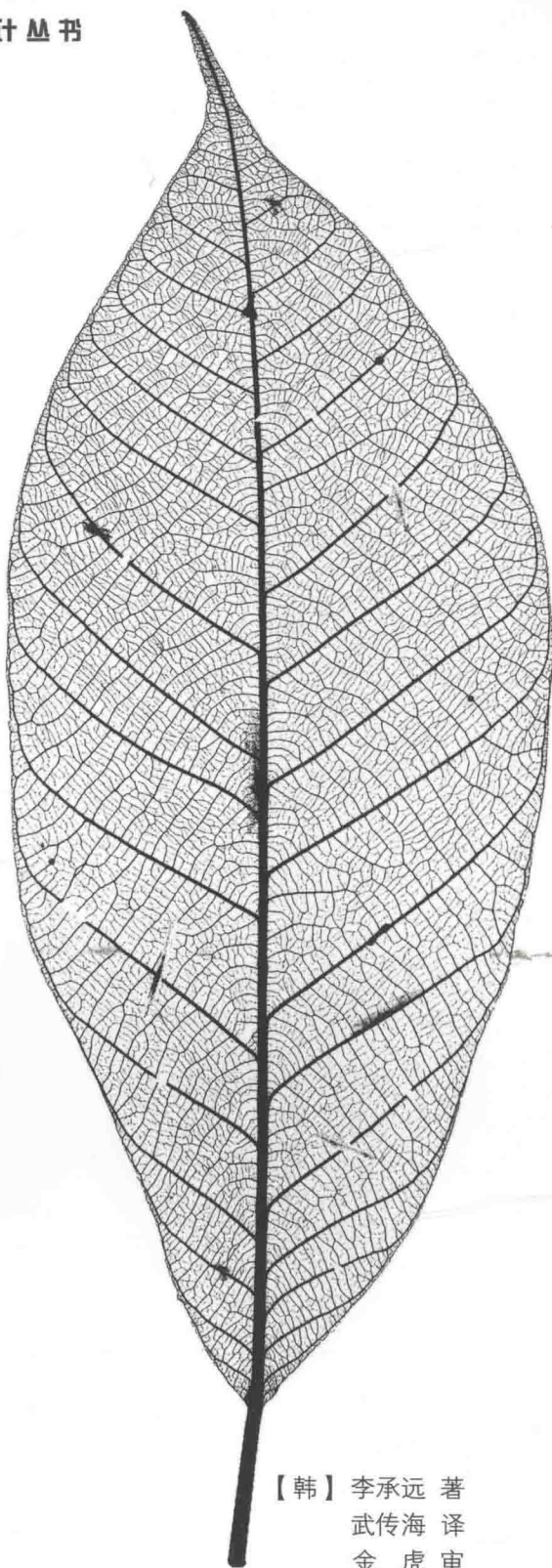


TURING

图灵程序设计丛书

인사이트
insight

核心原理



逆向工程

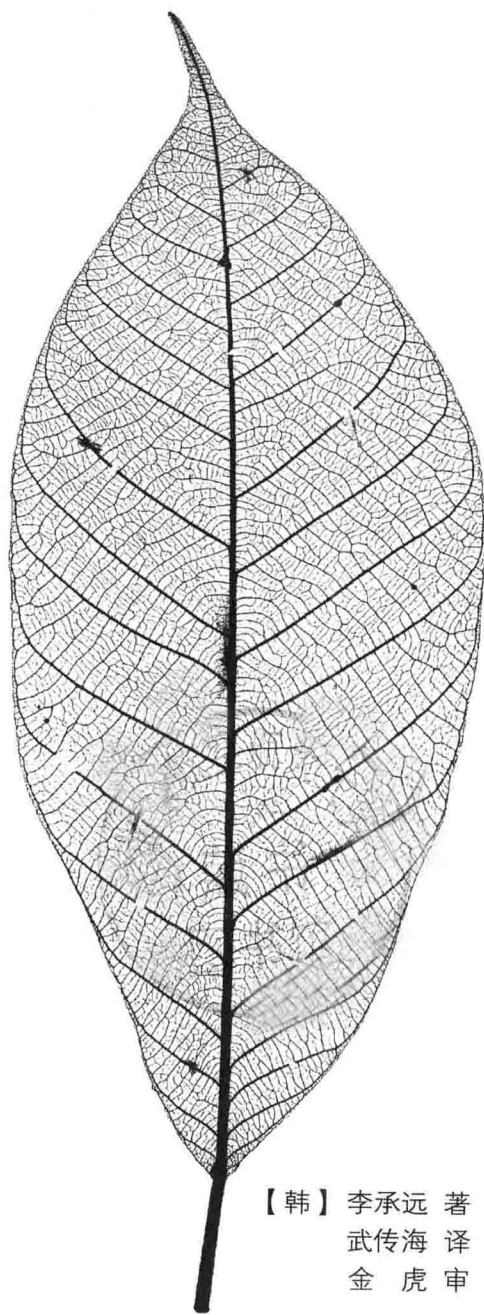
【韩】李承远 著
武传海 译
金虎 审

 人民邮电出版社
POSTS & TELECOM PRESS

TURING

图灵程序设计丛书

核心原理



逆向工程

【韩】李承远 著
武传海 译
金虎 审

人民邮电出版社
北京

图书在版编目 (CIP) 数据

逆向工程核心原理 / (韩) 李承远著 ; 武传海译.
— 北京 : 人民邮电出版社, 2014. 5
(图灵程序设计丛书)
ISBN 978-7-115-35018-3

I. ①逆… II. ①李… ②武… III. ①软件工程
IV. ①TP311.5

中国版本图书馆CIP数据核字(2014)第049919号

版 权 声 明

리버싱 핵심 원리 © 2012 by Seungwon Lee

All rights reserved.

Translation rights arranged by INSIGHT through Shinwon Agency Co., Korea.

Simplified Chinese Translation Copyright © 2014 by POSTS & TELECOM PRESS.

本书中文简体字版由 Insight 授权人民邮电出版社独家出版。未经出版者书面许可, 不得以任何方式复制或抄袭本书内容。

版权所有, 侵权必究。

内 容 提 要

本书十分详尽地介绍了代码逆向分析的核心原理。作者在 Ahnlab 研究所工作多年, 书中不仅包括其以此经验为基础亲自编写的大量代码, 还包含了逆向工程研究人员必须了解的各种技术和技巧。彻底理解并切实掌握逆向工程这门技术, 就能在众多 IT 相关领域进行拓展运用, 这本书就是通向逆向工程大门的捷径。

想成为逆向工程研究员的读者或正在从事逆向开发工作的开发人员一定会通过本书获得很大帮助。同时, 想成为安全领域专家的人也可从本书轻松起步。

◆ 著 [韩] 李承远
译 武传海
审 金 虎
责任编辑 傅志红
执行编辑 陈 曦
责任印制 焦志炜

◆ 人民邮电出版社出版发行 北京市丰台区成寿寺路11号
邮编 100164 电子邮件 315@ptpress.com.cn
网址 <http://www.ptpress.com.cn>
三河市海波印务有限公司印刷

◆ 开本: 787×1092 1/16
印张: 43.75
字数: 1 202千字 2014年5月第1版
印数: 1-4 000册 2014年5月河北第1次印刷
著作权合同登记号 图字: 01-2013-8251号



定价: 109.00元

读者服务热线: (010)51095186转600 印装质量热线: (010)81055316

反盗版热线: (010)81055315

广告经营许可证: 京崇工商广字第 0021 号

推 荐 语

向安全技术专家迈出第一步的必选书!

最近,越来越多的人开始关心信息技术安全,但是相关领域的安全技术专家仍然十分匮乏。主要有两方面原因导致这种现象形成:一方面是因为必须做大量准备,努力学习;另一方面是因为市面上缺乏系统讲解这类内容的专业书籍。

信息技术安全领域的图书很少有讲解恶意代码分析的,本书恰好填补了这一空白。无论是刚开始学习恶意代码分析的朋友,还是从事恶意代码分析的专家,都会为本书面世而激动。

虽然读者阅读本书时需要具有基本的汇编语言知识,但是本书内容讲解非常细致,涵盖了从恶意代码分析基础知识到高级技术的全部内容,系统而有条理,语言简洁,通俗易懂,并在讲解中选配了恰当的示例程序,使内容更易理解。对于最近出现的恶意代码中的各种常用技术,本书都做了详细讲解,无论你是初学者还是分析专家,都能从中获益。

信息安全技术涉及各领域,需要知识渊博、经验丰富的专家。本书将帮你轻松迈出成为安全技术专家的第一步。

——韩昌奎, ASEC 中心主任

如果此刻你手上正捧着这本书,说明你已经被代码逆向分析的魅力深深吸引了!

对于刚开始学习代码逆向分析技术的人而言,需要学习的内容很多,这容易让人心生畏惧、止步不前。其实不需担心,本书在学习过程中给出了大量提示,各位借助这些提示可以更好地理解所讲的内容。

本书比较重视代码逆向分析者的心态引导与培养,在内容讲解上也与其他书籍不同,并不是单纯的技巧罗列,而是深刻讲解了相关技术的深层含义、技术的工作原理以及内部实现结构,这也是本书的重点所在。同时配合丰富示例,让内容变得更具体形象、更易理解,作者的良苦用心可见一斑。

你想成为代码逆向分析员吗?如果你感到困惑:“我是开发人员,难道也需要读它吗?”不妨试试,它将成为你最好的同伴。

——郑官真, CISCO 高级研究员

前 言

本书将指引你进入美妙又刺激的代码逆向分析世界，开启一段神奇之旅！

软件逆向工程（代码逆向分析）是一种探究应用程序内部组成结构及工作原理的技术。欢迎各位来到代码逆向分析世界，经历各种神奇的冒险，迎接各种富有趣味的挑战。

不论是我们自己编写的程序，还是其他人编写的无源码程序，只要运用逆向分析技术，我们就能轻松窥探程序内部结构、掌握工作原理。灵活运用逆向分析技术可以在程序的开发与测试阶段发现 Bug 和漏洞，并直接修改程序文件或内存解决这些隐含的问题。而且，我们还可以借助逆向分析技术为程序添加新功能，使程序更强大。这就像是一种魔法，魅力无限。

学习逆向分析技术前并不需要准备太多。下面给各位讲讲我的经历。几年来，我一直维护着一个逆向分析技术学习博客，访客们问得最多的问题是：“究竟该怎样学习逆向分析技术？”我结合自身经验并分析了一些学习失败的案例后发现，失败的最大原因并不是学习本身的难度与要学内容的数量，而是对学习逆向技术的恐惧与忧虑——“我连 C 语言还不懂呢”、“一定要掌握汇编语言吗”、“OS 架构我还没搞明白”、“不知道怎么用调试器”、“学完这么多内容才能真正开始学逆向分析技术啊”，以上这些担心正是迫使学习者们中途放弃的主要原因。其实，学习逆向分析与学习 C 语言、汇编语言、OS 架构、调试器用法等内容是一样的。将这些内容全部掌握的人已经是专家了，当然不需要这个入门过程。

你仍对逆向分析技术一无所知？没关系，不必沮丧，这反而是件好事。因为你会在以后的学习过程中学到很多东西，会变得更聪明、更有价值，谁说这不是件好事呢？

如果你梦想成为逆向分析工程师，但不知如何入门；如果你是一名程序开发者，又对逆向分析技术非常感兴趣，那么本书将非常适合阅读。学习逆向分析技术并不像公式一样背下来就可以了，死记硬背的结果是你会不知道如何灵活运用。学习某些知识技术时，不仅要掌握其本身，还要知道它们的内部机制与工作原理，既知其然又知其所以然，这才是最重要的。所以，本书讲解相关知识与技术时，将讲解重点放在对其工作原理的分析与说明上，这将更有利于各位真正掌握它们。为什么本书非常适合作为逆向分析技术的入门书呢？以下是我的几点理由。

第一，开发与分析的经验。一名逆向分析工程师不仅要具备专业的逆向分析技术，还要具有一定的程序开发能力。我以前从事网络应用程序开发，后来开始做恶意代码分析工作，慢慢就对逆向分析技术熟悉起来。可以说，我是从程序开发者转变为逆向分析员的为数不多的人之一。程序开发与逆向分析这两项技术相辅相成、互为补充、共同发展，形成相得益彰的效果。日常工作中，它们就像一双翅膀应用于各类业务。分析程序时，人们自然就会从程序开发与逆向分析这两个角度着手。书中用到的几乎所有示例都基于我在逆向分析实践中获得的知识与经验，是我亲自

开发的程序，紧扣各章主题，绝无累赘。

第二，培训与演讲的经验。我在公司慢慢有了资历，随之而来的培训、研讨会、演讲也逐渐多起来。我进行逆向分析技术培训时，面对的学员大都是初学者，这些机会非常有利于我了解他们遇到的困难与想知道的东西。所以，我一直在用心思考，如何用更易理解的方式传授要讲解的知识，这样就慢慢形成了自己特有的讲解技巧与风格。编写本书时，我又将这些培训经验应用到本书的组织结构、内容讲解、示例选择等各方面，以求将较为难懂的技术以更易懂的方式呈现给各位。

第三，丰富的沟通经验。我几年前就开设了一个逆向技术学习博客并运营至今。刚开始的想法非常简单，就是想归纳整理自己学到的逆向分析技术。后来访客越来越多，留下的问题也多起来。我感到很惊讶，之前一直以为韩国是逆向分析技术的“不毛之地”，结果关注逆向分析技术的人比我想得要多，并且他们关注的范围也非常广泛。这大大拓宽了我的视野，于是我开始访问其他逆向分析技术学习博客，接触到了更多文章与问题，慢慢了解了他们关注的部分。我在此过程中逐渐认识到，初学者们想知道的是逆向分析技术的系统的学习方法。他们入门之后迫切要学习的是专业的逆向分析技术与内部工作原理。有感于此，我就萌生了写一本系统学习逆向分析技术书的想法，就这样，在逆向分析与开发、培训、演讲、交流等经验基础上，这本逆向分析技术学习入门书诞生了。

那么，读者应该如何使用本书学习逆向分析技术呢？对此，我给出如下几点建议，供各位参考。

第一，技术书不是装饰书架的道具，它们是提高各位技术水平的工具。所以阅读时要勾画出重要部分，在书页空白处写下自己的想法与心得等。阅读时，在书页上记录相关技术、注意事项、技术优缺点、与作者的不同见解等，让它成为只属于你的书。读完这样一本逆向分析技术书后，不知不觉间就构建出自己独特的逆向分析世界，最终成为代码逆向分析专家。

第二，拥有积极乐观的心态。逆向分析是一项深奥的技术，会涉及 OS 底层知识。要学的内容很多，并且大部分内容需要亲自测试并确认才能最终理解。必须用积极乐观的心态对待这一过程，学习逆向技术无关聪明与否，只跟投入时间的多少有关。学习时，不要太急躁，请保持轻松的心态。

第三，不断挑战。逆向分析不尽如人意时，不要停下来，要尝试其他方法，不断挑战。要相信一定会有解决的方法，可能几年前早已有人成功过了。搜索相关资料并不断尝试，不仅能提高自身技术水平，解决问题后，心里还能感受到一种成就感。这样的成功经验一点点积累起来，自信心就会大大增强，自身的逆向分析水平也会得到明显提高。这种从经验中获得的自信会不知不觉地对逆向分析过程产生积极影响，让逆向分析往更好的方向发展。

希望本书能够帮助各位把“心愿表”上的愿望一一实现，也希望各位把本书讲解的知识、技术广泛应用到逆向分析过程中，发挥更大的作用。谢谢。

感谢

动笔容易写完难。我只身一人是无法完成本书的，写作过程中得到了很多人的关心、支持与

鼓励，没有他们就不会有本书。借此机会，我向所有给予帮助的人表示最诚挚的谢意。

首先，感谢爱妻素英，谢谢你一直以来相信我、默默支持我，你的微笑是我的能量之源。还要感谢我的两个宝贝儿子浩俊、姜宪，你们的陪伴让疲劳烟消云散，让我心里始终充满幸福。还有我的父母，你们总是关心着我和我的工作。正是你们给了我战胜一切困难的勇气，衷心感谢你们。

其次，要感谢崔景哲先生，您的称赞与鼓励一直激励着我写完全书。还要感谢韩昌圭先生与郑宽镇先生，两位前辈写的推荐词使本书增色不少。我早在动笔时就嘱托二位为本书写推荐语了。请一定允许我为两位的新书写推荐词。

再次，向 Insight 出版社的韩基晟社长以及所有员工，特别是赵岩熹编辑表示最诚挚的谢意。他将一块粗糙的原石打磨成了珍贵的宝石，我以后写书也会无条件请他负责。

最后，对关注本书的同事、公司实习生，以及博客访客们表示感谢。你们总是热心地询问：“什么时候出书啊？”你们的关心最终促成我写完全书。还要感谢购买本书的读者们，你们的梦想与热情一直鼓舞着我。

李承远

www.reversecore.com

reverscore@gmail.com

目 录

第一部分 代码逆向技术基础	
第 1 章 关于逆向工程	2
1.1 逆向工程	2
1.2 代码逆向工程	2
1.2.1 逆向分析法	2
1.2.2 源代码、十六进制代码、汇编 代码	4
1.2.3 “打补丁”与“破解”	5
1.3 代码逆向准备	5
1.3.1 目标	5
1.3.2 激情	6
1.3.3 谷歌	6
1.4 学习逆向分析技术的禁忌	6
1.4.1 贪心	6
1.4.2 急躁	7
1.5 逆向分析技术的乐趣	7
第 2 章 逆向分析 Hello World!程序	8
2.1 Hello World!程序	8
2.2 调试 HelloWorld.exe 程序	9
2.2.1 调试目标	9
2.2.2 开始调试	9
2.2.3 入口点	10
2.2.4 跟踪 40270C 函数	10
2.2.5 跟踪 40104F 跳转语句	12
2.2.6 查找 main()函数	12
2.3 进一步熟悉调试器	14
2.3.1 调试器指令	14
2.3.2 “大本营”	15
2.3.3 设置“大本营”的四种方法	15
2.4 快速查找指定代码的四种方法	17
2.4.1 代码执行法	18
2.4.2 字符串检索法	19
2.4.3 API 检索法 (1): 在调用代码 中设置断点	20
2.4.4 API 检索法 (2): 在 API 代码 中设置断点	21
2.5 使用“打补丁”方式修改“Hello World!”字符串	23
2.5.1 “打补丁”	23
2.5.2 修改字符串的两种方法	24
2.6 小结	28
第 3 章 小端序标记法	31
3.1 字节序	31
3.1.1 大端序与小端序	32
3.1.2 在 OllyDbg 中查看小端序	32
第 4 章 IA-32 寄存器基本讲解	34
4.1 什么是 CPU 寄存器	34
4.2 IA-32 寄存器	34
4.3 小结	40
第 5 章 栈	41
5.1 栈	41
5.1.1 栈的特征	41
5.1.2 栈操作示例	41
第 6 章 分析 abex' crackme#1	44
6.1 abex' crackme #1	44
6.1.1 开始调试	45
6.1.2 分析代码	45
6.2 破解	47
6.3 将参数压入栈	47
6.4 小结	48
第 7 章 栈帧	49
7.1 栈帧	49
7.2 调试示例: stackframe.exe	49
7.2.1 StackFrame.cpp	50
7.2.2 开始执行 main()函数&生成栈帧	51
7.2.3 设置局部变量	52
7.2.4 add()函数参数传递与调用	53

7.2.5	开始执行 add()函数&生成栈帧	54	10.1.2	stdcall	77
7.2.6	设置 add()函数的局部变量 (x, y)	55	10.1.3	fastcall	78
7.2.7	ADD 运算	55	第 11 章	视频讲座	79
7.2.8	删除函数 add()的栈帧&函数 执行完毕 (返回)	56	11.1	运行	79
7.2.9	从栈中删除函数 add()的参数 (整理栈)	57	11.2	分析	79
7.2.10	调用 printf()函数	58	11.2.1	目标 (1): 去除消息框	79
7.2.11	设置返回值	58	11.2.2	打补丁 (1): 去除消息框	81
7.2.12	删除栈帧&main()函数终止	58	11.2.3	目标 (2): 查找注册码	83
7.3	设置 OllyDbg 选项	59	11.3	小结	85
7.3.1	Disasm 选项	59	第 12 章	究竟应当如何学习代码逆向 分析	86
7.3.2	Analysis1 选项	60	12.1	逆向工程	86
7.4	小结	61	12.1.1	任何学习都应当有目标	86
第 8 章	abex' crackme #2	62	12.1.2	拥有积极心态	86
8.1	运行 abex' crackme #2	62	12.1.3	要感受其中的乐趣	86
8.2	Visual Basic 文件的特征	63	12.1.4	让检索成为日常生活的 一部分	87
8.2.1	VB 专用引擎	63	12.1.5	最重要的是实践	87
8.2.2	本地代码和伪代码	63	12.1.6	请保持平和的心态	87
8.2.3	事件处理程序	63	第二部分	PE 文件格式	
8.2.4	未文档化的结构体	63	第 13 章	PE 文件格式	90
8.3	开始调试	63	13.1	介绍	90
8.3.1	间接调用	64	13.2	PE 文件格式	90
8.3.2	RT_MainStruct 结构体	64	13.2.1	基本结构	91
8.3.3	ThunRTMain()函数	65	13.2.2	VA&RVA	92
8.4	分析 crackme	65	13.3	PE 头	92
8.4.1	检索字符串	65	13.3.1	DOS 头	93
8.4.2	查找字符串地址	66	13.3.2	DOS 存根	94
8.4.3	生成 Serial 的算法	68	13.3.3	NT 头	94
8.4.4	预测代码	69	13.3.4	NT 头: 文件头	95
8.4.5	读取 Name 字符串的代码	69	13.3.5	NT 头: 可选头	97
8.4.6	加密循环	70	13.3.6	节区头	101
8.4.7	加密方法	70	13.4	RVA to RAW	104
8.5	小结	72	13.5	IAT	105
第 9 章	Process Explorer——最优秀 的进程管理工具	74	13.5.1	DLL	105
9.1	Process Explorer	74	13.5.2	IMAGE_IMPORT_ DESCRIPTOR	107
9.2	具体有哪些优点呢	75	13.5.3	使用 notepad.exe 练习	108
9.3	sysinternals	75	13.6	EAT	112
第 10 章	函数调用约定	76	13.6.1	IMAGE_EXPORT_ DIRECTORY	113
10.1	函数调用约定	76	13.6.2	使用 kernel32.dll 练习	114
10.1.1	cdecl	76			

13.7 高级 PE.....	116	17.2.2 删除.reloc 节区.....	143
13.7.1 PEView.exe.....	116	17.2.3 修改 IMAGE_FILE_	
13.7.2 Patched PE.....	117	HEADER.....	143
13.8 小结.....	118	17.2.4 修改 IMAGE_OPTIONAL_	
第 14 章 运行时压缩	121	HEADER.....	144
14.1 数据压缩.....	121	17.3 小结.....	145
14.1.1 无损压缩.....	121	第 18 章 UPack PE 文件头详细分析	146
14.1.2 有损压缩.....	121	18.1 UPack 说明.....	146
14.2 运行时压缩器.....	122	18.2 使用 UPack 压缩 notepad.exe.....	146
14.2.1 压缩器.....	122	18.3 使用 Stud_PE 工具.....	148
14.2.2 保护器.....	123	18.4 比较 PE 文件头.....	148
14.3 运行时压缩测试.....	123	18.4.1 原 notepad.exe 的 PE 文件	
第 15 章 调试 UPX 压缩的 notepad		头.....	149
程序.....	127	18.4.2 notepad_upack.exe 运行时	
15.1 notepad.exe 的 EP 代码.....	127	压缩的 PE 文件头.....	149
15.2 notepad_upx.exe 的 EP 代码.....	127	18.5 分析 UPack 的 PE 文件头.....	150
15.3 跟踪 UPX 文件.....	129	18.5.1 重叠文件头.....	150
15.3.1 OllyDbg 的跟踪命令.....	129	18.5.2 IMAGE_FILE_HEADER.	
15.3.2 循环 #1.....	129	SizeOfOptionalHeader.....	150
15.3.3 循环 #2.....	130	18.5.3 IMAGE_OPTIONAL_	
15.3.4 循环 #3.....	131	HEADER.NumberOf-	
15.3.5 循环 #4.....	131	RvaAndSizes.....	152
15.4 快速查找 UPX OEP 的方法.....	132	18.5.4 IMAGE_SECTION_	
15.4.1 在 POPAD 指令后的 JMP		HEADER.....	153
指令处设置断点.....	132	18.5.5 重叠节区.....	155
15.4.2 在栈中设置硬件断点.....	133	18.5.6 RVA to RAW.....	156
15.5 小结.....	133	18.5.7 导入表 (IMAGE_IMPORT_	
第 16 章 基址重定位表	135	DESCRIPTOR array).....	158
16.1 PE 重定位.....	135	18.5.8 导入地址表.....	160
16.1.1 DLL/SYS.....	135	18.6 小结.....	161
16.1.2 EXE.....	136	第 19 章 UPack 调试 - 查找 OEP	162
16.2 PE 重定位时执行的操作.....	136	19.1 OllyDbg 运行错误.....	162
16.3 PE 重定位操作原理.....	138	19.2 解码循环.....	163
16.3.1 基址重定位表.....	138	19.3 设置 IAT.....	165
16.3.2 IMAGE_BASE_RELOCATION		19.4 小结.....	166
结构体.....	139	第 20 章 “内嵌补丁” 练习	167
16.3.3 基址重定位表的分析方法.....	139	20.1 内嵌补丁.....	167
16.3.4 练习.....	141	20.2 练习: Patchme.....	168
第 17 章 从可执行文件中删除.reloc		20.3 调试: 查看代码流.....	168
节区.....	142	20.4 代码结构.....	172
17.1 .reloc 节区.....	142	20.5 “内嵌补丁” 练习.....	173
17.2 reloc.exe.....	142	20.5.1 补丁代码要设置在何处呢.....	173
17.2.1 删除.reloc 节区头.....	142	20.5.2 制作补丁代码.....	175

20.5.3 执行补丁代码	176
20.5.4 结果确认	177
第三部分 DLL 注入	
第 21 章 Windows 消息钩取	180
21.1 钩子	180
21.2 消息钩子	180
21.3 SetWindowsHookEx()	181
21.4 键盘消息钩取练习	182
21.4.1 练习示例 HookMain.exe	182
21.4.2 分析源代码	185
21.5 调试练习	187
21.5.1 调试 HookMain.exe	188
21.5.2 调试 Notepad.exe 进程内的 KeyHook.dll	190
21.6 小结	192
第 22 章 恶意键盘记录器	194
22.1 恶意键盘记录器的目标	194
22.1.1 在线游戏	194
22.1.2 网上银行	194
22.1.3 商业机密泄露	194
22.2 键盘记录器的种类与发展趋势	195
22.3 防范恶意键盘记录器	195
22.4 个人信息	195
第 23 章 DLL 注入	197
23.1 DLL 注入	197
23.2 DLL 注入示例	198
23.2.1 改善功能与修复 Bug	198
23.2.2 消息钩取	198
23.2.3 API 钩取	198
23.2.4 其他应用程序	199
23.2.5 恶意代码	199
23.3 DLL 注入的实现方法	199
23.4 CreateRemoteThread()	199
23.4.1 练习示例 myhack.dll	199
23.4.2 分析示例源代码	203
23.4.3 调试方法	208
23.5 AppInit_DLLs	210
23.5.1 分析示例源码	211
23.5.2 练习示例 myhack2.dll	212
23.6 SetWindowsHookEx()	214
23.7 小结	214
第 24 章 DLL 卸载	216
24.1 DLL 卸载的工作原理	216
24.2 实现 DLL 卸载	216
24.2.1 获取进程中加载的 DLL 信息	219
24.2.2 获取目标进程的句柄	220
24.2.3 获取 FreeLibrary() API 地址	220
24.2.4 在目标进程中运行线程	220
24.3 DLL 卸载练习	220
24.3.1 复制文件及运行 notepad.exe	220
24.3.2 注入 myhack.dll	221
24.3.3 卸载 myhack.dll	222
第 25 章 通过修改 PE 加载 DLL	224
25.1 练习文件	224
25.1.1 TextView.exe	224
25.1.2 TextView_patched.exe	225
25.2 源代码 - myhack3.cpp	227
25.2.1 DllMain()	227
25.2.2 DownloadURL()	228
25.2.3 DropFile()	229
25.2.4 dummy()	230
25.3 修改 TextView.exe 文件的准备工作	231
25.3.1 修改思路	231
25.3.2 查看 IDT 是否有足够空间	231
25.3.3 移动 IDT	233
25.4 修改 TextView.exe	235
25.4.1 修改导入表的 RVA 值	235
25.4.2 删除绑定导入表	235
25.4.3 创建新 IDT	235
25.4.4 设置 Name、INT、IAT	236
25.4.5 修改 IAT 节区的属性值	238
25.5 检测验证	240
25.6 小结	241
第 26 章 PE Tools	242
26.1 PE Tools	242
26.1.1 进程内存转储	243
26.1.2 PE 编辑器	245
26.2 小结	245
第 27 章 代码注入	247
27.1 代码注入	247
27.2 DLL 注入与代码注入	247
27.3 练习示例	249
27.3.1 运行 notepad.exe	249
27.3.2 运行 CodeInjection.exe	249

27.3.3	弹出消息框	250	28.6.11	压入 MessageBoxA()函数的 参数 4 -NULL	279
27.4	CodeInjection.cpp	250	28.6.12	调用 MessageBoxA()	279
27.4.1	main()函数	251	28.6.13	设置 ThreadProc()函数的 返回值	280
27.4.2	ThreadProc()函数	251	28.6.14	删除栈帧及函数返回	280
27.4.3	InjectCode()函数	254	28.7	小结	280
27.5	代码注入调试练习	256	第四部分 API 钩取		
27.5.1	调试 notepad.exe	256	第 29 章 API 钩取：逆向分析之“花” 282		
27.5.2	设置 OllyDbg 选项	256	29.1	钩取	282
27.5.3	运行 CodeInjection.exe	257	29.2	API 是什么	282
27.5.4	线程开始代码	258	29.3	API 钩取	283
27.6	小结	259	29.3.1	正常调用 API	283
第 28 章 使用汇编语言编写注入代码 260			29.3.2	钩取 API 调用	284
28.1	目标	260	29.4	技术图表	284
28.2	汇编编程	260	29.4.1	方法对象 (是什么)	285
28.3	OllyDbg 的汇编命令	260	29.4.2	位置 (何处)	285
28.3.1	编写 ThreadProc()函数	262	29.4.3	技术 (如何)	286
28.3.2	保存文件	265	29.4.4	API	286
28.4	编写代码注入程序	266	第 30 章 记事本 WriteFile() API 钩取 288		
28.4.1	获取 ThreadProc()函数的二 进制代码	266	30.1	技术图表 - 调试技术	288
28.4.2	CodeInjection2.cpp	267	30.2	关于调试器的说明	289
28.5	调试练习	270	30.2.1	术语	289
28.5.1	调试 notepad.exe	270	30.2.2	调试器功能	289
28.5.2	设置 OllyDbg 选项	270	30.2.3	调试器的工作原理	289
28.5.3	运行 CodeInjection2.exe	271	30.2.4	调试事件	289
28.5.4	线程起始代码	272	30.3	调试技术流程	290
28.6	详细分析	272	30.4	练习	291
28.6.1	生成栈帧	272	30.5	工作原理	293
28.6.2	THREAD_PARAM 结构体 指针	273	30.5.1	栈	293
28.6.3	“User32.dll” 字符串	274	30.5.2	执行流	295
28.6.4	压入 “user32.dll” 字符串 参数	274	30.5.3	“脱钩” & “钩子”	295
28.6.5	调用 LoadLibraryA (“user32.dll”)	275	30.6	源代码分析	295
28.6.6	“MessageBoxA” 字符串	276	30.6.1	main()	296
28.6.7	调用 GetProcAddress(hMod, “MessageBoxA”)	276	30.6.2	DebugLoop()	296
28.6.8	压入 MessageBoxA()函数的 参数 1 - MB_OK	277	30.6.3	EXIT_PROCESS_DEBUG_ EVENT	298
28.6.9	压入 MessageBoxA()函数的 参数 2 - “ReverseCore”	277	30.6.4	CREATE_PROCESS_DEBUG_ EVENT-OnCreateProcess- DebugEvent()	298
28.6.10	压入 MessageBoxA()函数 的参数 3 - “www.reversecore. com”	278	30.6.5	EXCEPTION_DEBUG_ EVENT-OnException- DebugEvent()	300

第 31 章 关于调试器	305	33.7.1 复制 stealth2.dll 文件到 %SYSTEM%文件夹中	345
31.1 OllyDbg	305	33.7.2 运行 HideProc2.exe -hide	346
31.2 IDA Pro	305	33.7.3 运行 ProcExp.exe& notepad.exe	346
31.3 WinDbg	306	33.7.4 运行 HideProc2.exe -show	347
第 32 章 计算器显示中文数字	308	33.8 源代码分析	348
32.1 技术图表	308	33.8.1 HideProc2.cpp	348
32.2 选定目标 API	309	33.8.2 stealth2.cpp	348
32.3 IAT 钩取工作原理	312	33.9 利用“热补丁”技术钩取 API	350
32.4 练习示例	314	33.9.1 API 代码修改技术的问题	350
32.5 源代码分析	316	33.9.2 “热补丁”(修改 7 个字节 代码)	350
32.5.1 DllMain()	316	33.10 练习 #3: stealth3.dll	353
32.5.2 MySetWindowTextW()	317	33.11 源代码分析	353
32.5.3 hook_iat()	319	33.12 使用“热补丁”API 钩取技术时 需要考虑的问题	356
32.6 调试被注入的 DLL 文件	322	33.13 小结	357
32.6.1 DllMain()	325	第 34 章 高级全局 API 钩取: IE 连接 控制	359
32.6.2 hook_iat()	325	34.1 目标 API	359
32.6.3 MySetWindowTextW()	327	34.2 IE 进程结构	361
32.7 小结	328	34.3 关于全局 API 钩取的概念	362
第 33 章 隐藏进程	329	34.3.1 常规 API 钩取	363
33.1 技术图表	329	34.3.2 全局 API 钩取	363
33.2 API 代码修改技术的原理	329	34.4 ntDll!ZwResumeThread() API	364
33.2.1 钩取之前	330	34.5 练习示例: 控制 IE 网络连接	368
33.2.2 钩取之后	330	34.5.1 运行 IE	368
33.3 进程隐藏	332	34.5.2 注入 DLL	369
33.3.1 进程隐藏工作原理	332	34.5.3 创建新选项卡	369
33.3.2 相关 API	332	34.5.4 尝试连接网站	370
33.3.3 隐藏技术的问题	333	34.5.5 卸载 DLL	371
33.4 练习 #1 (HideProc.exe, stealth.dll)	333	34.5.6 课外练习	372
33.4.1 运行 notepad.exe、 procxp.exe、taskmgr.exe	334	34.6 示例源代码	372
33.4.2 运行 HideProc.exe	334	34.6.1 DllMain()	372
33.4.3 确认 stealth.dll 注入成功	334	34.6.2 NewInternetConnectW()	373
33.4.4 查看 notepad.exe 进程是否 隐藏成功	335	34.6.3 NewZwResumeThread()	374
33.4.5 取消 notepad.exe 进程隐藏	336	34.7 小结	375
33.5 源代码分析	336	第 35 章 优秀分析工具的五种标准	376
33.5.1 HideProc.cpp	336	35.1 工具	376
33.5.2 stealth.cpp	338	35.2 代码逆向分析工程师	376
33.6 全局 API 钩取	344	35.3 优秀分析工具的五种标准	376
33.6.1 Kernel32.CreateProcess() API	344	35.3.1 精简工具数量	377
33.6.2 Ntdll.ZwResumeThread() API	345	35.3.2 工具功能简单、使用方便	377
33.7 练习 #2 (HideProc2.exe,Stealth2.dll)	345		

35.3.3 完全掌握各种功能.....	377	39.1.3 内核调试.....	407
35.3.4 不断升级更新.....	377	39.1.4 WinDbg 基本指令.....	409
35.3.5 理解工具的核心工作原理.....	377		
35.4 熟练程度的重要性.....	377		
第五部分 64 位 Windows 内核 6			
第 36 章 64 位计算	380	第 40 章 64 位调试	411
36.1 64 位计算环境.....	380	40.1 x64 环境下的调试器.....	411
36.1.1 64 位 CPU.....	380	40.2 64 位调试.....	411
36.1.2 64 位 OS.....	381	40.3 PE32: WOW64Test_x86.exe.....	413
36.1.3 Win32 API.....	381	40.3.1 EP 代码.....	414
36.1.4 WOW64.....	381	40.3.2 Startup 代码.....	414
36.1.5 练习: WOW64Test.....	384	40.3.3 main() 函数.....	415
36.2 编译 64 位文件.....	385	40.4 PE32+: WOW64Test_x64.exe.....	416
36.2.1 Microsoft Windows SDK		40.4.1 系统断点.....	416
(Software Development Kit)	386	40.4.2 EP 代码.....	417
36.2.2 设置 Visual C++ 2010 Express		40.4.3 Startup 代码.....	418
环境.....	386	40.4.4 main() 函数.....	420
		40.5 小结.....	423
第 37 章 x64 处理器	389	第 41 章 ASLR	424
37.1 x64 中新增或变更的项目.....	389	41.1 Windows 内核版本.....	424
37.1.1 64 位.....	389	41.2 ASLR.....	424
37.1.2 内存.....	389	41.3 Visual C++.....	424
37.1.3 通用寄存器.....	389	41.4 ASLR.exe.....	425
37.1.4 CALL/JMP 指令.....	390	41.4.1 节区信息.....	426
37.1.5 函数调用约定.....	391	41.4.2 IMAGE_FILE_HEADER\ Characteristics.....	427
37.1.6 栈 & 栈帧.....	392	41.4.3 IMAGE_OPTIONAL_HEADER\ DLL Characteristics.....	428
37.2 练习: Stack32.exe & Stack64.exe.....	392	41.5 练习: 删除 ASLR 功能.....	428
37.2.1 Stack32.exe.....	392	第 42 章 内核 6 中的会话	430
37.2.2 Stack64.exe.....	394	42.1 会话.....	430
37.3 小结.....	397	42.2 会话 0 隔离机制.....	432
第 38 章 PE32+	398	42.3 增强安全性.....	432
38.1 PE32+ (PE+, PE64).....	398	第 43 章 内核 6 中的 DLL 注入	433
38.1.1 IMAGE_NT_HEADERS.....	398	43.1 再现 DLL 注入失败.....	433
38.1.2 IMAGE_FILE_HEADER.....	398	43.1.1 源代码.....	433
38.1.3 IMAGE_OPTIONAL_		43.1.2 注入测试.....	435
HEADER.....	399	43.2 原因分析.....	436
38.1.4 IMAGE_THUNK_DATA.....	401	43.2.1 调试 #1.....	436
38.1.5 IMAGE_TLS_DIRECTORY.....	403	43.2.2 调试 #2.....	438
第 39 章 WinDbg	405	43.3 练习: 使 CreateRemoteThread() 正常	
39.1 WinDbg.....	405	工作.....	440
39.1.1 WinDbg 的特征.....	405	43.3.1 方法 #1: 修改 CreateSuspended	
39.1.2 运行 WinDbg.....	406	参数值.....	440
		43.3.2 方法 #2: 操纵条件分支.....	441

43.4	稍作整理	443	46.3	小结	472
43.5	InjectDll_new.exe	443	第 47 章 PEB		473
43.5.1	InjectDll_new.cpp	443	47.1	PEB	473
43.5.2	注入练习	446	47.1.1	PEB 访问方法	473
第 44 章 InjDll.exe: DLL 注入专用工具		448	47.1.2	PEB 结构体的定义	474
44.1	InjDll.exe	448	47.1.3	PEB 结构体的成员	475
44.1.1	使用方法	448	47.2	PEB 的重要成员	477
44.1.2	使用示例	449	47.2.1	PEB.BeingDebugged	478
44.1.3	注意事项	450	47.2.2	PEB.ImageBaseAddress	478
第六部分 高级逆向分析技术			47.2.3	PEB.Ldr	479
第 45 章 TLS 回调函数		452	47.2.4	PEB.ProcessHeap & PEB.NtGlobalFlag	480
45.1	练习 #1: HelloTls.exe	452	47.3	小结	480
45.2	TLS	453	第 48 章 SEH		481
45.2.1	IMAGE_DATA_DIRECTORY[9]	453	48.1	SEH	481
45.2.2	IMAGE_TLS_DIRECTORY	454	48.2	SEH 练习示例 #1	481
45.2.3	回调函数地址数组	454	48.2.1	正常运行	481
45.3	TLS 回调函数	455	48.2.2	调试运行	482
45.4	练习 #2: TlsTest.exe	456	48.3	OS 的异常处理方法	484
45.4.1	DLL_PROCESS_ATTACH	457	48.3.1	正常运行时的异常处理方法	484
45.4.2	DLL_THREAD_ATTACH	457	48.3.2	调试运行时的异常处理方法	484
45.4.3	DLL_THREAD_DETACH	457	48.4	异常	485
45.4.4	DLL_PROCESS_DETACH	457	48.4.1	EXCEPTION_ACCESS_VIOLATION(C0000005)	486
45.5	调试 TLS 回调函数	458	48.4.2	EXCEPTION_BREAKPOINT(80000003)	486
45.6	手工添加 TLS 回调函数	459	48.4.3	EXCEPTION_ILLEGAL_INSTRUCTION(C000001D)	488
45.6.1	修改前的原程序	460	48.4.4	EXCEPTION_INT_DIVIDE_BY_ZERO(C0000094)	488
45.6.2	设计规划	460	48.4.5	EXCEPTION_SINGLE_STEP(80000004)	489
45.6.3	编辑 PE 文件头	461	48.5	SEH 详细说明	489
45.6.4	设置 IMAGE_TLS_DIRECTORY 结构体	463	48.5.1	SEH 链	489
45.6.5	编写 TLS 回调函数	464	48.5.2	异常处理函数的定义	489
45.6.6	最终完成	464	48.5.3	TEB.NtTib.ExceptionList	491
45.7	小结	465	48.5.4	SEH 安装方法	492
第 46 章 TEB		466	48.6	SEH 练习示例 #2 (seh.exe)	492
46.1	TEB	466	48.6.1	查看 SEH 链	493
46.1.1	TEB 结构体的定义	466	48.6.2	添加 SEH	493
46.1.2	TEB 结构体成员	466	48.6.3	发生异常	494
46.1.3	重要成员	469	48.6.4	查看异常处理器参数	494
46.2	TEB 访问方法	470	48.6.5	调试异常处理器	496
46.2.1	Ntdll.NtCurrentTeb()	470	48.6.6	删除 SEH	498
46.2.2	FS 段寄存器	471			

48.7 设置 OllyDbg 选项.....	499	50.3.2 动态反调试技术.....	528
48.7.1 忽略 KERNEL32 中发生的 内存非法访问异常.....	500	第 51 章 静态反调试技术	529
48.7.2 向被调试者派送异常.....	500	51.1 静态反调试的目的.....	529
48.7.3 其他异常处理.....	500	51.2 PEB.....	529
48.7.4 简单练习.....	500	51.2.1 BeingDebugged(+0x2).....	531
48.8 小结.....	501	51.2.2 Ldr(+0xC).....	531
第 49 章 IA-32 指令	502	51.2.3 Process Heap(+0x18).....	532
49.1 IA-32 指令.....	502	51.2.4 NtGlobalFlag(+0x68).....	533
49.2 常用术语.....	502	51.2.5 练习: StaAD_PEB.exe.....	534
49.2.1 反汇编器.....	503	51.2.6 破解之法.....	534
49.2.2 反编译器.....	504	51.3 NtQueryInformationProcess().....	537
49.2.3 反编译简介.....	504	51.3.1 ProcessDebugPort(0x7).....	538
49.3 IA-32 指令格式.....	506	51.3.2 ProcessDebugObjectHandle (0x1E).....	539
49.3.1 指令前缀.....	507	51.3.3 ProcessDebugFlags(0x1F).....	539
49.3.2 操作码.....	507	51.3.4 练习: StaAD_NtQIP.exe.....	540
49.3.3 ModR/M.....	507	51.3.5 破解之法.....	540
49.3.4 SIB.....	508	51.4 NtQuerySystemInformation().....	542
49.3.5 位移.....	508	51.4.1 SystemKernelDebugger- Information(0x23).....	544
49.3.6 立即数.....	509	51.4.2 练习: StaAD_NtQSI.exe.....	545
49.4 指令解析手册.....	509	51.4.3 破解之法.....	545
49.4.1 下载 IA-32 用户手册.....	509	51.5 NtQueryObject().....	545
49.4.2 打印指令解析手册.....	509	51.6 ZwSetInformationThread().....	549
49.5 指令解析练习.....	510	51.6.1 练习: StaAD_ZwSIT.exe.....	549
49.5.1 操作码映射.....	510	51.6.2 破解之法.....	550
49.5.2 操作数.....	511	51.7 TLS 回调函数.....	550
49.5.3 ModR/M.....	512	51.8 ETC.....	551
49.5.4 Group.....	514	51.8.1 练习: StaAD_FindWindow .exe.....	551
49.5.5 前缀.....	516	51.8.2 破解之法.....	551
49.5.6 双字节操作码.....	518	51.9 小结.....	553
49.5.7 移位值&立即数.....	519	第 52 章 动态反调试技术	554
49.5.8 SIB.....	520	52.1 动态反调试技术的目的.....	554
49.6 指令解析课外练习.....	524	52.2 异常.....	554
49.7 小结.....	524	52.2.1 SEH.....	554
		52.2.2 SetUnhandledException- Filter().....	558
		52.3 Timing Check.....	562
		52.3.1 时间间隔测量法.....	562
		52.3.2 RDTSC.....	563
		52.4 陷阱标志.....	565
		52.4.1 单步执行.....	566
		52.4.2 INT 2D.....	569
第七部分 反调试技术			
第 50 章 反调试技术	526		
50.1 反调试技术.....	526		
50.1.1 依赖性.....	526		
50.1.2 多种反调试技术.....	526		
50.2 反调试破解技术.....	526		
50.3 反调试技术的分类.....	527		
50.3.1 静态反调试技术.....	528		

52.5	0xCC 探测	572	55.3	示例程序源代码	618
52.5.1	API 断点	573	55.4	调试练习	620
52.5.2	比较校验和	575	55.4.1	需要考虑的事项	620
第 53 章 高级反调试技术		577	55.4.2	JIT 调试	621
53.1	高级反调试技术	577	55.4.3	DebugMe2.exe	622
53.2	垃圾代码	577	55.5	小结	626
53.3	扰乱代码对齐	578	第 56 章 调试练习 3: PE 映像切换		627
53.4	加密/解密	581	56.1	PE 映像	627
53.4.1	简单的解码示例	581	56.2	PE 映像切换	628
53.4.2	复杂的解码示例	582	56.3	示例程序: Fake.exe、Real.exe、 DebugMe3.exe	628
53.4.3	特殊情况: 代码重组	584	56.4	调试 1	631
53.5	Stolen Bytes (Remove OEP)	584	56.4.1	Open - 输入运行参数	631
53.6	API 重定向	587	56.4.2	main() 函数	632
53.6.1	原代码	588	56.4.3	SubFunc_1()	634
53.6.2	API 重定向示例 #1	588	56.4.4	CreateProcess("fake.exe", CREATE_SUSPENDED)	635
53.6.3	API 重定向示例 #2	589	56.4.5	SubFunc_2()	635
53.7	Debug Blocker (Self Debugging)	593	56.4.6	SubFunc_3()	641
53.8	小结	595	56.4.7	ResumeThread()	644
第八部分 调试练习			56.5	调试 2	644
第 54 章 调试练习 1: 服务		598	56.5.1	思考	645
54.1	服务进程的工作原理	598	56.5.2	向 EP 设置无限循环	645
54.1.1	服务控制器	598	56.6	小结	647
54.1.2	服务启动过程	599	第 57 章 调试练习 4: Debug Blocker		648
54.2	DebugMe1.exe 示例讲解	600	57.1	Debug Blocker	648
54.2.1	安装服务	600	57.2	反调试特征	648
54.2.2	启动服务	602	57.2.1	父与子的关系	649
54.2.3	源代码	604	57.2.2	被调试进程不能再被其他 调试器调试	649
54.3	服务进程的调试	606	57.2.3	终止调试进程的同时也终 止被调试进程	649
54.3.1	问题在于 SCM	606	57.2.4	调试器操作被调试者的代 码	649
54.3.2	调试器无所不能	606	57.2.5	调试器处理被调试进程中 发生的异常	649
54.3.3	常用方法	606	57.3	调试练习: DebugMe4.exe	650
54.4	服务调试练习	606	57.4	第一次调试	650
54.4.1	直接调试: 强制设置 EIP	606	57.4.1	选定调试的起始位置	650
54.4.2	服务调试的常用方法: “附加”方式	609	57.4.2	main()	650
54.5	小结	615	57.5	第二次调试	651
第 55 章 调试练习 2: 自我创建		616	57.6	第三次调试	653
55.1	自我创建	616	57.7	第四次调试	656
55.2	工作原理	617			
55.2.1	创建子进程 (挂起模式)	617			
55.2.2	更改 EIP	618			
55.2.3	恢复主线程	618			