

 安全 专栏收录该内容

 34 篇文章 0 订阅

 订阅专栏

/*

消息断点偶尔总是不如意, 留档学习学习。

*/

标题:【原创】011yDBG入门系列(五)一消息断点及 RUN 跟踪

作 者: CCDebuger 时 间: 2006-02-19,16:02:46 链 接: http://bbs.pediy.com/showthread.php?t=21532

011yDBG 入门系列(五)一消息断点及 RUN 跟踪

作者: CCDebuger

找了几十个不同语言编写的 crackme,发现只用消息断点的话有很多并不能真正到达我们要找的关键位置,想想还是把消息断 点和 RUN 跟踪结合在一起讲,更有效一点。关于消息断点的更多内容大家可以参考 jingulong 兄的那篇《几种典型程序 Button处理代码的定位》的文章,堪称经典之作。今天仍然选择 crackmes.cjb.net 镜像打包中的一个名称 为 cycle 的 crackme。按照惯例,我们先运行一下这个程序看看:

🗖 CycleCr 🔀						
Name	CCDebuger					
Serial	78787878					
Check About						

我们输入用户名 CCDebuger,序列号 78787878,点上面那个"Check"按钮,呵, 没反应!看来是要注册码正确才有动静。 现在关掉这个 crackme,用 PEiD 查一下壳,原来是 MASM32 / TASM32 [Overlay]。启动 011yDBG 载入这个程序,F9让它运 行。这个程序按我们前面讲的采用字串参考或函数参考的方法都很容易断下来。但我们今天主要学习的是消息断点及 RUN 跟 踪,就先用消息断点来断这个程序吧。在设消息断点前,有两个内容我们要简单了解一下:首先我们要了解的是消息。 Windows 的中文翻译就是"窗口",而 Windows 上面的应用程序也都是通过窗口来与用户交互的。现在就有一个问题,应用 程序是如何知道用户作了什么样的操作的?这里就要用到消息了。Windows 是个基于消息的系统,它在应用程序开始执行后, 为该程序创建一个"消息队列",用来存放该程序可能创建的各种不同窗口的信息。比如你创建窗口、点击按钮、移动鼠标等 等,都是通过消息来完成的。通俗的说,Windows 就像一个中间人,你要干什么事是先通知它,然后它才通过传递消息的方式 通知应用程序作出相应的操作。说到这,又有个问题了,在 Windows 下有多个程序都在运行,那我点了某个按钮,或把某个 窗口最大化,Windows 知道我是点的哪个吗?这里就要说到另一个内容:句柄(handle)了。句柄一般是个 32 位的数,表示 一个对象。Windows 通过使用句柄来标识它代表的对象。比如你点击某个按钮,Windows 就是通过句柄来判断你是点击了那一 个按钮,然后发送相应的消息通知程序。说完这些我们再回到我们调试的程序上来,你应该已经用 011yDBG 把这 个 crackme 载入并按 F9 键运行了吧?现在我们输入用户名"CCDebuger",序列号"78787878",先不要点那 个"Check"按钮,我们来到 011yDBG 中,点击菜单 查看->窗口(或者点击工具栏上那个"W"的图标),我们会看到以下内 容:

國窗口								X
句柄	标题	父窗口	WinProc	ID	风格	扩展风格	线程	^
0006049A 00040492 00040494 00040496 000505498 00050524 00050542 00050542 00050542 00050542 00050542 00050542 00050553A 000070412	CycleCrackme Name Serial About <u>About</u> <u>Check</u> Default IME M 在选中的句柄上 (灰	Topmost 0006049A 0006049A 0006049A 0006049A 0006049A 0006049A 0006049A 0006049A 同新 眼鏡 ClassH	roc	000003E8 0000FFFF 0000FFFF 00000067 00000003	94C80A44 50010080 50020000 50020000 50010000 50010000 50010000 0080 008	00010100 00000204 00000004 00000004 00000004 00000004 000000		
句枘,程序9 次重新载入8 都会不同	➡ 色显示)按右键选择 寸 图中这个菜单项 ➤	在 ClassPro 在 ClassPro 在 ClassPro	oc 上切换 oc 上设条) oc 上设消;	新点 件记录断点 ^{息断点})	į			*
		复制到剪贴树 排序按 界面选项	Q	.0	+ + +			

我们在选中的条目上点右键,再选择上图所示的菜单项,会来到下面这个窗口:

在 ▼inProc 上设置断	点			
消息: 任何消息				Ī
 中断在任意窗口 中断在全部相同标题 仅中断在当前窗口 	圆的窗□ (下一时] 才间无效)	点这 我们	(个按钮选择)需要的消息
	从不	按消息	条件满足	次数(十进制)
暂停程序:	0	\odot	0.	
记录 WinProc 参数:	\odot	0		
□ 根据名称排列消息			确定	取消

现在我们点击图上的那个下拉菜单,呵,原来里面的消息真不少。这么多消息我们选哪个呢? 注册是个按钮,我们就在按下按 钮再松开时让程序中断。查一下 MSDN,我们知道这个消息应该是 WM_LBUTTON_UP,看字面意思也可以知道是左键松开时的消息:

在 VinPro	c 上设置断点 🛛 🗙
消息:	任何消息
 中断 	1A1 LB_GETITEMHEIGHT 1A2 LB_FINDSTRINGEXACT
中断	IAS LB_SETIDUCALE IAS LB_GETLOCALE IAT LB SETCOUNT
〇仅中国	1A8 LB_INITSTORAGE 1A9 LB_ITEMFROMPOINT
	200 WM_MOUSEMOVE 近中这个的古技左键
	203 WM_LBUTTONDBLCLK 204 WM_RBUTTONDOWN
记录 Wir	Proc 参数: • • •
🔲 根据名称	R排列消息 确定 取消

从下拉菜单中选中那个 202 WM_LBUTTON_UP,再按确定按钮,我们的消息断点就设好了。现在我们还要做一件事,就是 把 RUN 跟踪打开。有人可能要问,这个 RUN 跟踪是干什么的?简单的说,RUN 跟踪就是把被调试程序执行过的指令保存下 来,让你可以查看被调试程序运行期间干了哪些事。RUN 跟踪会把地址、寄存器的内容、消息以及已知的操作数记录 到 RUN 跟踪缓冲区中,你可以通过查看 RUN 跟踪的记录来了解程序执行了那些指令。在这还要注意一个缓冲区大小的问题, 如果执行的指令太多,缓冲区满了的话,就会自动丢弃前面老的记录。我们可以在调试选项->跟踪中设置:

🗄 调试说	项															
命令	反汇编	CPU TT /th	寄存器	堆栈	分析 1	分析 2	分析 3									
安全 RUN]]]] []]]]]]]]]]]]]]]	→ 调试 跟踪的缓冲 記录命令 記示 ESP 記示标志位 <u>急是跟踪步う</u> 司步 CPU 及 丸行到 RET 充计时将相邻	● 事件 区大小(字 立系统 DIL 立事操作命号 正NM 跟踪 后,单步步 部的命令分)	异常 节/记录)	 1 M / 60 上这个,[系统 DLL	SFX <u>K</u> 授沖 拉打 五为我们· 不感兴趣	● 字■ ● ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○	」地址 点下 段置									
					确定	撤消	取消									
现在我们 击就是清 右键,在 <u>RUN</u> 跟 此处为新	回到 01 除 RUN 弾出菜車 ^余	1yDBG 中 跟踪的记 全中选择	,点击到 录,对 RUN 跟跟 +Gray *		->打开 宗熟悉时 所有函 择部分 致过程	或清除 R 还可以读 数过程的	RUN 跟踪 及置条件) 入口:	(第 ⁻	一次点 保证当	点这个菜 当前在到	奖单是打 我们调i	打开 R 式的程	UN 跟踪 序领空,	,在打 在反	丁开的情 汇编窗	§况下点 口中点击

·添加函数过程中的分支(B) 转到 ۲ 添加所有函数过程的入口(2) 数据窗口中跟随 选这个 Ctrl+K 查看调用树 跟踪时忽略选定的部分(K) 设置条件(C) Ctrl+T 查找(S) ۲ 我们可以看到 011yDBG 把识别出的函数过程都在前面加了灰色条: 地址 HEX 数据 00401000 C\$ 6A 00 反汇编 注释
 6A 00
 PUSH 0

 E8 A4020000
 CALL
 (IMF. &KERNEL32. GetModuleHandleA)

 A3 94214000
 MOV DWORD PTR DS: [402194], EAX

 6A 00
 PUSH 0
 pModule = NULL GetModuleHandleA 00401002 00401007 rlParam = NVLL 00401000 6A 00 PUSH 0 68 29104000 PUSH cycle.00401029 0A 0 灰色条显示 0 6A 6 0 数过程 DWORD PTR DS:[0 DlgProc = cycle.00401029 hOwner = NVLL 0040100E 00401013 pTemplate = 68 hInst = 00400000 00401015 . 6A 6i - 568 FF35 凶数过程 : DWORD PTR DS:[402194] E8 87020000 - CALL

 6A 00
 PUSH 0

 . 68 7C020000
 CALL

 . 68 7C020000
 CALL
 00401017 DialogBoxParamA ExitCode = 0 ExitProcess 0040101I 00401022 00401024 现在我们回到那个 crackme 中按那个 "Check" 按钮, 被 011yDBG 断下了: HEX 数据 反汇编 地址 注释 8BFF MOV EDI, EDI 55、粉红色表示 PUSH EBP 7D3B010

77D3B011 8BE是条件断点 77D3B013 8B4

这时我们点击菜单查看->内存,或者点击工具栏上那个 "M" 按钮(也可以按组合键 ALT+M),来到内存映射窗口:

🛯 内存映	谢								
地址 🗇	大小	属主	区段	包含		类型 访问	可 🔼		
003D0000 0 003E0000 0 003F0000 0 00400000 0	00001000 00001000 00004000	cycle	CODE	<u>PE 文</u> 件	:头	Priv RW Map RW Priv RW Imag R			
00401000 00402000 00403000 00404001左 00405001 00410001的	00001000 00001000 键选中4 I代码段	cycle cycle cycle 01000	CODE DATA .idata .reloc .rsrc	代码 数新 五 定 位 源	刷新在加新	TTTT 所 反汇编窗口 CPV 数据 居	コ中查看 窗口中查	Enter 著	
004右击在	弾出菜	单中选这	条,或直		查	L 戈		Ctrl+H	3
008接在40 008接在40 0094000000)1000的1)0010000	代码段上	按F2键		在议	方问上设置	呈中断 	F2	
					设i 设i 设i	置内存访问 置内存写 <i>)</i> 置访问	割断点(Δ) ∖断点(Ψ)	V	•
					复制 排所 界面	制到剪贴机 茅按 面选项	Ę		* * *

为什么在这里设访问断点,我也说一下。我们可以看一下常见的 PE 文件,没加过壳的用 PEiD 检测是这样:



看完上面的图我们应该了解为什么在 401000 处的代码段下访问断点了,我们这里的意思就是在消息断点断下后,只要 按 F9 键运行时执行到程序代码段的指令我们就中断,这样就可以回到程序领空了(当然在 401000 处所在的段不是绝对的, 我们主要是要看程序的代码段在什么位置,其实在上面图中 011yDBG 内存窗口的"包含"栏中我们就可以看得很清楚了)。 设好访问断点后我们按 F9 键,被 011yDBG 断下:

地址	HEX	数据	反汇编	注释
00401022		6A 00	PUSH 0	ExitCode = 0
00401024	L,	E8 7C020000	CALL <tmp. &kernel32,="" exitprocess=""></tmp.>	ExitProcess
00401029	r .	C8 000000	ENTER 0.0	
0040102D		53	PUSH EBX	
0040102E		57	PUSH EDI 下红 下断在这里	
0040102F		56	PUSH_ESI	
00401030		817D OC 10010	[CMP_DWORD_PTR_SS:[EBP+C],110	
00401037		74 25	JE SHORT cycle.0040105E	
00401039		817D OC 11010	CMP_DWORD_PTR_SS:[EBP+C],111	
00401040		74 40	JE SHORT cycle.00401082	
00401042		837D OC 10	CMP_DWORD_PTR_SS:[EBP+C], 10	
00401046	. ~	74 OF	JE SHORT cycle.00401057	
00401048		837D OC 02	CMP_DWORD_PTR_SS:[EBP+C], 2	
0040104C	. ~	74 09	JE SHORT cycle.00401057	
0040104E		3300	XOR EAX, EAX	
00401050	12	5E	POP ESI 注意这个行品名 主子	
00401051		E.F.	POP EDI (IEL) (IEL) EX; XXV	
00401052		5B	LUP_KBX 在RUN跟踪中这部分代码	
00401053		C9	LEAVE at th (=) t	
00401054		C2 1000	RETN 10 TXJ701J X2	
00401057	\geq	6A OO	PUSH 0	FExitCode = 0

现在我们先不管,按 F9 键(或者按 CTR+F12 组合键跟踪步过)让程序运行,再点击菜单查看->RUN 跟踪,或者点击工具栏上的那个"…"符号,打开 RUN 跟踪的记录窗口看看:

🔜 RUN	麗踪						
返回	线程	模块	地址			修改后的	寄存器 🔺
217. 216. 215. 214. 213. 212. 211. 210. 209. 209. 207. 206.	王王王王王王王王王王王王王王王王王王王王王王王王王王王王王王王王王王王王王	ntdll cycle cycle cycle cycle cycle cycle cycle cycle cycle cycle cycle trd 子 , 7 计模块, 过	7C92EB94 00401029 0040102D 0040102E 0040102E 0040102F 00401030 条cycle 古击选择到 看看有哪	RETN ENTER 0,0 PUSH EBX PUSH EDI PUSH ESI CMP DWORD PTR SS: SHORT cycle.004 SHORT cycle.004 DWORD PTR SS: MORT cycle.004 DWORD PTR SS:	反汇编窗口。 显示注释 (2) 高亮寄存器 标记选择的时 标记地址	EAX=7FFDJ EBP=00121 中跟随() () () () () () () () () () () () () (7000, I FC4C Enter
205. 204.	圭	cycie cycle	0040104C 0040104E	JE SHORT Cycle.004 XOR EAX, EAX	转到上一行 转到下一行	(P) (N)	Minus Plus
203.	±	cycie	00401050	TOLEZI	统计模块 🛯		
					全局统计 @ 清除 RUN 踞) 。 限	
					记录到文件		
					复制到剪贴树 界面选项	反	+
我们现在	主再来看	看统计问	的情况:				



现在我们在地址 4010A6 处的那条指令上按 F2, 删除所有其它的断点, 点菜单调试->关闭 RUN 跟踪, 现在我们就可以开始分 析了:

004010E2 . 8BFE	MOV EDI, ESI	; 用户名送 EDI	
004010E4 . 03F8	ADD EDI, EAX		
004010E6 . FC	CLD		
004010E7 . F3:A4	REP MOVS BYTE PTR ES: [EDI], BYTE PTR DS: [ESI]		
004010E9 . 33C9	XOR ECX, ECX	; 清零, 设循环计数器	
004010EB . BE 71214000	MOV ESI, cycle.00402171	; 注册码送ESI	
004010F0 > 41	INC ECX		
004010F1 . AC	LODS BYTE PTR DS: [ESI]	; 取注册码的每个字符	
004010F2 . 0AC0	OR AL, AL	; 判断是否为空	
004010F4 . 74 OA	JE SHORT cycle.00401100	;没有则跳走	
004010F6 . 3C 7E	CMP AL, 7E	; 判断字符是否为非ASCII字符	
004010F8 . 7F 06	JG SHORT cycle.00401100	; 非ASCII字符跳走	
004010FA . 3C 30	CMP AL, 30	; 看是否小于30H, 主要是判断是	릩
不是数字或字母等			
004010FC . 72 02	JB SHORT cycle.00401100	; 小于跳走	
004010FE . ÈB F0	JMP SHORT cycle.004010F0		
00401100 > 83F9 11	CMP ECX, 11	; 比较注册码位数,必须为十进	
制17位			
00401103 . 75 1A	JNZ SHORT cycle.0040111F		
00401105 . E8 E7000000	CALL cycle.004011F1	; 关键, F7跟进去	
0040110A . B9 01FF0000	MOV ECX, OFF01		
0040110F . 51	PUSH ECX		
00401110 . E8 7B000000	CALL cycle.00401190	; 关键, 跟进去	
00401115 . 83F9 01	CMP ECX, 1		
00401118 . 74 06	JE SHORT cycle.00401120		
0040111A > E8 47000000	CALL cycle.00401166	; 注册失败对话框	
0040111F > C3	RETN		
00401120 > A1 68214000	MOV EAX, DWORD PTR DS: [402168]		
00401125 . 8B1D 6C214000	MOV EBX, DWORD PTR DS: [40216C]		
0040112B . 33C3	XOR EAX, EBX		
0040112D . 3305 82214000	XOR EAX, DWORD PTR DS: [402182]		
00401133 . 0D 40404040	OR EAX, 40404040		
00401138 . 25 77777777	AND EAX, 77777777		

0040113D . 3305 79214000	XOR EAX, DWORD PTR DS: [402179]	
00401143 . 3305 7D214000	XOR EAX, DWORD PTR DS: [40217D]	
00401149 . 75 CF	JNZ SHORT cycle.0040111A	; 这里跳走就完蛋
0040114B . E8 2B000000	CALL cycle.0040117B	; 注册成功对话框

写到这准备跟踪算法时,才发现这个 crackme 还是挺复杂的,具体算法我就不写了,实在没那么多时间详细跟踪。有兴趣的 可以跟一下,注册码是17位,用户名采用复制的方式扩展到 16 位,如我输入 "CCDebuger",扩展后就 是 "CCDebugerCCDebug"。大致是先取扩展后用户名的前 8 位和注册码的前 8 位,把用户名的前四位和后四位分别与注册码 的前四位和后四位进行运算,算完后再把扩展后用户名的后 8 位和注册码的后 8 位分两部分,再与前面用户名和注册码的 前 8 位计算后的值进行异或计算,最后结果等于 0 就成功。注册码的第 17 位我尚未发现有何用处。对于新手来说,可能这 个 crackme 的难度大了一点。没关系,我们主要是学习 011yDBG 的使用,方法掌握就可以了。

最后说明一下:

1、这个程序在设置了消息断点后可以省略在代码段上设访问断点那一步,直接打开 RUN 跟踪,消息断点断下后按 CTR+F12 组合键让程序执行,RUN 跟踪记录中就可以找到关键地方。

2、对于这个程序,你可以不设消息断点,在输入用户名和注册码后先不按那个"Check"按钮,直接打开 RUN 跟踪,添加"所有函数过程的入口"后再回到程序中点"Check"按钮,这时在 011yDBG 中打开 RUN 跟踪记录同样可以找到关键位置。

【版权声明】 本文纯属技术交流, 转载请注明作者并保持文章的完整, 谢谢!

本站声明:看雪论坛文章版权属于作者,受法律保护。没有作者书面许可不得转载。若作者同意转载,必须以超链接形式标明文章原始出处 和作者信息及本声明!