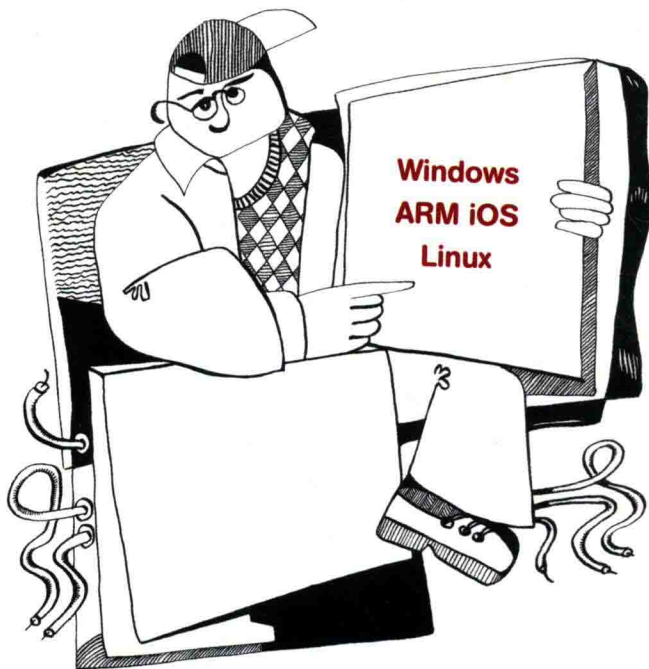


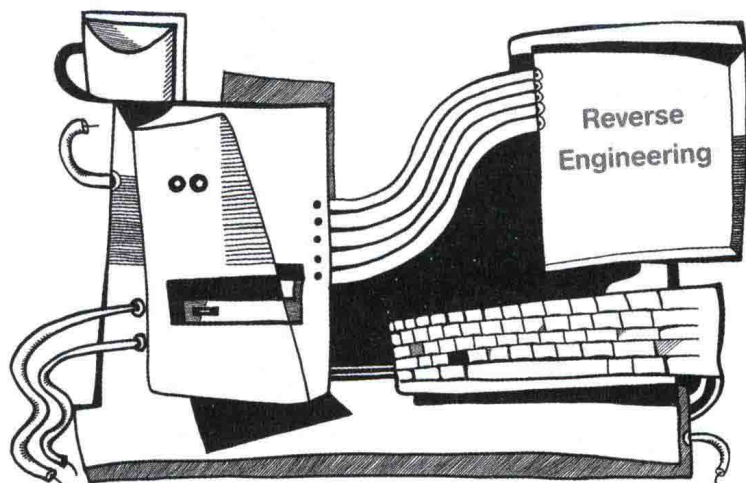
- 了解逆向工程的权威指南
- 初学者必备的大百科全书
- 安天网络安全工程师培训必读书目

Reverse Engineering for Beginners

逆向工程 权威指南 上册

[乌克兰] Dennis Yurichev◎著
Archer 安天安全研究与应急处理中心◎译





Reverse Engineering for Beginners

逆向工程 权威指南 上册

[乌克兰] Dennis Yurichev◎著

Archer 安天安全研究与应急处理中心◎译

人民邮电出版社

北京

图书在版编目 (C I P) 数据

逆向工程权威指南 / (乌克兰) 丹尼斯
(Dennis Yurichev) 著 ; Archer, 安天安全研究与应急
处理中心译. — 北京 : 人民邮电出版社, 2017. 4
ISBN 978-7-115-43445-6

I. ①逆… II. ①丹… ②A… ③安… III. ①工业产
品—计算机辅助设计—指南 IV. ①TB472-39

中国版本图书馆CIP数据核字(2016)第243654号

版权声明

Simplified Chinese translation copyright ©2017 by Posts and Telecommunications Press
ALL RIGHTS RESERVED

Reverse Engineering for Beginners, by Dennis Yurichev

Copyright © 2016 by Dennis Yurichev

本书中文简体版由作者 Dennis Yurichev 授权人民邮电出版社出版。未经出版者书面许可, 对本书的任何部分不得以任何方式或任何手段复制和传播。
版权所有, 侵权必究。

-
- ◆ 著 [乌克兰] Dennis Yurichev
 - 译 Archer 安天安全研究与应急处理中心
 - 责任编辑 陈冀康
 - 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>
 - 北京天宇星印刷厂印刷
 - ◆ 开本: 787×1092 1/16
 - 印张: 61.5 彩插: 1
 - 字数: 1 990 千字 2017 年 4 月第 1 版
 - 印数: 1-3 000 册 2017 年 4 月北京第 1 次印刷
- 著作权合同登记号 图字: 01-2014-3227 号
-

定价: 168.00 元(上、下册)

读者服务热线: (010) 81055410 印装质量热线: (010) 81055316

反盗版热线: (010) 81055315

广告经营许可证: 京东工商广字第 8052 号

内 容 提 要

逆向工程是一种分析目标系统的过程，旨在识别系统的各组件以及组件间关系，以便能够通过其他形式或在较高的抽象层次上，重建系统的表征。

本书专注于软件的逆向工程，是写给初学者的一本权威指南。全书共分为上、下两册，十二个部分，共 102 章，涉及 X86/X64、ARM/ARM-64、MIPS、Java/JVM 等重要内容，详细解析了 Oracle RDBMS、Itanium、软件狗、LD_PRELOAD、栈溢出、ELF、Win32 PE 文件格式、x86-64 (critical sections、syscalls、线程本地存储 TLS、地址无关代码 (PIC)、以配置文件为导向的优化、C++ STL、OpenMP、SHE 等众多技术问题，堪称是逆向工程技术百科全书。除了详细讲解，本书还给出了很多习题来帮助读者巩固所学的知识，附录部分给出了习题的解答。

本书适合对逆向工程技术、操作系统底层技术、程序分析技术感兴趣的读者阅读，也适合专业的程序开发人员参考。

序 从逆向视角透视代码大厦的那些砖石

相对于本书的中文名字《逆向工程权威指南》，我更喜欢它的英文原名《Reverse Engineering for Beginners》，可以直译为“逆向工程入门者必读”。在国外的程序员圈中，这本书被简称为《RE4B》，作者的 Blog 式连载已经引发了持续关注和好评。

本书的重要特色在于其清晰的章节结构，涵盖了函数、语句、结构体、数组等这些构成代码大厦的基本元素，如显微镜般逐一对比了这些元素在 X86、ARM 和 MIPS 体系架构下的“切片”影像。本书几乎每个章节都按照 X86、ARM、MIPS 三个体系架构分别展开，可以让读者深入对比理解在三种体系架构下，同样的指令、函数及数据结构呈现的异同和特点。无论对分析工程师，还是编码工程师来说，本书都是能更深入理解代码大厦原材料的工作指南。这是基础性、系统性的文献工作，也是令人耳目一新的结构安排。对于在中国传统高校计算机教育中，通过简单的 X86 ASM 来学习和了解体系结构的工程师来说，这也是尽快打通 ARM、MIPS 知识背景的“全栈”指南。对于那些从各种破解教程来切入逆向领域的安全爱好者来说，这更像是一份“正餐”。对于作者来说，我相信这个写作过程是充满激情的，但也充满了艰难和痛苦。对于 Dennis 这样的逆向工程专家来说，这本书的写作过程，更多的不是在探索未知的世界，也不是对高级技巧和经验的总结，而是要对看起来相对枯燥的单元式资料进行整理，将一些对其已经是常识的东西转化为系统而书面化（并易于读者理解）的语言。

逆向工程一直被宣传为一种充满乐趣和带有神秘主义的技能，而逆向分析者的工作看起来更有乐趣的原因，似乎就是在指令奔涌中逆流而上，绕开种种限制和保护，找到代码中的“宝藏”——错误、漏洞或者是被加密算法层层掩盖的数据结构。这个由媒体所打造的形象，也为部分逆向爱好者所自矜。而这也导致逆向领域的书籍和教程，更多地围绕着一些高阶的技巧展开，更多地讲解如何绕过加密和保护，却忘记了逆向工程的初衷是要洞察系统及软件的整体结构和机理。由于之前的逆向领域相关出版物中缺少“代码大全”式的基础读本，本书的完成填补了这一空白，然而，这不仅需要作者有高超的逆向研究能力，也要有正向系统的思维和视野。

本书的译者之一 Archer 提出想邀请安天的工程师们一同完成本书的翻译时，我曾经觉得他是不是“疯了”。本书当时并无定稿的英文版本，只是 Dennis 持续发布的 Blog 式的连载。尽管网络连载已经引发了好评如潮，但其精彩篇章彼时还如散落在海底的珍珠，尚未串成珠链。以 Dennis 天马行空的写作风格和追求完美主义的性格，他必然会在未完成全部章节的期间，不断地增加体系结构的新热点（如 ARM64）和修补既有章节的内容，甚至会改动原有的样例代码，导致本书原版定稿遥遥无期。我觉得，与其说正在养病的 Archer 接手翻译这本书是一份艰难的任务，倒不如说，Dennis 写完本书根本就是一份不可完成的任务。

因此当接近千页的样书摆在我面前时，我被深深震撼了，那种感觉和我之前靠自己极为蹩脚的英语扫过的英文电子稿不同，我能想象 Archer 是如何一边咳嗽着、一边码字或者校对代码；以及安天 CERT 的同事们是如何在样本分析任务饱和的情况下挤出时间来一点点推动翻译进展的。当我在视频例会上向安天各地的部门负责人展示样书时，我能看到每个人的赞叹和兴奋，那种兴奋不啻于我们自己发布了一份最新的长篇分析报告。

二进制分析，是安全分析工程师，包括有志于投身安全领域的编码工程师的基本功底之一。硅谷一些安全人士都曾提及一个有趣的说法——华人的系统安全天赋。中国安全厂商和安全工作者，在系统安全领域和分析工作中，正在不断取得新的进步。在国际网络安全企业中，华人承担关键系统安全研究和分析工作的占比也很高，国际高校的一些华人研究者在新兴领域展开系统安全研究，同样取得了很大的

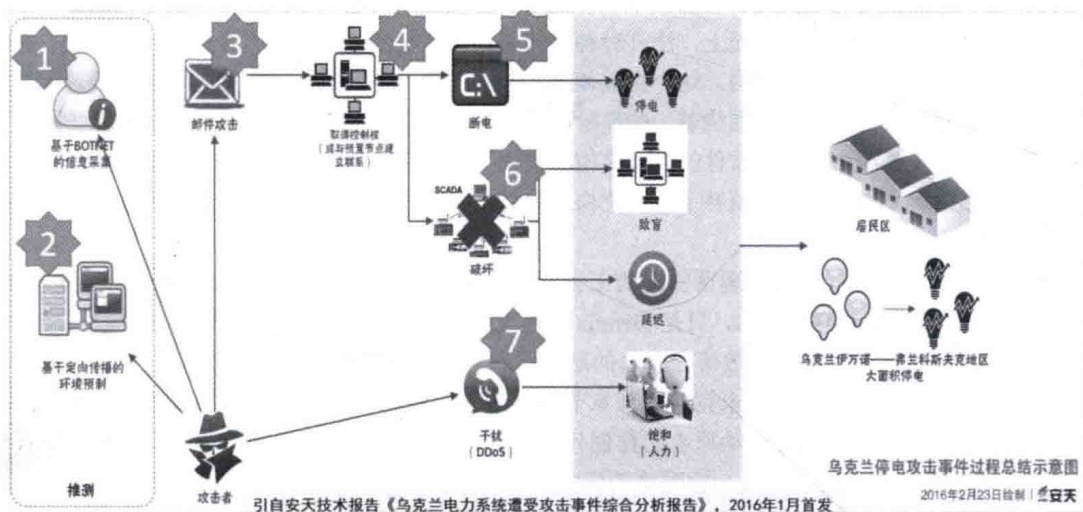
国际影响。遗憾的是，Web 和应用开发的兴起，一定程度上让更多年轻安全从业人员的目光转向 Web 和应用安全，当前新的安全从业者整体的二进制分析能力正在下降，存在着基本功缺失的问题，非常需要系统性的补课。

二进制分析能力，是对安全工程师的基础能力要求。但要对抗 APT 等新兴威胁，仅仅有基本功是不够的，还需要在更开阔的架构视野上，具备从攻击链、体系化等角度逆向分析研判攻击行为的全程、攻击对手的全貌的能力。

当年受到一些破解教程的误导，一些接触逆向工程的年轻人把破解序列号等当成逆向工程的目的，从而导致浅尝辄止，没有真正理解逆向工程的博大精深。即使是安天部分资深的分析工程师，也往往更乐于寻找解析攻击者的个性技巧和那些有“难度”的分析细节，并常以此感到自豪，而不能站到对攻击作业过程完整复盘和对攻击者画像的角度来看待问题，而后者才是一类高阶的逆向工程。

我们是从分析“震网”所遇到的系列挫折中，从对 Flame 蠕虫马拉松式的、收效甚微的模块分析中，认识到了传统分析工程师所面临的这种困境的。在“震网”分析中，由国际知名厂商的架构师和分析工程师所组成的混编分析团队，就能比安天完全由软硬件安全分析工程师组成的分析小组，取得更多的分析成果和进展。因为安全厂商和 APT 攻击之间，不是简单的查杀反制和单点的分析对抗，而是体系化对抗。从认识到这个问题开始，安天的分析团队一方面继续强化二进制分析的基本功底；另一方面自我批判那种“视野从入口点开始”的狭隘分析视角，避免我们自己从透析攻击者载荷的高级技巧中找到太多“快感”，而忘记那些更重要的全局性因素。

过去三年间，安天发布了针对 APT-TOCS、白象、方程式等组织相关的多篇分析报告，也分析了类似乌克兰国家电网遭遇攻击停电等与关键基础设施有关的安全事件。在乌克兰停电事件的分析中，我们已经初步走出了入口点视野，而尝试在更大的时空视角上还原事件。

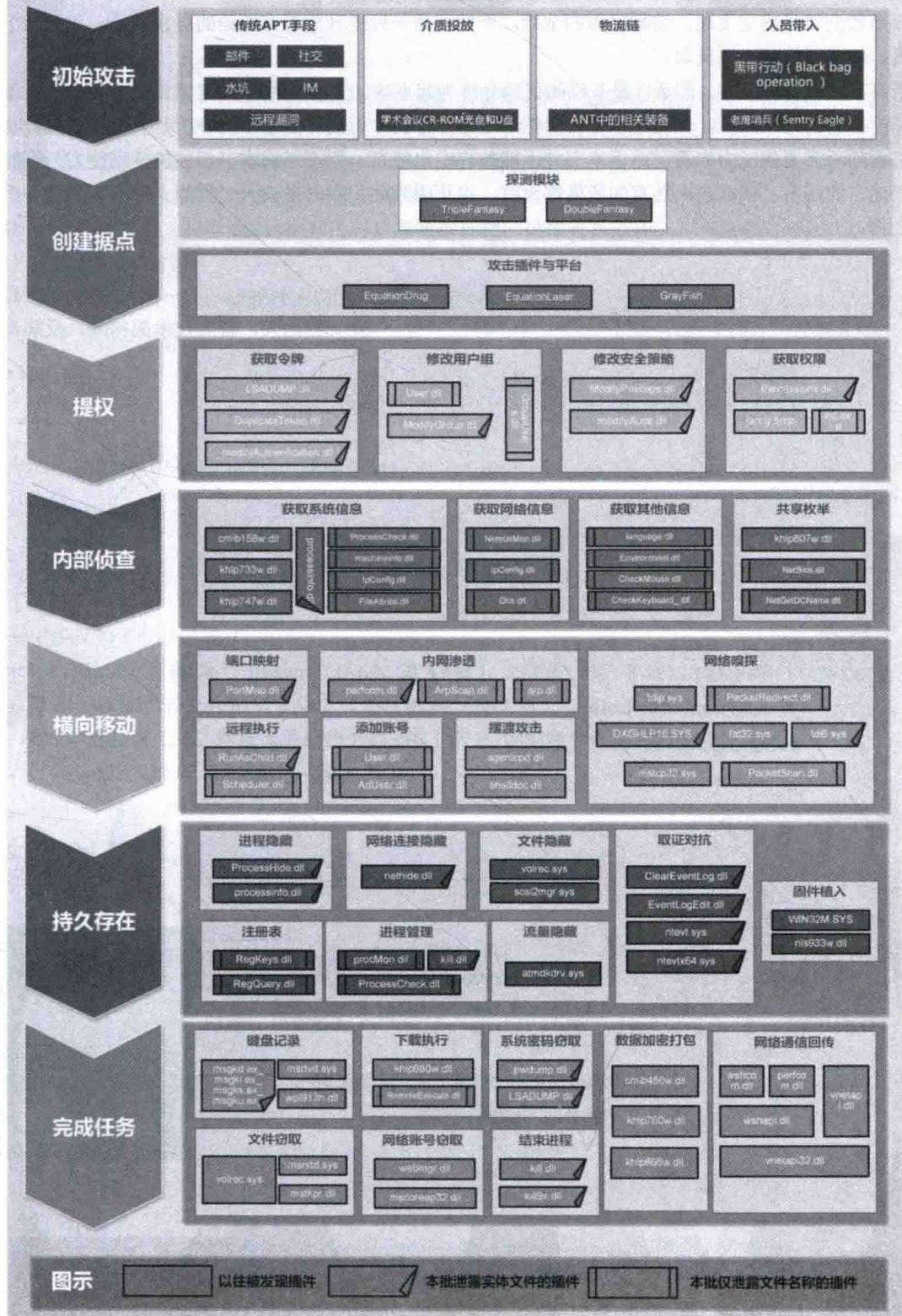


而即使从微观的分析来看，高阶恶意代码已经从一个附着在宿主上的代码片段或单体文件的木马，进化为一个庞大的工程体系。对于高级恶意代码来说，不仅有前端的持久化载荷和大量按需投放的“原子化”模块，其后更有一个庞大的支撑工程体系。对于前端，我们还能够依靠代码逆向来分析，而对于后面这个大到无形的“魅影”，我们更多的只能依靠分析恶意代码的指令体系、有限的网络感知捕获、少得可怜的可公开检索信息，以及作为信息安全动物的逻辑和本能来进行猜测。我们所要完成的工作，不仅仅是要逐一进行载荷模块分析，而是要对整个攻击面和攻击链的深入复盘。

2017年1月25日，安天分析小组更新了针对方程式攻击组织的第四篇分析报告《方程式组织 EQUATION DRUG 平台解析》，基于对方程式组织大量恶意代码模块的分析，我们初步完成了一份有价值的工作，那就是方程式组织的主机作业模块积木拼图。

方程式组织主机作业模块积木图

2017.1.24 安天根据分析和猜绘制



我的同事在《方程式组织 EQUATION DRUG 平台解析》中写道：

“这些庞杂的模块展开了一组拼图碎片，每一张图上都有有意义的图案，但如果逐一跟进，这些图案就会组成一个巨大的迷宫。迷宫之所以让人迷惑，不在于其处处是死路，而在于其看起来处处有出口，但所有的砖块都不能给进入者以足够的提示。此时，最大的期待，不只是有一只笔，可以在走过的地方做出标记，而是插上双翼凌空飞起，俯瞰迷宫的全貌，当然这是一种提升分析方法论的自我期待。我辈当下虽身无双翼，但或可练就一点灵犀。”

所有高阶的分析工作，都是以最基础的代码分析为起点的。无论是面对 APT 攻击的深度分析，还是对高级黑产行为的系统挖掘，仅有基本分析固然是远远不够的，但没有最基本的分析是万万不能的。从未来安全工程师所需要的能力来看，熟读本书并不能使你走出迷宫。但不掌握本书相关的基础能力，不能去透视代码迷宫的砖石，就连走入迷宫的资格都没有。也正因为此，本书是安天工程师人手一本的必备手册和必读之物。

肖新光

安天创始人，首席技术架构师，反病毒老兵

前言

“逆向工程”一词用在软件工程领域的具体含义历来模糊不清。逆向工程是一种分析目标系统的过程，旨在识别系统的各组件以及组件之间的关系，以便通过其他形式或在较高的抽象层次上，重建系统的表征。按照目标系统的分类划分，人们常说的“逆向工程”大体可分为：

- (1) 软件逆向工程。研究编译后的可执行程序。
- (2) 建模逆向工程。扫描 3D 结构并进行后续数据处理，以便重现原物。
- (3) 重建 DBMS 结构。

本书仅涉及上述第一项，即软件逆向工程的知识范畴。

重点议题

x86/x64、ARM/ARM64、MIPS、Java/JVM。

涉及话题

本书中涉及如下一些话题：

Oracle RDBMS (第 81 章)、Itanium (IA64, 第 93 章)、加密狗 (第 78 章)、LD_PRELOAD (67.2 节)、栈溢出、ELF、Win32 PE 文件格式 (68.2 节)、x86-64 (26.1 节)、critical sections (68.4 节)、syscalls (第 66 章)、线程本地存储 TLS, 地址无关代码 PIC (67.1 节)、以配置文件为导向的优化 (95.1 节)、C++ STL (51.4 节)、OpenMP (第 92 章)、SEH (68.3 节)。

作者简介



姓名: Dennis Yurichev

特长: 逆向工程及计算机编程

联系方式: E-mail [dennis\(a\)yurichev.com](mailto:dennis(a)yurichev.com)

Skype [dennis.yurichev](https://www.skype.com/en/contacts/yurichev)

译者简介

安天安全研究与应急处理中心 (安天 CERT), 是承担安天安全威胁应急处理、恶意代码分析、APT 攻

击分析取证等方面工作的综合研究和服务的部门，由资深安全工程师团队组成。安天 CERT 以“第一时间启动，同时应对两线严重威胁”为自身能力建设目标，在红色代码 II、口令蠕虫、震荡波、冲击波等重大安全疫情的响应中，提供了先发预警、深度分析和解决方案；并在 2010 年后针对震网、毒曲、白象、方程式等 APT 攻击组织和行动，进行了深入的跟踪分析。分析成果有效推动了安天核心引擎和产品能力的成长，获得了主管部门和用户的好评。

致谢

感谢耐心回答我提问的 Andrey “hermlt” Baranovich 和 Slava “Avid” Kazakov。

感谢帮助我勘误的 Stanislav “Beaver” Bobrytskyy、Alexander Lysenko、Shell Rocket、Zhu Ruijin 和 Changmin Heo。

感谢 Andrew Zubinski、Arnaud Patard (rtp on #debian-arm IRC) 和 Aliaksandr Autayeu 的鼎力支持。

感谢本书的中文版翻译 Archer 和安天安全研究与应急处理中心。

感谢本书的韩语版翻译 Byungho Min。

感谢校对人员 Alexander “Lstar” Chernenkiy、Vladimir Botov、Andrei Brazhuk、Mark “Logxen” Cooper、Yuan Jochen Kang、Mal Malakov、Lewis Porter 和 Jarle Thorsen。

特别感谢承担最多校对工作的，同时也是补救最多纰漏的朋友 Vasil Kolev。

感谢封面设计 Andy Nechaevsky。

感谢 github.com 的朋友，谢谢他们所发的各种资料以及勘误。

本书使用了 LATEX 的多种工具。在此，我向它们的作者表示敬意。

捐赠

希望鼓励作者继续创作的读者，通过下述网站进行捐赠：

[Dennis Yurichevdonate.html](http://Dennis.Yurichevdonate.html)

为了表示感谢，每位捐赠者都会获得题名。此外，捐赠者感兴趣的内容将会被优先更新或补充。

捐赠人名录

25 * anonymous, 2 * Oleg Vygovsky (50+100 UAH), Daniel Bilar (\$50), James Truscott (\$4.5), Luis Rocha (\$63), Joris van de Vis (\$127), Richard S Shultz (\$20), Jang Minchang (\$20), Shade Atlas (5 AUD), Yao Xiao (\$10), Pawel Szczur (40 CHF), Justin Simms (\$20), Shawn the R0ck (\$27), Ki Chan Ahn (\$50), Triop AB (100 SEK), AngeAlbertini (10+50 EUR), Sergey Lukianov (300 RUR), Ludvig Gislason (200 SEK), Gérard Labadie (40 EUR), Sergey Volchkov (10 AUD), Vankayala Vigneswararao (\$50), Philippe Teuwen (\$4), Martin Haeberli (\$10), Victor Cazacov (5 EUR), Tobias Sturzenegger (10 CHF), Sonny Thai (\$15), Bayna AlZaabi (\$75), Redfive B.V. (25 EUR), Joon Oskari Heikkilä (5 EUR), Marshall Bishop (\$50), Nicolas Werner (12 EUR), Jeremy Brown (\$100), Alexandre Borges (\$25), Vladimir Dikovski (50 EUR), Jiarui Hong (100.00 SEK), Jim_Di (500 RUR), Tan Vincent (\$30), Sri Harsha Kandrakota (10 AUD), Pillay Harish (10 SGD), Timur Valiev (230 RUR), Carlos Garcia Prado (10 EUR), Salikov Alexander (500 RUR), Oliver Whitehouse (30 GBP), Katy Moe (\$14), Maxim Dyakonov (\$3), Sebastian Aguilera (20 EUR), Hans-Martin Münch (15 EUR), Jarle Thorsen (100 NOK), Vitaly Osipov (\$100)。

FAQ

Q: 学习汇编语言有何用武之地?

A: 除去开发操作系统的研发人员之外, 现在几乎没有什么人还要用汇编语言编写程序了。目前, 编译程序优化汇编指令的水平已经超越编程人员的脑算水平^①, 而且 CPU 越来越复杂——即使一个人具备丰富的汇编语言的知识, 也不代表他有多么了解计算机硬件。但是不可否认的是, 汇编语言的知识至少有两用处: 首先, 它有助于安全人员进行安全研究、分析恶意软件; 其次, 它还有助于帮助编程人员调试程序。本书旨在帮助人们理解汇编语言, 而不是要指导读者用汇编语言进行编程。所以, 作者组织了大量的源代码和对应的汇编指令, 供读者研究。

Q: 这本书太厚了, 有没有精简版?

A: 精简版可从网上下载: <http://beginners.re/#lite>。

Q: 逆向工程方面的就业情况如何?

A: 在 reddit 等著名网站里, 很多著名公司一直在招聘熟悉汇编语言和逆向工程的 IT 专家, 甚至专门招聘逆向工程领域的安全专家。有兴趣的读者可以访问以下网址:

<http://www.reddit.com/r/ReverseEngineering/>

http://www.reddit.com/r/netsec/comments/221xxu/rnetsecs_q2_2014_information_security_hiring

Q: 我想要提些问题……

A: 作者的邮件地址是: [dennis\(a\)yurichev.com](mailto:dennis(a)yurichev.com)。

读者还可以在我們的网站 forum.yurichev.com 参与互动。

读者点评

- “构思精巧……而且免费……确实不错”——Daniel Bilar, Siege Technologies, LLC。
- “……了不起的免费读物!”——Pete Finnigan, Oracle RDBMS security guru。
- “……引人入胜, 值得一读!”——Michael Sikorski, 《Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software》的作者。
- “……谨向这本出色的教材致以个人的敬意!”——Herbert Bos, 阿姆斯特丹自由大学教授, 《Modern Operating Systems (4th Edition)》作者之一。
- “……令人惊讶、难以置信”。——Luis Rocha, CISSP / ISSAP, 技术经理, Verizon 业务部网络及信息安全中心。
- “感谢如此辛劳的作者、感谢如此精彩的书。”——Joris van de Vis, SAP 集成平台及安全专员。
- “……部分内容可圈可点。”——Mike Stay, 教员, 联邦执法训练中心, Georgia, US。
- “这本书超赞! 我的数名学生都在学习这本书, 我计划把它当作研究生教材。”——Sergey Bratus, 计算机科学系研究助理教授, (美) 达特茅斯学院。
- “Dennis Yurichev 发表了一本逆向工程方面的宝典 (还免费)!” Tanel Poder, Oracle RDBMS 性能调控专家。
- “这本书可谓初学者的百科全书……”——Archer, 中文译者, IT 安全研究员。

^① 可查阅参考文献[Fog13b]。

目 录

第一部分 指令讲解

第 1 章 CPU 简介	3
1.1 指令集架构	3
第 2 章 最简函数	5
2.1 x86	5
2.2 ARM	5
2.3 MIPS	5
MIPS 指令集与寄存器名称	6
第 3 章 Hello, world!	7
3.1 x86	7
3.1.1 MSVC	7
3.1.2 GCC	9
3.1.3 GCC:AT&T 语体	9
3.2 x86-64	11
3.2.1 MSVC-x86-64	11
3.2.2 GCC-x86-64	12
3.3 GCC 的其他特性	12
3.4 ARM	13
3.4.1 Keil 6/2013 —— 未启用优化功 能的 ARM 模式	14
3.4.2 Thumb 模式下、未开启优化选项 的 Keil	15
3.4.3 ARM 模式下、开启优化选项的 Xcode	15
3.4.4 Thumb-2 模式下、开启优化选项 的 Xcode (LLVM)	16
3.4.5 ARM64	18
3.5 MIPS	19
3.5.1 全局指针 Global pointer	19
3.5.2 Optimizing GCC	19
3.5.3 Non-optimizing GCC	21
3.5.4 栈帧	23
3.5.5 Optimizing GCC: GDB 的分 析方法	23
3.6 总结	24
3.7 练习题	24
3.7.1 题目 1	24
3.7.2 题目 2	24

第 4 章 函数序言和函数尾声	25
递归调用	25
第 5 章 栈	26
5.1 为什么栈会逆增长	26
5.2 栈的用途	27
5.2.1 保存函数结束时的返回地址	27
5.2.2 参数传递	28
5.2.3 存储局部变量	29
5.2.4 x86:alloca()函数	29
5.2.5 (Windows) SEH 结构化 异常处理	31
5.2.6 缓冲区溢出保护	31
5.3 典型的栈的内存存储格式	31
5.4 栈的噪音	31
5.5 练习题	34
5.5.1 题目 1	34
5.5.2 题目 2	34
第 6 章 printf()函数与参数调用	36
6.1 x86	36
6.1.1 x86: 传递 3 个参数	36
6.1.2 x64: 传递 9 个参数	41
6.2 ARM	44
6.2.1 ARM 模式下传递 3 个参数	44
6.2.2 ARM 模式下传递 8 个参数	46
6.3 MIPS	50
6.3.1 传递 3 个参数	50
6.3.2 传递 9 个参数	52
6.4 总结	56
6.5 其他	57
第 7 章 scanf()	58
7.1 演示案例	58
7.1.1 指针简介	58
7.1.2 x86	58
7.1.3 MSVC + OllyDbg	60
7.1.4 x64	62
7.1.5 ARM	63
7.1.6 MIPS	64
7.2 全局变量	65

7.2.1	MSVC: x86	66	10.1	全局变量	96
7.2.2	MSVC: x86+OllyDbg	67	10.2	局部变量	98
7.2.3	GCC: x86	68	10.3	总结	100
7.2.4	MSVC: x64	68	第 11 章 GOTO 语句		101
7.2.5	ARM: Optimizing Keil 6/2013 (Thumb 模式)	69	11.1	无用代码 Dead Code	102
7.2.6	ARM64	70	11.2	练习题	102
7.2.7	MIPS	70	第 12 章 条件转移指令		103
7.3	scanf()函数的状态监测	74	12.1	数值比较	103
7.3.1	MSVC: x86	74	12.1.1	x86	103
7.3.2	MSVC: x86: IDA	75	12.1.2	ARM	109
7.3.3	MSVC: x86+OllyDbg	77	12.1.3	MIPS	112
7.3.4	MSVC: x86+Hiew	78	12.2	计算绝对值	115
7.3.5	MSVC:x64	79	12.2.1	Optimizing MSVC	115
7.3.6	ARM	80	12.2.2	Optimizing Keil 6/2013: Thumb mode	116
7.3.7	MIPS	81	12.2.3	Optimizing Keil 6/2013: ARM mode	116
7.3.8	练习题	82	12.2.4	Non-optimizng GCC 4.9 (ARM64)	116
7.4	练习题	82	12.2.5	MIPS	117
	题目 1	82	12.2.6	不使用转移指令	117
第 8 章 参数获取		83	12.3	条件运算符	117
8.1	x86	83	12.3.1	x86	117
8.1.1	MSVC	83	12.3.2	ARM	118
8.1.2	MSVC+OllyDbg	84	12.3.3	ARM64	119
8.1.3	GCC	84	12.3.4	MIPS	119
8.2	x64	85	12.3.5	使用 if/else 替代条件运算符	120
8.2.1	MSVC	85	12.3.6	总结	120
8.2.2	GCC	86	12.4	比较最大值和最小值	120
8.2.3	GCC: uint64_t 型参数	87	12.4.1	32 位	120
8.3	ARM	88	12.4.2	64 位	123
8.3.1	Non-optimizing Keil 6/2013 (ARM mode)	88	12.4.3	MIPS	125
8.3.2	Optimizing Keil 6/2013 (ARM mode)	89	12.5	总结	125
8.3.3	Optimizing Keil 6/2013 (Thumb mode)	89	12.5.1	x86	125
8.3.4	ARM64	89	12.5.2	ARM	125
8.4	MIPS	91	12.5.3	MIPS	126
第 9 章 返回值		93	12.5.4	无分支指令 (非条件指令)	126
9.1	void 型函数的返回值	93	12.6	练习题	127
9.2	函数返回值不被调用的情况	94	第 13 章 switch()/case/default		128
9.3	返回值为结构体型数据	94	13.1	case 陈述式较少的情况	128
第 10 章 指针		96	13.1.1	x86	128

13.1.2	ARM: Optimizing Keil 6/2013 (ARM mode)	133	14.4.1	题目 1	165
13.1.3	ARM: Optimizing Keil 6/2013 (Thumb mode)	133	14.4.2	题目 2	165
13.1.4	ARM64: Non-optimizing GCC (Linaro) 4.9	134	14.4.3	题目 3	166
13.1.5	ARM64: Optimizing GCC (Linaro) 4.9	134	14.4.4	题目 4	167
13.1.6	MIPS	135	第 15 章 C 语言字符串的函数		170
13.1.7	总结	136	15.1	strlen()	170
13.2	case 陈述式较多的情况	136	15.1.1	x86	170
13.2.1	x86	136	15.1.2	ARM	174
13.2.2	ARM: Optimizing Keil 6/2013 (ARM mode)	140	15.1.3	MIPS	177
13.2.3	ARM: Optimizing Keil 6/2013 (Thumb mode)	141	15.2	练习题	178
13.2.4	MIPS	143	15.2.1	题目 1	178
13.2.5	总结	144	第 16 章 数学计算指令的替换		181
13.3	case 从句多对一的情况	145	16.1	乘法	181
13.3.1	MSVC	145	16.1.1	替换为加法运算	181
13.3.2	GCC	147	16.1.2	替换为位移运算	181
13.3.3	ARM64: Optimizing GCC 4.9.1	147	16.1.3	替换为位移、加减法的 混合运算	182
13.4	Fall-through	149	16.2	除法运算	186
13.4.1	MSVC x86	149	16.2.1	替换为位移运算	186
13.4.2	ARM64	150	16.3	练习题	186
13.5	练习题	151	16.3.1	题目 1	186
13.5.1	题目 1	151	第 17 章 FPU		188
第 14 章 循环		152	17.1	IEEE 754	188
14.1	举例说明	152	17.2	x86	188
14.1.1	x86	152	17.3	ARM、MIPD、x86/x64 SIMD	188
14.1.2	x86: OllyDbg	155	17.4	C/C++	188
14.1.3	x86: 跟踪调试工具 tracer	156	17.5	举例说明	189
14.1.4	ARM	157	17.5.1	x86	189
14.1.5	MIPS	160	17.5.2	ARM: Optimizing Xcode 4.6.3 (LLVM) (ARM mode)	193
14.1.6	其他	161	17.5.3	ARM: Optimizing Keil 6/2013 (Thumb mode)	193
14.2	内存块复制	161	17.5.4	ARM64: Optimizing GCC (Linaro) 4.9	194
14.2.1	编译结果	161	17.5.5	ARM64: Non-optimizing GCC (Linaro) 4.9	195
14.2.2	编译为 ARM 模式的 程序	162	17.5.6	MIPS	195
14.2.3	MIPS	163	17.6	利用参数传递浮点型数据	196
14.2.4	矢量化技术	164	17.6.1	x86	196
14.3	总结	164	17.6.2	ARM + Non-optimizing Xcode 4.6.3 (LLVM) (Thumb-2 mode)	197
14.4	练习题	165			

17.6.3	ARM + Non-optimizing Keil 6/2013 (ARM mode).....	198	18.7.4	总结.....	255
17.6.4	ARM64 + Optimizing GCC (Linaro) 4.9.....	198	18.8	本章小结.....	255
17.6.5	MIPS	199	18.9	练习题.....	255
17.7	比较说明.....	200	18.9.1	题目 1.....	255
17.7.1	x86.....	200	18.9.2	题目 2.....	258
17.7.2	ARM.....	216	18.9.3	题目 3.....	263
17.7.3	ARM64.....	219	18.9.4	题目 4.....	264
17.7.4	MIPS	220	18.9.5	题目 5.....	265
17.8	栈、计算器及逆波兰表示法.....	221	第 19 章 位操作		270
17.9	x64.....	221	19.1	特定位.....	270
17.10	练习题.....	221	19.1.1	x86.....	270
17.10.1	题目 1.....	221	19.1.2	ARM.....	272
17.10.2	题目 2.....	221	19.2	设置/清除特定位.....	274
第 18 章 数组		223	19.2.1	x86.....	274
18.1	简介.....	223	19.2.2	ARM + Optimizing Keil 6/2013 (ARM mode).....	277
18.1.1	x86.....	223	19.2.3	ARM + Optimizing Keil 6/2013 (Thumb mode).....	278
18.1.2	ARM.....	225	19.2.4	ARM + Optimizing Xcode (LLVM)+ ARM mode	278
18.1.3	MIPS	228	19.2.5	ARM: BIC 指令详解	278
18.2	缓冲区溢出.....	229	19.2.6	ARM64: Optimizing GCC(Linaro) 4.9	278
18.2.1	读取数组边界以外的内容.....	229	19.2.7	ARM64: Non-optimizing GCC (Linaro) 4.9	279
18.2.2	向数组边界之外的地址赋值.....	231	19.2.8	MIPS	279
18.3	缓冲区溢出的保护方法.....	234	19.3	位移.....	279
18.3.1	Optimizing Xcode 4.6.3 (LLVM) (Thumb-2 mode).....	236	19.4	在 FPU 上设置特定位.....	279
18.4	其他.....	238	19.4.1	XOR 操作详解	280
18.5	字符串指针.....	238	19.4.2	x86.....	280
18.5.1	x64.....	239	19.4.3	MIPS	282
18.5.2	32 位 MSVC.....	239	19.4.4	ARM.....	282
18.5.3	32 位 ARM.....	240	19.5	位校验.....	284
18.5.4	ARM64.....	241	19.5.1	x86.....	286
18.5.5	MIPS	242	19.5.2	x64.....	289
18.5.6	数组溢出.....	242	19.5.3	ARM + Optimizing Xcode 4.6.3 (LLVM) + ARM mode	291
18.6	多维数组.....	245	19.5.4	ARM + Optimizing Xcode 4.6.3 (LLVM) + Thumb-2 mode	292
18.6.1	二维数组举例.....	246	19.5.5	ARM64 + Optimizing GCC 4.9.....	292
18.6.2	以一维数组的方式访问 二维数组.....	247	19.5.6	ARM64 + Non-optimizing GCC 4.9.....	292
18.6.3	三维数组.....	248	19.5.7	MIPS	293
18.6.4	更多案例.....	251			
18.7	二维字符串数组的封装格式.....	251			
18.7.1	32 位 ARM.....	253			
18.7.2	ARM64.....	254			
18.7.3	MIPS	254			

19.6 本章小结	295
19.6.1 检测特定定位 (编译阶段)	295
19.6.2 检测特定定位 (runtime 阶段)	295
19.6.3 设置特定定位 (编译阶段)	296
19.6.4 设置特定定位 (runtime 阶段)	296
19.6.5 清除特定定位 (编译阶段)	296
19.6.6 清除特定定位 (runtime 阶段)	297
19.7 练习题	297
19.7.1 题目 1	297
19.7.2 题目 2	298
19.7.3 题目 3	301
19.7.4 题目 4	301

第 20 章 线性同余法与伪随机函数

20.1 x86	304
20.2 x64	305
20.3 32 位 ARM	306
20.4 MIPS	306
MIPS 的重新定位	307
20.5 本例的线程安全改进版	309

第 21 章 结 构 体

21.1 MSVC: systemtime	310
21.1.1 OllyDbg	311
21.1.2 以数组替代结构体	312
21.2 用 malloc() 分配结构体的空间	313
21.3 UNIX: struct tm	315
21.3.1 Linux	315
21.3.2 ARM	317
21.3.3 MIPS	319
21.3.4 数组替代法	320
21.3.5 替换为 32 位 words	322
21.3.6 替换为字节型数组	323
21.4 结构体的字段封装	325
21.4.1 x86	325
21.4.2 ARM	329
21.4.3 MIPS	330
21.4.4 其他	331
21.5 结构体的嵌套	331
OllyDbg	332
21.6 结构体中的位操作	333
21.6.1 CPUID	333
21.6.2 用结构体构建浮点数	337
21.7 练习题	339
21.7.1 题目 1	339

21.7.2 题目 2	340
-------------	-----

第 22 章 共用体 (union) 类型

22.1 伪随机数生成程序	345
22.1.1 x86	346
22.1.2 MIPS	347
22.1.3 ARM (ARM mode)	348
22.2 计算机器精度	349
22.2.1 x86	350
22.2.2 ARM64	350
22.2.3 MIPS	351
22.2.4 本章小结	351
22.3 快速平方根计算	351

第 23 章 函数指针

23.1 MSVC	353
23.1.1 MSVC+OllyDbg	354
23.1.2 MSVC+tracer	355
23.1.3 MSVC+tracer (指令分析)	356
23.2 GCC	357
23.2.1 GCC+GDB (有源代码的情况)	358
23.2.2 GCC+GDB (没有源代码的情况)	359

第 24 章 32 位系统处理 64 位数据

24.1 64 位返回值	362
24.1.1 x86	362
24.1.2 ARM	362
24.1.3 MIPS	362
24.2 参数传递及加减运算	363
24.2.1 x86	363
24.2.2 ARM	365
24.2.3 MIPS	365
24.3 乘法和除法运算	366
24.3.1 x86	367
24.3.2 ARM	368
24.3.3 MIPS	369
24.4 右移	370
24.4.1 x86	370
24.4.2 ARM	371
24.4.3 MIPS	371
24.5 32 位数据转换为 64 位数据	371
24.5.1 x86	372
24.5.2 ARM	372

24.5.3 MIPS	372	31.4 双模二元数据格式	416
第 25 章 SIMD	373	31.5 转换字节序	416
25.1 矢量化	373	第 32 章 内存布局	417
25.1.1 用于加法计算	374	第 33 章 CPU	418
25.1.2 用于内存复制	379	33.1 分支预测	418
25.2 SIMD 实现 strlen()	383	33.2 数据相关性	418
第 26 章 64 位平台	387	第 34 章 哈希函数	419
26.1 x86-64	387	单向函数与不可逆算法	419
26.2 ARM	394	第三部分 一些高级的例子	
26.3 浮点数	394	第 35 章 温度转换	423
第 27 章 SIMD 与浮点数的并行运算	395	35.1 整数值	423
27.1 样板程序	395	35.1.1 x86 构架下 MSVC 2012 优化	423
27.1.1 x64	395	35.1.2 x64 构架下的 MSVC 2012	
27.1.2 x86	396	优化	425
27.2 传递浮点型参数	399	35.2 浮点数运算	425
27.3 浮点数之间的比较	400	第 36 章 斐波拉契数列	428
27.3.1 x64	400	36.1 例子 1	428
27.3.2 x86	401	36.2 例子 2	430
27.4 机器精	402	36.3 总结	433
27.5 伪随机数生成程序(续)	402	第 37 章 CRC32 计算的例子	434
27.6 总结	403	第 38 章 网络地址计算实例	437
第 28 章 ARM 指令详解	404	38.1 计算网络地址函数 calc_network_	
28.1 立即数标识(#)	404	address()	438
28.2 变址寻址	404	38.2 函数 form_IP()	439
28.3 常量赋值	405	38.3 函数 print_as_IP()	440
28.3.1 32 位 ARM	405	38.4 form_netmask()函数和 set_bit()函数	442
28.3.2 ARM64	405	38.5 总结	442
28.4 重定位	406	第 39 章 循环: 几个迭代	444
第 29 章 MIPS 的特点	409	39.1 三个迭代器	444
29.1 加载常量	409	39.2 两个迭代器	445
29.2 阅读推荐	409	39.3 Intel C++ 2011 实例	446
第二部分 硬件基础		第 40 章 达夫装置	449
第 30 章 有符号数的表示方法	413	第 41 章 除以 9	452
第 31 章 字节序	415	41.1 x86	452
31.1 大端字节序	415		
31.2 小端字节序	415		
31.3 举例说明	415		