
77961020	00 00 02 00	8C 59 96 77 E0 61 99 77 00 5B 99 77
77961030	00 00 00 00	B0 B5 A6 77 30 EE 9C 77 10 C6 A6 77
77961040	60 77 99 77	00 F8 00 77 00 00 00 00 00 00 00 00 00 00
77961050	20 66 99 77	A0 40
77961060	F0 70 99 77	E0
77961070	C0 19 96 77	40
77961080	08 00 0A 00	E0
77961090	E0 0C 9D 77	10
779610A0	60 E7 99 77	10
779610B0	B0 C4 A6 77	90
779610C0	10 C5 A6 77	90
779610D0	00 00 00 00	90
779610E0	02 00 04 00	DC
779610F0	46 15 C5 43	A5 FE 00 8D EE E3 D3 F0 06 00 00 00
77961100	0C AE 96 77	01 00 00 00 9A B8 13 35 96 5D BD 4F .@.w.....5.%o
77961110	8E 2D A2 44	02 25 F9 3A 06 00 01 00 F0 AD 96 77 .-¢.%.ù:....ð..w
77961120	02 00 00 00	E3 28 2F 4A B9 53 41 44 BA 9C D6 9D ...ä(/)SADº.Ö.
77961130	4A 4A 6E 38	06 00 02 00 D4 AD 96 77 03 00 00 00 jjn8....ð..w...
77961140	76 6C 67 1F	E1 80 39 42 95 BB 83 D0 F6 D0 DA 78 vlg.á.9B.».ĐÖĐÚX
77961150	06 00 03 00	B8 AD 96 77 04 00 00 00 12 7A 0F 8E,w.....z..
77961160	B3 BF E8 4F	B9 A5 48 FD 50 A1 5A 9A 0A 00 00 00 'èo i¥HýPjZ....
77961170	28 AE 96 77	18 00 1A 00 20 B0 96 77 1C 00 1E 00 (®,w....°.w....
77961180	5C B0 96 77	00 00 00 00 41 63 4D 67 FF FF FF 7F \°.w...AcMñVñVñ



南开大学
Nankai University



允公允能 日新月异

在数据窗口中跟踪

The screenshot shows a debugger's assembly window. The assembly code is as follows:

```
call debug2.40119A
push dword ptr ds:[eax+4]
push dword ptr ds:[eax]
call dword ptr ds:[<__stdio_common_vfprintf>]
add esp,18
ret
int3
int3
int3
int3
call debug2.4010B4
xor eax,eax
ret
int3
int3
call debug2.4011B3
call debug2.4010AA
push eax
call <debug2.__set_new_mode>
pop ecx
ret
```

A context menu is open over the assembly code, with the "在内存窗口中转到(F)" option highlighted. A tooltip shows the selected address: 常数(c): 004071D8 <debug2.__stdio_common_vfprintf> 值(v): [004071D8]. Other options in the menu include "二进制(B)", "复制(C)", "断点", "在反汇编中转到(F)", "在内存布局中转到", "流程图", "符号名称帮助", and "指令帮助".

The screenshot shows a memory dump window with tabs for 内存 1 to 内存 5 and 监视 1. The 内存 1 tab is active. The table has columns for 地址 (Address), 十六进制 (Hex), and ASCII. The first two rows of data are circled in red.

地址	十六进制	ASCII
0040149A	FF 15 D8 71 40 00 83 C4 18 C3 CC CC CC CC E8 07	ÿ. Øq@.. Ä. Äffffè.
004014AA	FC FF FF 33 C0 C3 CC CC E8 FC FC FF FF E8 EE FB	üÿ3AAítéüüÿèíü
004014BA	FF FF 50 E8 68 14 00 00 59 C3 CC CC CC CC E8 09	ÿÿPéh... YÄffffè.
004014CA	FD FF FF E9 02 00 00 00 CC CC 6A 14 68 08 57 40	ÿÿyé... fíj.h.w@
004014DA	00 E8 8E FB FF FF 6A 01 E8 81 FC FF FF 59 84 C0	.e. üÿj. è. üÿÿ. A
004014EA	0F 84 50 01 00 00 32 DB 88 5D E7 83 65 FC 00 E8	. P... 20.]ç. eü. è
004014FA	8D FC FF FF 88 45 DC A1 30 61 40 00 33 C9 41 3B	. üÿ. EÜ; Oa@. 3ÉA;
0040150A	C1 0F 84 2F 01 00 00 85 C0 75 49 89 0D 30 61 40	Á.../. . . Áui.. 0a@
0040151A	00 68 18 46 40 00 68 0C 43 40 00 E8 C4 13 00 00	. h. F@. h. c@. èÄ... .
0040152A	59 59 85 C0 74 11 C7 45 FC FE FF FF FF B8 FF 00	YY. At. ÇEÜþÿÿ. ý.
0040153A	00 00 E9 EF 00 00 00 68 08 42 40 00 68 00 40 40	. éi... h. B@. h. @@
0040154A	00 E8 98 13 00 00 59 59 C7 05 30 61 40 00 02 00	. è... YYç. Oa@...



南开大学
Nankai University



允公允能 日新月异

数据编辑

The screenshot shows the OllyDbg debugger interface with the data editor window open. The address bar at the top shows the current address as 004014CD and the value as E9 02000000. The assembly code pane displays the instruction: `jmp <debug2._scrt_common_main_seh>`. The data editor pane shows memory starting at address 0040149A. The memory dump table has columns for Address, Hex, ASCII, and Binary. The ASCII column contains Chinese characters and some English punctuation. A context menu is open over the memory dump table, with the "二进制编辑(I)" option highlighted. Other options include "编辑(E)", "填充(F)...", "复制(C)", "保存到文件", "修改(M)", "断点(B)", "搜索匹配特征(F)...", "与表达式同步(S)", "分配内存", and "转到(G)". The "EIP" register is also visible in the assembly pane.



南开大学
Nankai University



允公允能 日新月异

寄存器窗口

- 寄存器面板窗口(Registers window)显示CPU各寄存器的值，支持浮点、MMX寄存器。
- 红色用来高亮显示，上一条CPU指令执行之后，数值发生变化的寄存器



南开大学
Nankai University



允公允能 日新月异

寄存器窗口

- 通过双击一个寄存器值，修改该寄存器的数值

隐藏FPU		
EAX	00000000	
EBX	00726000	
ECX	00900000	"Actx "
EDX	00900000	"Actx "
EBP	008FFC7C	
ESP	008FFC48	
ESI	00401014	<debug2.OptionalHeader
EDI	00401014	<debug2.OptionalHeader
EIP	00401501	debug2.00401501



南开大学
Nankai University



允公允能 日新月异

栈窗口

- 栈窗口(Stack window)显示栈的内容，即ESP指向内存地址的内容。
- 栈中有函数的参数、局部变量、函数的返回地址等重要信息



南开大学
Nankai University



x64dbg界面有哪几个窗口

- A 反汇编窗口(Disassembler window)
- B 信息窗口(Information Window)
- C 寄存器窗口(Register Window)
- D 数据窗口(Dump Window)
- E 栈窗口 (Stack Window)

提交



南开大学
Nankai University



x64dbg的栈窗口可以看到下面哪些信息

- A 函数返回地址
- B 函数的参数
- C 函数的局部变量
- D 全局变量

提交



南开大学
Nankai University



x64dbg中可以修改以下哪些内容？

- A 内存数据
- B CPU寄存器
- C 栈上的数据
- D CPU指令

提交



南开大学
Nankai University



南開大學

NANKAI UNIVERSITY, P.R.CHINA 1919

允公允能 日新月異

Memory Map



允公允能 日新月异

Memory Map

地址	大小	方	页面信息	内容	类型	页面保护	初始保护
00400000	00001000	用户模块	debug2.exe		IMG	-R---	ERWC-
00401000	00003000	用户模块	".text"		IMG	ER---	ERWC-
00404000	00002000	用户模块	".rdata"		IMG	-R---	ERWC-
00406000	00001000	用户模块	".data"		IMG	-RW--	ERWC-
00407000	00001000	用户模块	".idata"		IMG	-R---	ERWC-
00408000	00001000	用户模块	".0cfg"		IMG	-R---	ERWC-
00409000	00001000	用户模块	".rsrc"		IMG	-R---	ERWC-
0040A000	00001000	用户模块	".reloc"		IMG	-R---	ERWC-
00570000	00011000	用户模块	\Device\HarddiskVolume3\windows\		MAP	-R---	-R---
00590000	00010000	用户模块			MAP	-RW--	-RW--
005A0000	0001F000	用户模块			MAP	-R---	-R---
005C0000	00035000	用户模块	保留		PRV	-RW-	-RW-
005F5000	0000B000	用户模块			PRV	-RW-G	-RW-
00600000	00125000	用户模块	保留		PRV	-RW-	-RW-
00725000	00016000	用户模块	PEB, TEB (14660), wow64 TEB (14660)		PRV	-RW--	-RW--
0073B000	000C5000	用户模块	保留 (00600000)		PRV	-RW--	-RW--
00800000	000FB000	用户模块	保留		PRV	-RW--	-RW--
008FB000	00005000	用户模块	堆栈 (14660)		PRV	-RW-G	-RW--
00900000	00004000	用户模块			MAP	-R---	-R---
00910000	00001000	用户模块			MAP	-R---	-R---
00920000	00002000	用户模块			PRV	-RW--	-RW--
00930000	00011000	用户模块	\Device\HarddiskVolume3\windows\		MAP	-R---	-R---
00950000	00011000	用户模块	\Device\HarddiskVolume3\windows\		MAP	-R---	-R---
00970000	00003000	用户模块	\Device\HarddiskVolume3\windows\		MAP	-R---	-R---
00980000	00011000	用户模块	\Device\HarddiskVolume3\windows\		MAP	-R---	-R---
009A0000	00003000	用户模块	\Device\HarddiskVolume3\windows\		MAP	-R---	-R---
009B0000	00003000	用户模块	\Device\HarddiskVolume3\windows\		MAP	-R---	-R---
009C0000	000CE000	用户模块	\Device\HarddiskVolume3\windows\		MAP	-R---	-R---
00A90000	00008000	用户模块	保留 (00A90000)		PRV	-RW--	-RW--
00AA0000	00011000	用户模块	\Device\HarddiskVolume3\windows\		MAP	-R---	-R---
00AC0000	00011000	用户模块	\Device\HarddiskVolume3\windows\		MAP	-R---	-R---
00AE0000	00002000	用户模块	保留		MAP	-R---	-R---
00AF0000	00002000	用户模块			MAP	-R---	-R---
00B00000	00001000	用户模块			MAP	-R---	-R---
00B10000	00035000	用户模块	保留		PRV	-RW-	-RW-
00B45000	00008000	用户模块			PRV	-RW-G	-RW--
00B50000	00035000	用户模块	保留		PRV	-RW-	-RW--
00B85000	0000B000	用户模块	保留		PRV	-RW-G	-RW--



南开大学
Nankai University



Memory Map

- 内存中的exe和dll程序的位置被x64dbg识别
- 双击有可执行属性的行，会跳转到反汇编窗口
- 双击没有可执行属性的行，会跳转到数据窗口



南开大学
Nankai University



允公允能 日新月异

重定位 (Rebasing)

- 重定位发生在PE文件装载时，没有装载到指定的Image Base地址时
- PE文件头中Image Base
 - exe文件会装载到Image Base
 - 大部分exe的Image Base是0x00400000
 - **ASLR** (Address Space Layout Randomization)



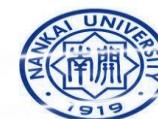
南开大学
Nankai University



允公允能 日新月异

dll重定位

- dll经常发生重定位
 - 一个应用程序会装载多个dll
 - Windows的dll文件通常会使用不同的Image Base
 - kernel32.dll **6B800000**, user32.dll **69E00000**
 - 第三方的dll通常都是用相同的Image Base



南开大学
Nankai University



绝对地址对相对地址

Example 10-1. Assembly code that requires relocation

```
00401203      mov eax, [ebp+var_8]
00401206      cmp [ebp+var_4], 0
0040120a      jnz loc_0040120
0040120c      1mov eax, dword_40CF60
```

- 相对地址（Relative Address）不受重定向影响
- 绝对地址（Absolute Address）受影响



选出图中相对地址?

- A [ebp + var_8]
- B [ebp + var_4]
- C jnz loc_0040120
- D dword_40CF60

Example 10-1. Assembly code that requires relocation

```
00401203    mov eax, [ebp+var_8]
00401206    cmp [ebp+var_4], 0
0040120a    jnz loc_0040120
0040120c    1mov eax, dword_40CF60
```

提交



南开大学
Nankai University



允公允能 日新月异

修改绝对地址

- 大部分dll文件，都有一个.reloc节，用于重定位发生时的修复
- dll文件的装载顺序是**不固定的**
- 重定位的位置也是**不可预测的**



南开大学
Nankai University



允公允能 日新月异

.reloc节

2020/2/26 11:34 应用程序扩展 119 KB

PEiD v0.95

File: C:\Windows\System32\user32.dll

Entrypoint: 000414D0 EP Section: .text >

File Offset: 000408D0 First Bytes: 8B,FF,55,8B >

Linker Info: 14.13 Subsystem: Win32 GUI >

Nothing found [Overlay] *

Multi Scan Task Viewer Options About Exit

Stay on top ?? ->

Section Viewer

Name	V. Offset	V. Size	R. Offset	R. Size	Flags
.text	00001000	000A2DE3	00000400	000A2E00	60000020
.data	000A4000	000013CA	000A3200	00000800	C0000040
.idata	000A6000	000083F0	000A3A00	00008400	40000040
.didat	000AF000	00000078	000ABE00	00000200	C0000040
.rsrc	000B0000	000E11A0	000AC000	000E1200	40000040
.reloc	00192000	00006048	0018D200	00006200	42000040

Close



南开大学
Nankai University



dll重定位

- 如果删除dll的.reloc节
 - dll不能进行重定位
 - 必须装载到PE文件头指定的Image Base地址
- 重定位会影响程序的装载时间
 - Good programmer specifies **non-default** base image for dlls。



南开大学
Nankai University



Binary Ninja 和 x64dbg

- 重定位
 - Binary Ninja静态反汇编是会将可执行文件加载到其默认地址
 - x64dbg的动态调试，是重定位之后的结果
 - Binary Ninja 和x64dbg的反汇编地址有可能**不一致**
 - 可以在Binary Ninja中进行rebase以匹配调试过程中的实际加载地址
 - 或者可以在x64dbg中关闭ASLR



南开大学
Nankai University



线程和栈

- Memory Map窗口，除了exe和dll，还有栈的信息
- 每个线程都有一个自己私有的栈

地址	大小	方	页面信息	内容	类型	页面保护	初始保护
00400000	00001000	用户模块	debug2.exe		IMG	-R---	ERWC-
00401000	00003000	用户模块	".text"		IMG	ER---	ERWC-
00404000	00002000	用户模块	".rdata"		IMG	-R---	ERWC-
00406000	00001000	用户模块	".data"		IMG	-RW--	ERWC-
00407000	00001000	用户模块	".idata"		IMG	-R---	ERWC-
00408000	00001000	用户模块	".00cfg"		IMG	-R---	ERWC-
00409000	00001000	用户模块	".rsrc"		IMG	-R---	ERWC-
0040A000	00001000	用户模块	".reloc"		IMG	-R---	ERWC-
00570000	00011000	用户模块	\Device\HarddiskVolume3\windows\!		MAP	-R---	-R---
00590000	00010000	用户模块			MAP	-RW--	-RW--
005A0000	0001F000	用户模块			MAP	-R---	-R---
005C0000	00035000	用户模块	保留		PRV		-RW--
005F5000	0000B000	用户模块			PRV	-RW-G	-RW--
00600000	00125000	用户模块	保留		PRV		-RW--
00725000	00016000	用户模块	PEB, TEB (14660), Wow64 TEB (14660)		PRV	-RW--	-RW--
0073B000	000C5000	用户模块	保留 (00600000)		PRV		-RW--
00800000	000FB000	用户模块	保留		PRV		-RW--
008FB000	00005000	用户模块	堆栈 (14660)		PRV	-RW-G	-RW--
00900000	00004000	用户模块			MAP	-R---	-R---
00910000	00001000	用户模块			MAP	-R---	-R---
00920000	00002000	用户模块			PRV	-RW--	-RW--





如何解决和避免dll的重定位问题？

- A 使用不同的Image Base地址
- B 使用编译器默认的Image Base地址
- C .reloc节记录需要修改的信息
- D 指定dll的装载顺序

提交



南开大学
Nankai University



以下哪些是线程私有的？

- A 栈
- B 寄存器
- C 内存空间
- D 代码

提交



南开大学
Nankai University



南開大學

NANKAI UNIVERSITY, P.R.CHINA 1919

允公允能 日新月异

执行指令



允公允能 日新月异

Function	Menu	Hotkey	Button
Run/Play	Debug ▶ Run	F9	▶
Pause	Debug ▶ Pause	F12	⏸
Run to selection	Breakpoint ▶ Run to Selection	F4	
Run until return	Debug ▶ Execute till Return	CTRL-F9	➡
Run until user code	Debug ▶ Execute till User Code	ALT-F9	
Single-step/step-into	Debug ▶ Step Into	F7	◀▶
Step-over	Debug ▶ Step Over	F8	◀



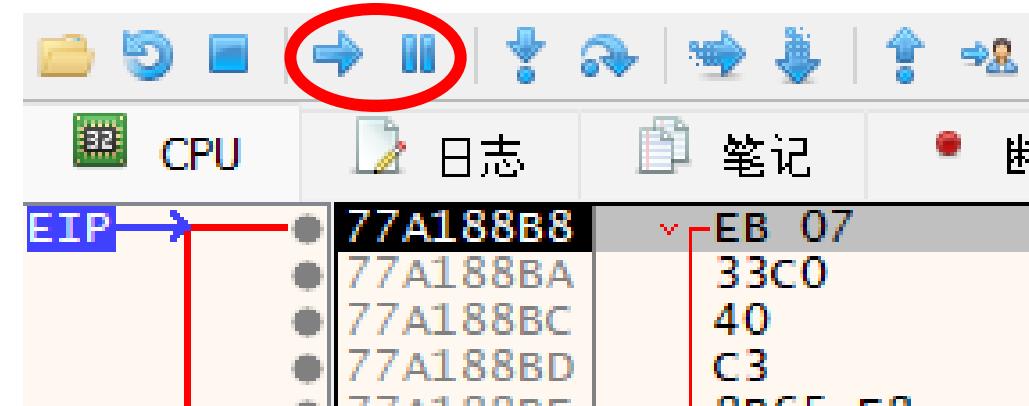
南開大學
Nankai University



允公允能 日新月异

运行和暂停

- 运行调试应用程序 (F9)
- 暂停调试程序 (F12)

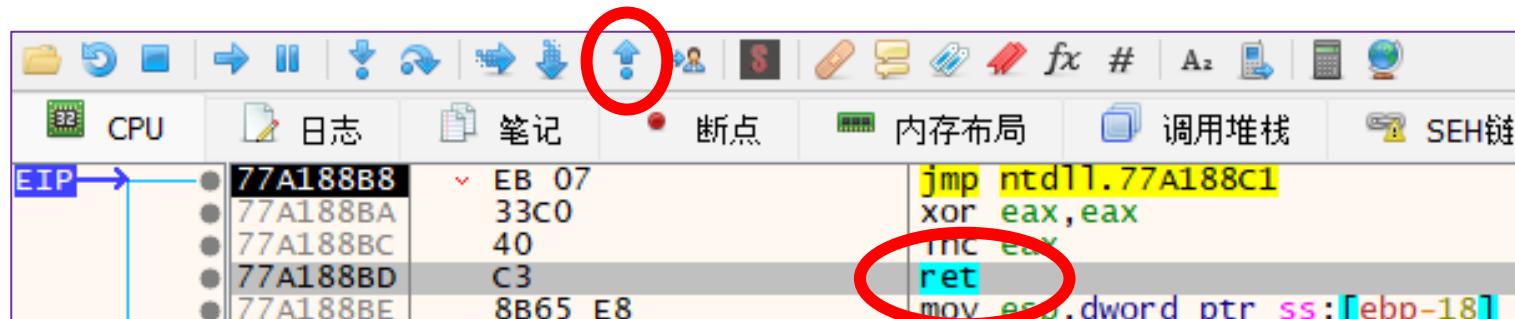


南开大学
Nankai University



执行直到返回

- “执行直到返回” 当前函数执行到返回指令之前的指令。
- 结束当前调试的函数





允公允能 日新月异

执行到用户代码

- “执行到用户代码”会从库函数中返回到程序的代码节，例如.text节的代码

The screenshot shows the Immunity Debugger interface. The CPU tab is selected, displaying assembly code. The instruction at address 0040172D is highlighted with a red circle, and the EIP register also points to this address. The assembly code for this instruction is `jmp <debug2.__scrt_common_main_seh>`. The assembly window shows the following code:

地址	汇编指令
00401728	E8 A9FAFFFF
0040172D	E9 A2FDFFFF
0040172E	CC
00401733	CC
00401734	55
00401735	8BEC
00401737	8B45 08
0040173A	56
0040173B	8B48 3C
0040173E	03C8
00401740	0FB741 14
00401744	8D51 18
00401747	02D0

The assembly code continues with:

```
call debug2.4011D6
jmp <debug2.__scrt_common_main_seh>
int3
int3
push ebp
mov ebp,esp
mov eax,dword ptr ss:[ebp+8]
push esi
mov ecx,dword ptr ds:[eax+3C]
add ecx,ecx
movzx eax,word ptr ds:[ecx+14]
lea edx,dword ptr ds:[ecx+18]
add edx,edx
```

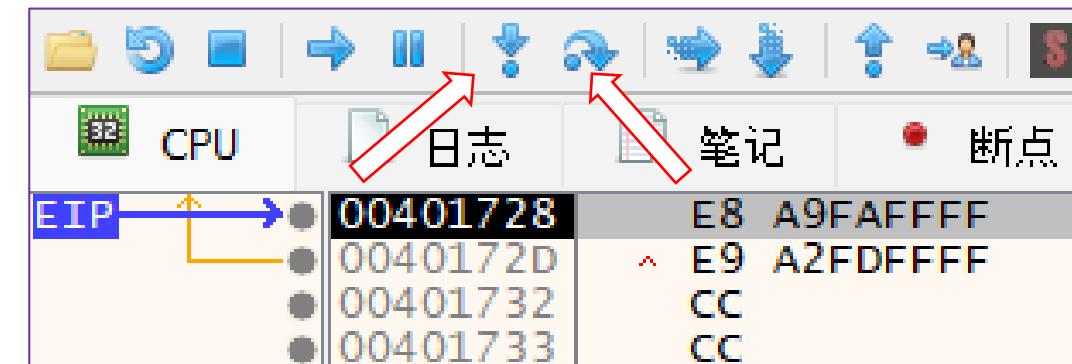


南开大学
Nankai University



按步执行

- 步入 (F7) , 也叫做单步执行
- 步过 (F8) , 对于函数, 可以使用步过, 执行完整个函数后暂停。



南开大学
Nankai University



x64dbg支持哪些代码执行操作？

- A 执行和暂停
- B 执行直到返回
- C 执行到用户代码
- D 按步执行

提交



南开大学
Nankai University

允公允能 日新月异

x64dbg调试中，如果想快速执行完当前函数，需要选择哪种执行方式？

- A 执行和暂停
- B 执行直到返回**
- C 执行到用户代码
- D 按步执行

 提交

南开大学
Nankai University

允公允能 日新月异

x64dbg调试中，如果想快速从dll库代码空间回到用户代码空间，需要选择哪种执行方式？

- A 执行和暂停
- B 执行直到返回
- C 执行到用户代码
- D 按步执行

提交



南开大学
Nankai University



南開大學

NANKAI UNIVERSITY, P.R.CHINA 1919

允公允能 日新月异

断点



断点 (Breakpoint)

- 断点 (Breakpoint) 是调试器的重要功能，使执行的程序中断在指定的地方，从而方便对其进行分析。
 - 查看间接跳转的位置
 - 查看函数的参数
 - 快速执行循环结构体





允公允能 日新月异

断点类型

- 软件断点
- 硬件断点
- 内存断点
- 消息断点
- 条件断点



南开大学
Nankai University



断点窗口

类型	地址	模块/标签/异常	状态	反汇编	命中	摘要
软件断点	762E7460	<kernel32.dll.ExitProcess>	已启用	push ebp	0	



南开大学
Nankai University



允公允能 日新月异

保存断点

- 关闭x64dbg之后， x64dbg会自动保存调试时设置的断点信息。
- 再次打开x64dbg调试相同的程序， 断点信息还在。

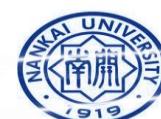


南开大学
Nankai University



软件断点

- 在x64dbg中可以使用bp命令或者“F2”快捷键来设置/消断点。
 - INT 3指令替换原始CPU指令，操作码是**0xCC**
 - INT 3指令产生异常，停止程序执行
 - x64dbg捕获异常
 - 恢复原始的CPU指令



南开大学
Nankai University



软件断点

- 优点
 - 可以设置无数个断点
- 缺点
 - 修改了原始程序的机器码
 - 容易被软件检测到
 - 检测API函数入口点是否为0xCC



南开大学
Nankai University



软件断点

- 一些软件会检测API的首地址是否为0xCC来检测INT 3断点。方法是取得检测函数的地址，然后读取它的第1个字节，判断它是否等于“CC”

```
FARPROC Uaddr ;
BYTE Mark = 0;
(FARPROC&) Uaddr =GetProcAddress ( LoadLibrary("user32.dll"), "MessageBoxA");
Mark = *((BYTE*)Uaddr);           //取 MessageBoxA 函数的第一个字节
if(Mark ==0xCC)                  //如果该字节为“CC”，则认为 MessageBoxA 函数被下断了
    return TRUE                   //发现断点
```





允公允能 日新月异

软件断点

- 将软件断点设在函数内部或末尾，例如将断点设在函数入口的下一行
- 使用硬件断点



南开大学
Nankai University



软件断点同时可以设置多少个？

A

1

B

2

C

4

D

任意多个

提交



南开大学
Nankai University



硬件断点

- 不需要将CPU指令首字节修改为"CC"
- 基于CPU中断**DRx**寄存器设置
- 硬件断点分为三类：
 - 硬件执行断点
 - 硬件读取断点
 - 硬件写入断点



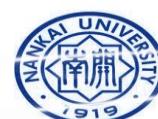
南开大学
Nankai University



允公允能 日新月异

硬件执行断点

- 实现方式
 - 将断点的内存地址存储到CPU的DRx寄存器
 - CPU执行过程中会判断下一条指令的地址与DRx寄存器中的地址是否一致
 - 如果一致，CPU产生异常，中止程序执行，将控制权转移给调试器

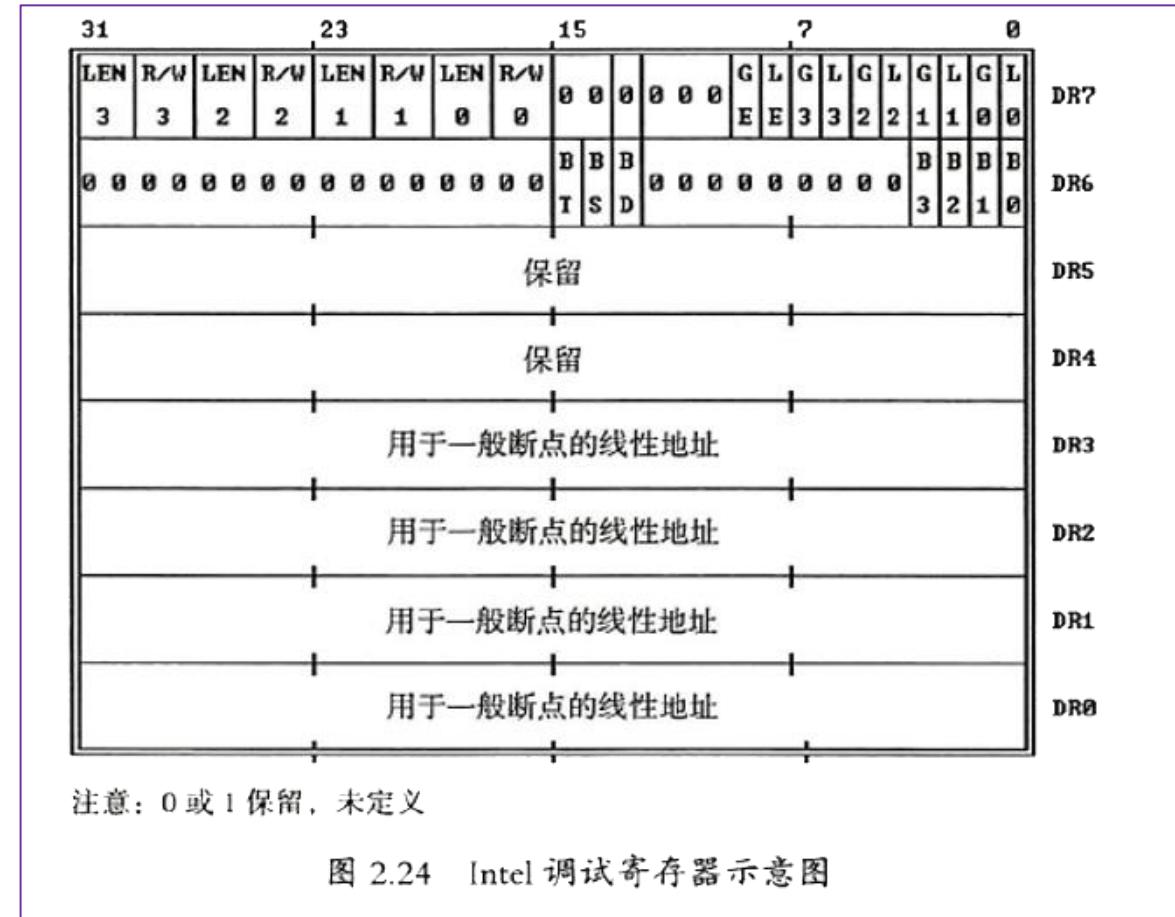


南开大学
Nankai University



允公允能 日新月异

硬件执行断点



南开大学
Nankai University



允公允能 日新月异

硬件执行断点

- 优点
 - 隐蔽性强，不修改CPU指令，更难检测到
 - 速度快，在CPU内部进行判断
- 缺点
 - 只有4个CPU寄存器存储断点地址，DR0到DR3
 - 所有硬件断点共享4个DRx寄存器，包括后面介绍的硬件读取/写入断点



南开大学
Nankai University



允公允能 日新月异

硬件执行断点设置

- 在指定的代码行单击右键，执行快捷菜单中的“断点”→“硬件断点”
- 在命令行中，输入命令“he 断点地址”



南开大学
Nankai University



硬件执行断点

The screenshot shows a debugger interface with assembly code and a context menu. The assembly code is as follows:

```
push 1
call debug2.401168
pop ecx
test al,al
je debug2.
xor b1,b1
mov byte p
and dword
call debug
mov byte p
mov eax,dw
xor ecx,ec
inc ecx
cmp eax,ec
je debug2.
test eax,e
jne debug2.
```

The context menu is open over the instruction `je debug2.`. The menu items include:

- 二进制(B)
- 复制(C)
- 断点
- 在内存窗口中转到(F)
- 在内存布局中转到
- 流程图
- 指令帮助
- 显示指令提示

Sub-menu for '断点':

- 设置条件断点 Shift+F2
- 切换 F2
- 设置硬件断点(执行)

File names visible in the background: exe_common.inl:237, exe_common.inl:240, exe_common.inl:249





允公允能 日新月异

查看和管理硬件断点

- 单击菜单项“查看”→“硬件断点”，打开硬件断点面板
- 单击“Delete”按钮删除相应的硬件断点



南开大学
Nankai University



允公允能 日新月异

硬件写入/读取断点

- 硬件断点除了可以在特定指令被执行时触发，还可以在特定字节被读取时触发
- 可以用于发现内存中的数据何时被修改
- 这样的断点我们成为**硬件写入断点**或者**硬件读取断点**



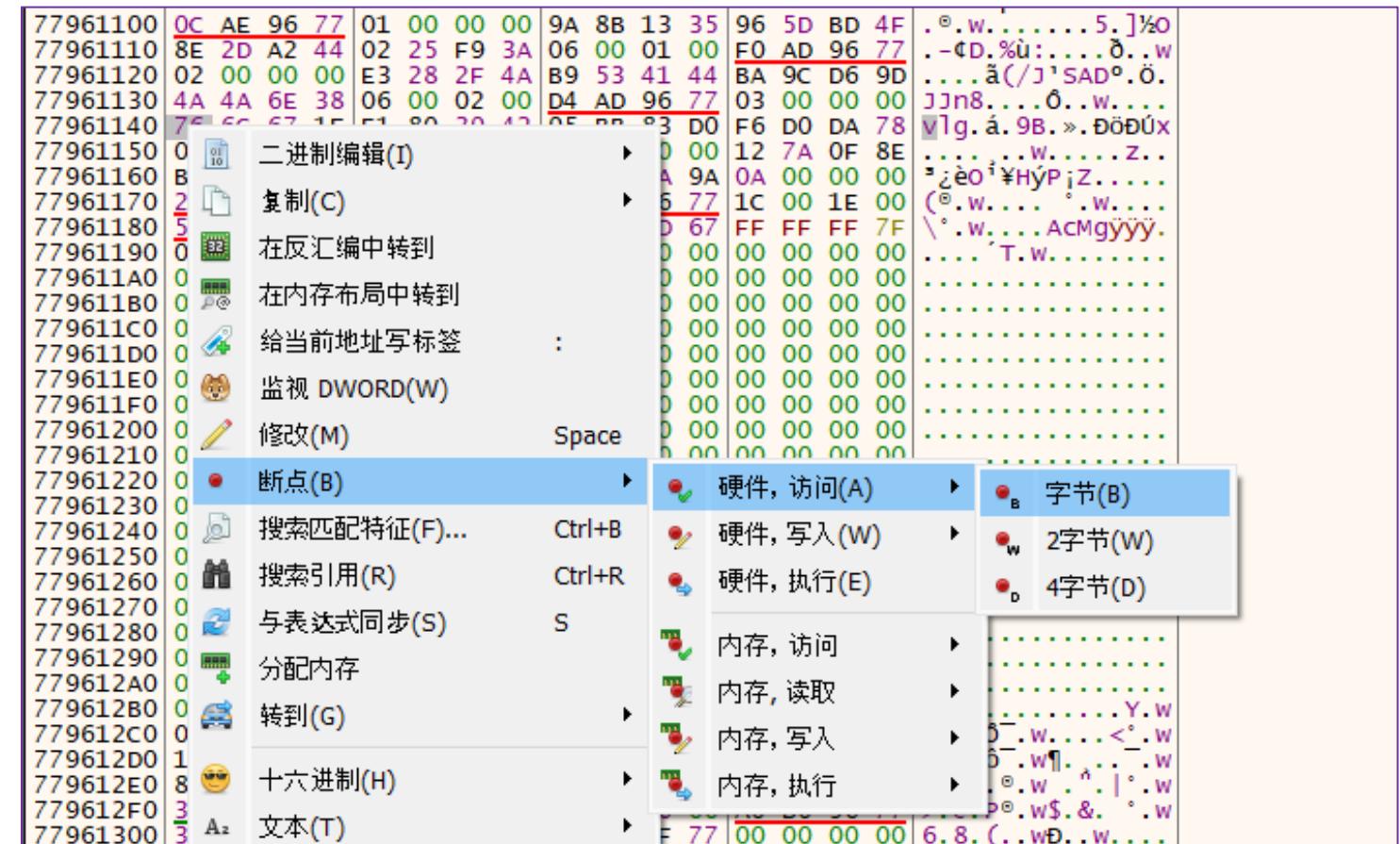
南开大学
Nankai University



允公允能 日新月异

设置硬件读取/写入断点

- 硬件写入/读取断点可以设置触发条件为读取/写入/执行
- 硬件写入/读取断点可以设置大小位1/2/4字节



南开大学
Nankai University



硬件断点同时最多可以设置几个？

- A 1
- B 2
- C 4
- D 任意多个

提交



南开大学
Nankai University



允公允能 日新月异

内存断点

- 内存页的属性

- 可读 readable
- 可写 writable
- 可执行 executable

```
mov    dword ptr  [405528], edx      ;对[405528]处的内存进行写入  
mov    dword ptr  edx,[405528]       ;对[405528]处的内存进行读取
```



南开大学
Nankai University



允公允能 日新月异

内存断点原理

- 对所设的内存地址页设置不可访问或者不可写属性
- 当被调试程序访问或者写入内存页时，产生异常，中断执行
- CPU将控制权交给x64dbg



南开大学
Nankai University



允公允能 日新月异

内存断点原理

- x64dbg比较异常地址是不是断点地址
- 如果是断点地址，就继续中断程序执行
- 如果不是断点地址，恢复程序的执行



南开大学
Nankai University



允公允能 日新月异

设置内存断点

The screenshot shows a memory dump window with columns for Address (地址), Hex (十六进制), ASCII (ASCII), and Binary (二进制). A context menu is open over the memory dump area, specifically at address 77961040. The menu path is:

- 断点(B) (Breakpoint)
- 硬件, 访问(A) (Hardware, Access)
- 硬件, 写入(W) (Hardware, Write)
- 硬件, 执行(E) (Hardware, Execute)
- 内存, 访问 (Memory, Access)
- 一次性(S) (One-time)
- 重复设置(R) (Repeat)

The "一次性(S)" option is highlighted in blue.



南开大学
Nankai University



内存断点

- 程序中断在访问内存的CPU指令
 - 还没有访问或者写入内存
- 硬件断点，程序中断在访问内存CPU指令的下一条指令
 - 内存的访问或者写操作已经完成



南开大学
Nankai University



允公允能 日新月异

内存断点

- 内存中的数据可以设置内存断点，内存中的**指令**也可以设置内存断点
 - CPU指令存储在内存中
 - 当CPU指令被执行的时候，CPU会访问内存读取下一条指令
 - 触发内存访问异常



南开大学
Nankai University



允公允能 日新月异

内存断点

- 优点：
 - 不需要修改CPU指令
 - 当INT 3断点和硬件断点无效的时候可以使用内存断点



南开大学
Nankai University



允公允能 日新月异

内存断点

- 缺点：
 - 执行速度慢
 - 每次出现内存异常时，都要通过比较来确定是否应该中断程序执行
 - 内存断点会降低x64dbg的执行速度
 - x64dbg同一时间，同一内存页上，只能设置**1**个内存断点



南开大学
Nankai University



内存断点

地址	大小	方	页面信息	内容	类型	页面保护	初始保护
00400000	00001000	用户模块	debug2.exe		IMG	-R---	ERWC-
00401000	00003000	用户模块	".text"		IMG	ER--G	ERWC-
00404000	00002000	用户模块	".rdata"		IMG	-R---	ERWC-
00406000	00001000	用户模块	".data"		IMG	-RW--	ERWC-
00407000	00001000	用户模块	".idata"		IMG	-R---	ERWC-
00408000	00001000	用户模块	".00cfg"		IMG	-R---	ERWC-
00409000	00001000	用户模块	".rsrc"		IMG	-R---	ERWC-
0040A000	00001000	用户模块	".reloc"		IMG	-R---	ERWC-
00480000	00011000	用户模块	\Device\HarddiskVolume3\windows\\$		MAP	-R---	-R---
004A0000	00010000	用户模块			MAP	-RW--	-RW--
004B0000	0001F000	用户模块			MAP	-R---	-R---
004D0000	00035000	用户模块	保留		PRV	-RW--	-RW--
00505000	0000B000	用户模块			PRV	-RW-G	-RW--

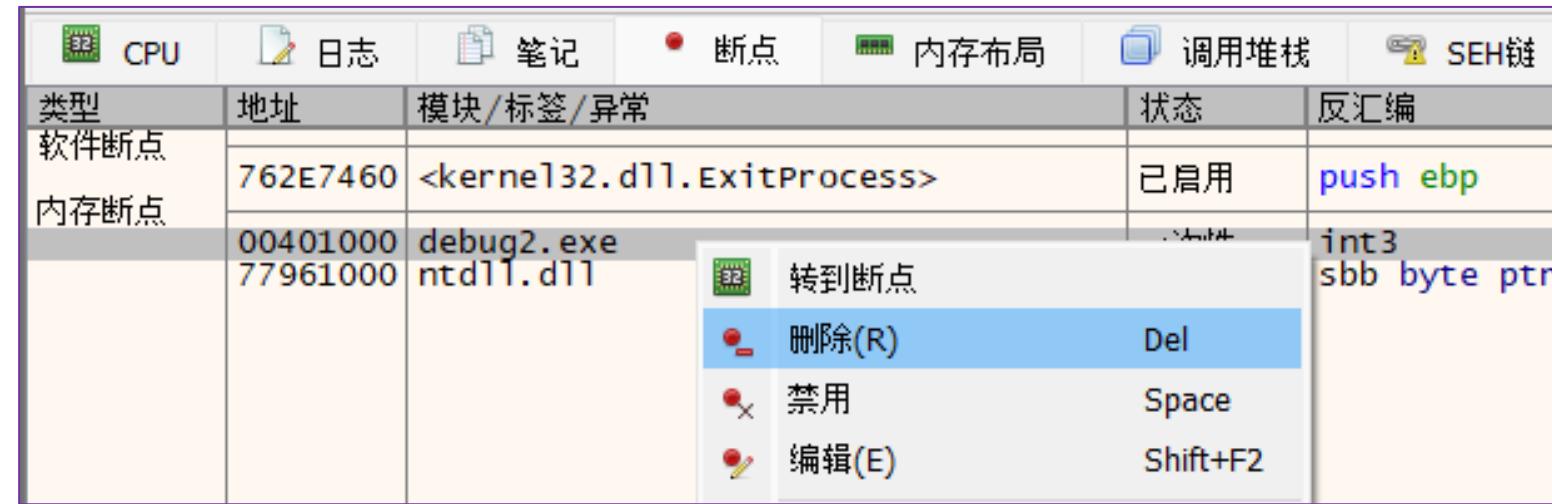


南开大学
Nankai University



允公允能 日新月异

内存断点的删除



南开大学
Nankai University



x64dbg支持哪些内存访问行为的断点

- A 内存读断点
- B 内存写断点
- C 内存执行断点
- D 内存读、写、执行组合的断点

提交



南开大学
Nankai University



对于同一地址，内存断点同时可以设置几个？

- A 1
- B 2
- C 4
- D 任意多个

提交



南开大学
Nankai University



允公允能 日新月异

消息断点

- Windows是由消息驱动的
- 消息断点：当某个特定窗口函数接收到某个特定消息时，消息断点将使程序中断



南开大学
Nankai University



允公允能 日新月异

消息断点

- 消息断点与INT 3断点的区别
 - INT 3断点可以在程序启动之前设置
 - 消息断点只有在窗口被创建之后才能被设置并拦截消息。



南开大学
Nankai University



Windows消息

- 当用户单击一个按钮、移动光标时，都会产生消息发送给当前窗体。
- Windows消息有4个参数
 - 1个窗口句柄(hwnd)
 - 1个消息编号(msg)
 - 2个32位长(long)的参数



南开大学
Nankai University



Windows消息

- Windows通过句柄来标识它所代表的对象。
- 在单击某个按钮时，Windows通过句柄来判断单击了哪一个按钮，然后发送相应的消息来通知程序



南开大学
Nankai University



Windows消息

- 在句柄页签中，单击右键-刷新，即可查看被调试进程所有窗口的信息
- 因为更新该信息会占用较多资源，所以x64dbg不会自动刷新，需要手动刷新才能看到最新的信息



南开大学
Nankai University



Windows消息

窗口过程	句柄	标题	窗口类名	线程	风格	扩展风格	父	大小	启用
7595B460	1050A44	CrackMe1 by crackinglessons.com	#32770	主线程 (19)	94C800CC	10101	0	(0,0); 375x264	已启用
75980980	70D42	Please enter the serial Key:	Static	主线程 (19)	50020000	4	1050A44	(81, 88); 173x16	已启用
759252F0	270BB6	Default IME	IME	主线程 (19)	8C000000	0	1050A44	(0,0); 0x0	已禁用
7594A1B0	C0DD2	Check	Button	主线程 (19)	50010001	4	1050A44	(81, 171); 94x29	已启用
7594A1B0	50D0E	About	Button	主线程 (19)	50010000	4	1050A44	(193, 171); 94x29	已启用
75961120	50CE8		Edit	主线程 (19)	50010080	204	1050A44	(80, 124); 199x29	已启用
769F72E0	50BE8	MSCTFIME UI	MSCTFIME UI	主线程 (19)	8C000000	0	270BB6	(0,0); 0x0	已禁用





允公允能 日新月异

消息断点

右键单击Check按钮，在菜单中选择“消息断点”

窗口过程	句柄	标题	窗口类名	线程
7595B460	1050A44	CrackMe1 by crackinglessons.com	#32770	主线程
75980980	70D42	Please enter the Serial Key:	Static	主线程
759252F0	270BB6	Default IME	IME	主线程
7594A1B0	C0DD2	Check	button	主线程
7594A1B0	50D0E	About		主线程
75961120	50CE8			主线程
769F72E0	50BE8	MSCTIME UI		主线程

A right-click context menu is displayed over the "Check" button in the window titled "Check". The menu items are:

- 刷新(R) F5
- 禁用窗口
- 在反汇编中转到窗口过程 Return
- Follow in Threads
- 在窗口过程切换断点 F2
- 消息断点 (highlighted in blue)
- 搜索...
- 复制(C)

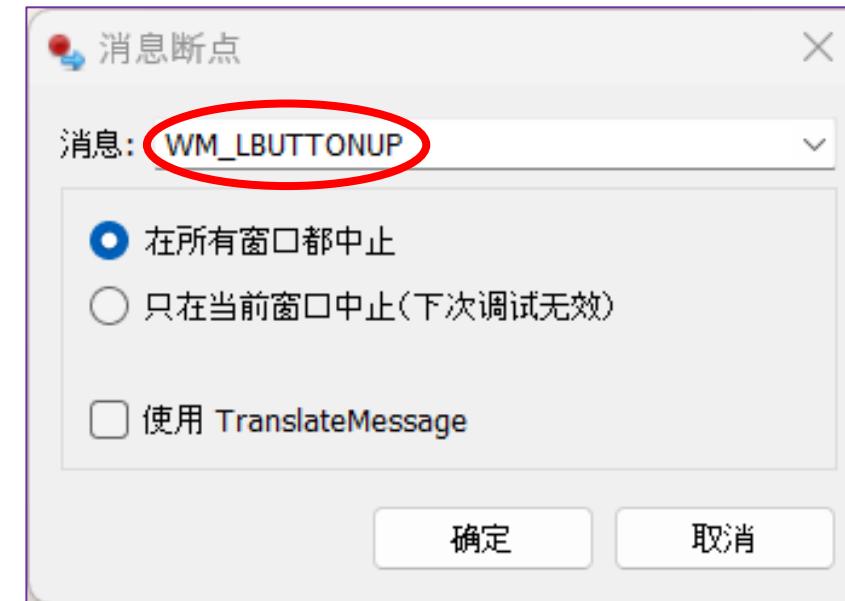


南开大学
Nankai University



消息断点

- 在弹出的消息窗口中，输入“**WM_COMMAND**”消息。

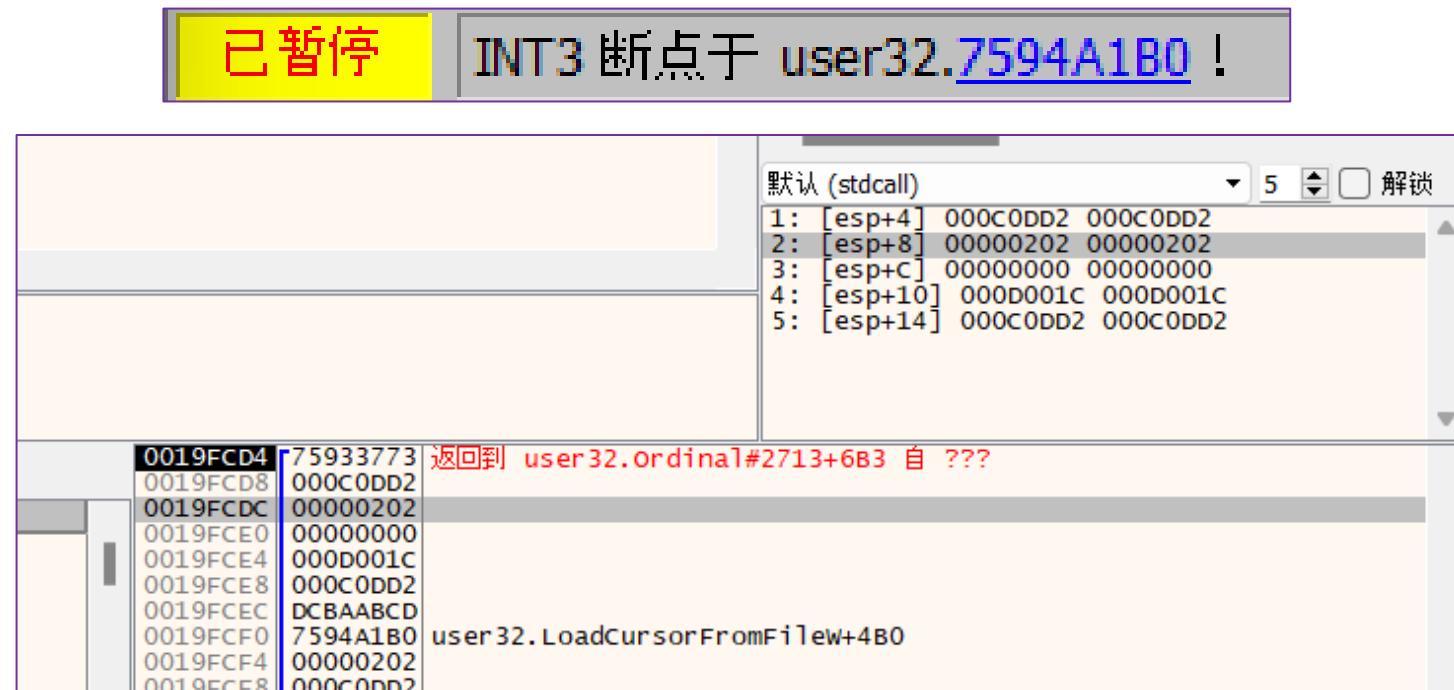




允公允能 日新月异

消息断点

- 点击Check按钮， 程序中断在Windows系统代码



南开大学
Nankai University



允公允能 日新月异

消息断点

- 打开内存窗口，对crackme1的.text段设置内存访问断点

内存断点(B)

- 访问
- 读取
- 写入
- 执行
 - 一次性(S) F2
 - 重复设置(R)



南开大学
Nankai University



允公允能 日新月异

消息断点

- F9运行程序，中断在0x00401090，crackme1的消息处理循环的位置

已暂停 内存断点(执行)位于[crackme1.00401000](#), 异常地址:[crackme1.00401090](#)!

EIP	ECX	ESP	操作码	汇编指令
00401090			55	push ebp
00401091			8BEC	mov ebp,esp
00401093			83EC 30	sub esp,30
00401096			8D45 D0	lea eax,dword ptr ss:[ebp-30]
00401099			6A 30	push 30
0040109B			6A 00	push 0
0040109D			50	push eax
0040109E			E8 2D0E0000	call crackme1.401ED0
004010A3			83C4 0C	add esp,c
004010A6			817D 0C 11010000	cmp dword ptr ss:[ebp+C],111
004010AD			0F85 C3000000	jne crackme1.401176
004010B3			0FB745 10	movzx eax,word ptr ss:[ebp+10]
004010B7			83E8 02	sub eax,2
004010BA			0F84 AE000000	je crackme1.40116E
004010C0			2D E7030000	sub eax,3E7
004010C5			74 23	je crackme1.4010EA
004010C7			83E8 01	sub eax,1
004010CA			0F85 A6000000	jne crackme1.401176



南开大学
Nankai University



消息断点

- 浏览消息循环的代码，发现crackme1使用GetDlgItemTextA读取用户输入的字符串，并与一个常量字符串比较，然后根据结果弹出不同的MessageBoxA

004010EF	50	push eax	
004010F0	68 E8030000	push 3E8	
004010F5	FF35 A0424100	push dword ptr ds:[4142A0]	
004010FB	FF15 10D14000	call dword ptr ds:[<GetDlgItemTextA>]	411AD8:"cr4ckingL3ssons"
00401101	B9 D81A4100	mov ecx,crackme1.411AD8	
00401106	8D45 D0	lea eax,dword ptr ss:[ebp-30]	
00401109	0F1F80 00000000	nop dword ptr ds:[eax],eax	
00401110	8A10	mov d1,byte ptr ds:[eax]	
00401112	3A11	cmp d1,byte ptr ds:[ecx]	
00401114	75 1A	jne crackme1.401130	
00401116	84D2	test d1,d1	
00401118	74 12	je crackme1.40112C	
0040111A	8A50 01	mov d1,byte ptr ds:[eax+1]	
0040111D	3A51 01	cmp d1,byte ptr ds:[ecx+1]	
00401120	75 0E	jne crackme1.401130	
00401122	83C0 02	add eax,2	
00401125	83C1 02	add ecx,2	
00401128	84D2	test d1,d1	
0040112A	75 E4	jne crackme1.401110	
0040112C	33C0	xor eax,eax	
0040112E	EB 05	jmp crackme1.401135	
00401130	1BC0	sbb eax,eax	
00401132	83C8 01	or eax,1	
00401135	6A 00	push 0	
00401137	85C0	test eax,eax	
00401139	75 19	jne crackme1.401154	
0040113B	68 E81A4100	push crackme1.411AE8	411AE8:"Congrats!"
00401140	68 F41A4100	push crackme1.411AF4	411AF4:"Well done!"
00401145	50	push eax	
00401146	FF15 18D14000	call dword ptr ds:[<MessageBoxA>]	
0040114C	33C0	xor eax,eax	
0040114E	8BE5	mov esp,ebp	
00401150	5D	pop ebp	
00401151	C2 1000	ret 10	
00401154	68 001B4100	push crackme1.411B00	411B00:"Sorry"
00401159	68 081B4100	push crackme1.411B08	411B08:"Wrong serial key. Try again."
0040115E	6A 00	push 0	
00401160	FF15 18D14000	call dword ptr ds:[<MessageBoxA>]	
00401166	33C0	xor eax,eax	
00401168	8BE5	mov esp,ebp	
0040116A	5D	pop ebp	





允公允能 日新月异

条件断点

- 在调试过程中，在满足一定条件时断点才会触发，这类断点称为
条件断点

x64dbg的条件断点可以按**寄存器、内存、消息**等设断点



南开大学
Nankai University



允公允能 日新月异

条件断点

- 条件断点是带有条件表达式的INT 3断点
 - 在INT3断点的基础上添加一个条件表达式
 - INT3断点中断程序执行
 - x64dbg判断表达式的值，如果是非零的值，则继续中断程序
 - 如果是0，恢复程序继续执行



南开大学
Nankai University



允公允能 日新月异

条件断点

The screenshot shows a debugger interface with a toolbar at the top and a table below it. The table has columns for Type, Address, Module/Label/Exception, Status, and Assembly. A specific row is selected, showing an address of 0040172D, module debug2.exe, and assembly code starting with jmp. A red circle highlights the 'Breakpoint Conditions' section of the 'Edit Breakpoint' dialog box, which is overlaid on the table. The dialog box contains fields for Log Text, Log Condition, Command, Command Condition, Name, and Count (set to 0). It also includes checkboxes for Once, No Log, and Fast Recovery, along with Save and Cancel buttons.

类型	地址	模块/标签/异常	状态	反汇编
软件断点	0040172D 762E7460	debug2.exe <kernel32.dll.ExitProcess>	已启用 已启用	jmp <debug2.__scrt_common_main_se push ebp

● 编辑断点 debug2.0040172D

暂停条件(B):

日志文本(L):

日志条件(G):

命令(C):

命令条件(O):

名称(N):

执行次数(H): 0

一次性(O) 不输出日志 快速恢复(F) 保存(S) 取消(A)



南开大学
Nankai University



允公允能 日新月异

条件断点

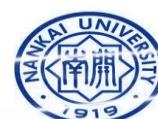
- 命令行设置条件断点

如果安装了命令行插件，可以直接输入如下代码。

```
bp CreateFileA, [STRING [esp+4]]=="c:\\1212.txt"
```

如果是 Unicode 字符串，可以输入如下命令。

```
bp CreateFileW, [UNICODE [esp+4]]=="C:\\1212.txt"
```



南开大学
Nankai University



允公允能 日新月异

条件断点

- 除了具有条件断点的功能
- 还能记录断点处函数表达式或参数的值
- 可以设置断点的次数，每次符合条件的中断，记数器的值就减一



南开大学
Nankai University



x64dbg支持的断点类型？

- A 软件断点
- B 硬件断点
- C 条件断点
- D 内存访问断点

提交



南开大学
Nankai University



下面哪些断点会使程序执行速度变慢？

- A 软件断点
- B 硬件断点
- C 条件断点
- D 内存访问断点

提交



南开大学
Nankai University



南開大學

NANKAI UNIVERSITY, P.R.CHINA 1919

允公允能 日新月异

汇编语言与逆向技术

第10章 动态逆向分析技术

王志

zwang@nankai.edu.cn

南开大学 网络空间安全学院

2024-2025学年



允公允能 日新月异

本章知识点

- 动态逆向分析
- x64dbg加载程序
- x64dbg的界面
- Memory Map
 - 难点：重定位
- 执行指令
 - 难点：执行到返回、执行到用户代码
- 断点 Breakpoint
 - 难点：软件断点、硬件断点、内存断点、条件断点、消息断点



南开大学
Nankai University