经济管理学术文库·其他类

软件逆向工程技术与应用

Technology and Application of Software Reverse Engineering

丁松阳／著

经济管理出版社
ECONOMY & MANAGEMENT PUBLISHING HOUSE

本书的出版受河南省教育厅科学技术研究重点项目（编号2A52(
应用漏洞发现技术研究”资助

经济管理学术文库·其他类

软件逆向工程技术与应用

Technology and Application of Software Reverse Engineering

丁松阳／著

经济管理出版社
ECONOMY & MANAGEMENT PUBLISHING HOUSE

图书在版编目（CIP）数据

软件逆向工程技术与应用/丁松阳著. —北京：经济管理出版社，2013.11
ISBN 978-7-5096-2827-0

Ⅰ.①软… Ⅱ.①丁… Ⅲ.①软件工程 Ⅳ.①TP311.5

中国版本图书馆 CIP 数据核字（2013）第 276701 号

组稿编辑：杨 雪
责任编辑：张 艳 赵喜勤
责任印制：杨国强
责任校对：超 凡 王纪慧

出版发行：经济管理出版社
（北京市海淀区北蜂窝 8 号中雅大厦 A 座 11 层 100038）
网 址：www.E-mp.com.cn
电 话：（010）51915602
印 刷：北京京华虎彩印刷有限公司
经 销：新华书店
开 本：720mm×1000mm/16
印 张：15.25
字 数：234 千字
版 次：2013 年 11 月第 1 版 2013 年 11 月第 1 次印刷
书 号：ISBN 978-7-5096-2827-0
定 价：45.00 元

前　言

软件逆向工程涉及的领域很广，初次涉入该领域的实践者往往感觉到无从下手。作者在涉入该领域的过程中，不仅遇到了工程实践中的许多问题，而且发现该领域还与计算机理论方面的一些基础问题密切相关。在不断的实践与摸索过程中，作者深切地体会到了从繁杂的资料中理出头绪的艰辛，所以也就萌生了一个想法：根据几年来在软件逆向工程方面积累的知识及实践经验，写一本既有一定的理论基础又有一定的实践应用的著作。

虽然逆向工程涉及的领域较广，但其核心问题是程序的静态分析。为了兼顾工程实践性，本书选择从 IA－64 下的反编译问题入手，逐步论述软件逆向工程的理论及工程实践。

本书共分为 15 章：

第 1 章介绍了软件逆向工程的基本概念及方法，接着基于抽象解释对于反编译过程进行了形式化描述，构建了基于抽象解释的反编译形式化框架。

第 2 章介绍了逆向开源工具 UQBT 设计的结构及如何利用该工具构建特定的软件逆向工具。

第 3 章对 IA－64 的体系结构及其特点进行了介绍。

第 4 章介绍了 ELF64 文件的读取及解析，并对 strip 处理后的二进制文件 main 函数的定位给出了解决方案。

第 5 章从反汇编的算法设计出发，讲解了如何利用自动反汇编构造工具 NJMCT 构造反汇编器。

第 6 章描述了语义描述语言 SSL 的文法设计以及为了适应 IA－64 体系结构而对 SSL 文法的扩展。

第 7 章分析了中间语言表示 RTL 及 HRTL，根据 IA－64 体系结构特点提出了 IA－64 指令的语义提升模型，并针对浮点指令语义问题设计了使用高级

语言模拟的解决方法。

第 8 章基于中间表示提出了谓词消除技术，并论述了相关算法，实现消除谓词后控制流图的优化。

第 9 章对数据流分析模型进行了归纳介绍，分析了数据流分析算法对反编译结果精确性的影响，从理论角度讨论了对反编译结果安全近似算法应具有的特征。

第 10 章介绍中间表示转换为低级 C 语言代码的过程及基本算法。

第 11 章首先建立基于抽象解释的间接跳转模型，运用结合数据流分析进行控制流分析的方法来解决间接跳转问题，并从抽象解释的角度证明了结合数据流分析构建控制流图算法最优解的存在性。最后给出了间接跳转及 C 语言 switch 语句的恢复算法。

第 12 章首先探讨了基于指令语义的数据类型恢复，接着论述了利用数据流分析技术来重命名变量的算法，并利用类型格理论推导变量的数据类型。

第 13 章针对间接调用、动态地址解析、回调函数等反编译中较难解决的问题，提出了地址映射算法，给出了工程上的解决方案。

第 14 章首先简要论述了流图的构建并介绍了控制流恢复的一些基本概念，论述了将括号定理应用到高级控制流恢复当中的方法，进而对各种高级控制流结构进行了详细的分析，给出了高级语言控制结构的识别算法。

第 15 章依据第 14 章分析所得到的高级控制流结构信息，对循环结构、TWO－WAY 结构、N－WAY 结构及 ONE－WAY 结构的代码输出做了详细的解析。

本书实现的 IA－64 反编译器与作者博士生期间研究的项目密切相关。在本书的写作过程中得到了项目组成员苏铭、齐宁、杨克峤、孙维新、崔平飞、朱晓珺等的支持，在此表示诚挚的感谢。

由于作者水平有限，书中难免存在缺点与不足之处，殷切期望各位读者能够批评指正，提出宝贵意见。

目 录

第1章　软件逆向工程概论

1.1　软件逆向工程概述

1.1.1　软件逆向工程概念

对于逆向工程比较一致的认识是：通过分析设备、物件或系统的结构、功能及操作而获取其技术原理的过程。逆向工程是一个技术过程，通过对目标对象进行分析，从而得到目标对象的处理流程、结构、规格等设计和技术原理。

逆向工程的动机比较复杂，以下是维基百科中列出的逆向工程的一些动机：

- 接口设计。由于互操作性，逆向工程被用来找出系统之间的协作协议。
- 军事或商业机密。窃取敌人或竞争对手的最新研究或产品原型。
- 改善文档。当原有的文档有不充分处，或当系统被更新而原设计人员不在时，逆向工程被用来获取所需数据，以补充说明或了解系统的最新状态。
- 软件升级或更新。出于功能、合规、安全等需求的更改，逆向工程被用来了解现有或遗留软件系统，以评估更新或移植系统所需的工作。
- 制造没有许可/未授权的副本。

- 学术/学习目的。
- 去除复制保护和伪装的登录权限。
- 文件丢失：采取逆向工程的情况往往是在某一个特殊设备的文件已经丢失（或者根本就没有），同时又找不到工程的负责人。新的系统时常需要基于陈旧的系统进行再设计，这就意味着想要集成原有的功能进行项目迭代的唯一方法便是采用逆向工程的方法分析已有的碎片进行再设计。
- 产品分析：用于调查产品的运作方式，部件构成；估计预算，识别潜在的侵权行为。

本书关注的重点是软件逆向工程技术，也就是通过分析软件系统，识别系统中各个组件的关系，然后以较高的抽象层次表征原系统。一种情况是软件的源代码可以获得，但缺乏软件的文档描述，通过逆向分析的办法，获得对系统的较高抽象层次的理解。这种情况属于软件工程过程的逆向。如通过分析源代码，获得程序的 UML 图等。另一种情况是软件只有可执行的二进制代码，通过逆向分析，实现软件的不同结构的移植或用较高层次语言重建源代码。

对于软件逆向工程的合法性不准备详细地探讨，但美国的法律规定软件逆向工程只要遵守版权法的合理使用规定，都是受保护的。如开源文字处理软件 OpenOffice 在微软公开 Office 文档格式规范之前就支持微软的 Office 文档，通过软件逆向工程的方法取得 Office 文档格式的规范是受法律保护的。另一个比较有名的逆向工程项目是 Samba，通过软件逆向工程的办法，获得 Windows 系统文件共享的规范，允许非 Windows 系统使用该文件共享规范、共享文件。

1.1.2 软件逆向工程的应用

软件逆向工程可应用于信息安全领域，其中包括恶意代码的识别、数字版权管理、二进制代码的漏洞发掘等。软件逆向工程还可以应用于软件开发领域，如软件的移植、遗产代码的理解、未公开的软件接口与协议的猜解等。

软件逆向工程的技术手段较多，但核心技术就是代码的静态分析技术。

具体的技术过程如反汇编、反编译等都属于代码的静态分析。但由于静态分析的一些局限性，近年来也不断有研究者提出结合动态的方法来提高静态分析结果的方法，但总体来看还不太成熟。本书关注于软件开发领域的逆向工程，研究的重点是反编译与二进制翻译。当然，研究的部分成果也可应用于信息安全领域。

1.2　反编译与二进制翻译

反编译与二进制翻译是软件逆向工程的具体应用，两者之间既有区别又有联系。反编译是从二进制可执行文件逆向出高级语言代码如 C 语言、JAVA 等。二进制翻译是将某个指令体系结构下的二进制代码移植到另一体系结构之上。二进制翻译有多种技术手段，其中一种技术就是先利用反编译将二进制代码逆向为 C 代码，再将 C 代码在目标机上编译，得到目标机上的可执行代码。

1.2.1　二进制翻译

从 IT 产业的发展过程来看，研究和开发新的计算机体系结构必须要有成熟的软件支持，只有这样才能使得新的产品得到推广与使用。在此需求的带动下，相应产生了二进制代码移植技术。二进制代码移植使用的主流技术是二进制翻译，通过使用该技术将原有指令集体系结构上的应用软件移植到新的体系结构之上，从而促进产品在新体系结构的推广和应用。

二进制翻译技术作为计算机软件逆向工程技术的重要分支，在国际上已经得到了广泛的重视和深入的研究。早在 1987 年 HP 公司就开发了一个二进制翻译系统，该系统将老的 HP3000 的程序移植到新的 PA－RISC 体系结构之上。该公司于 1991 年又开发了一个将 CISC 体系结构的可执行程序移植到 RISC 体系结构的静态翻译器，该翻译器采用了解释器作为补充的方式来处理间接跳转。从 1992 年开始，DEC 公司开发了一系列的二进制翻译工具，其目

的是吸引和促进用户向 Alpha 体系结构转移，这其中包括 VAX/VMS、MIPS/Unix、Sparc/Unix 以及 X86/NT 到 Alpha 目标机的翻译工具。Apple 公司于 1994 年在 PowerMac 上实现了一个 Motorola 68000 的解释器，后来又经过不断开发将其改进为翻译器。IBM 公司于 1996 年开发了一个调度 PowerPC 代码到超长指令字（VLIW）处理器的翻译系统，1999 年开发的 BOA 系统采用动态翻译技术，翻译了 PowerPC 的全部指令系统。1999 年 Queensland 大学开发了 UQBT 系统，它采用对机器指令和操作系统属性进行描述的技术方案，是一个可变源、可变目标的二进制翻译器框架。由于使用了灵活的前端与后端处理模块，为不同体系结构的二进制翻译提供了大量的可重用代码，该系统可以作为一个翻译器的生成器。Walkabout 系统是 UQBT 的动态翻译版本，该系统是一个动态二进制翻译器，其中动态部分由 SUN 公司开发完成。Transmeta 公司于 2000 年发布了 Transmeta 处理器芯片和相应的代码翻译软件，实现了在完全不同的硬件基础上翻译运行 X86 处理器的可执行代码。

1.2.2 反编译与二进制翻译关系

反编译与二进制翻译都是逆向工程的一种形式，两种技术的处理对象均为二进制可执行程序，二进制翻译以目标机上的可执行文件作为输出，而反编译以高级语言程序作为输出。

根据实现方法的不同，二进制翻译技术可以分为动态翻译和静态翻译两种类型。动态二进制翻译器对输入的二进制文件做动态分析，在运行时对执行程序片断进行动态翻译。动态翻译可以获取程序运行时的动态信息，利用这些信息可进行运行时优化。但由于程序受制于动态执行的一些限制，有些时候生成的代码效率较差。静态二进制翻译采取的是离线翻译形式，它并不执行被翻译的程序，而是使用程序静态分析技术来处理二进制文件。它的主要优点是可产生优化的代码，因此执行效率相对比较高，但对于间接跳转及间接过程调用的处理比较困难，难以处理自修改代码。

图 1－1 是目前较为通用的静态二进制翻译结构。整个翻译过程分为三个部分：前端、分析和后端。前端部分包括三个模块：二进制文件装载、机器

指令解码及生成中间表示语言代码。中间部分进行代码分析，去除与机器相关部分，生成高级程序结构。后端部分产生目标机器上可执行的程序代码。

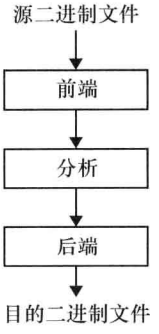

图 1－1　静态二进制翻译结构

本书论述的软件逆向工程的应用使用了图 1－1 所示的静态二进制翻译结构。该系统在不同阶段使用了两种中间表示语言：寄存器传输列表（RTL）和高级寄存器传输列表（HRTL）。其翻译的基本过程是：首先对二进制文件进行解码，生成与机器相关的中间表示 RTL，在低级中间表示上进行数据流、控制流等相关分析，消除机器相关部分，并恢复高级程序结构，得到高级中间表示 HRTL；在后端部分，首先将 HRTL 转换为 C 语言程序，接着在目标机上对生成的 C 程序进行编译、链接，最终完成翻译，生成目标机上的可执行二进制文件。

使用 C 语言作为后端输出，其最大的优点就是可变目标性，由于将源可执行文件提升为 C 程序，理论上来说，目标机可以是任何体系结构，但实践中在编译、链接产生目标机可执行文件时要考虑一些数据链接问题。由翻译结构流程可看出，项目的重点工作就是 C 程序的反编译问题，这也是本书研究分析的重点。本书研究内容不仅可用于静态二进制翻译工具的设计与实现，也可用于反编译工具的构建。

1.3 反编译过程的各个阶段

软件的分析可分为与系统机器相关的分析和独立于系统机器的分析。但在分析过程中，这两种分析之间可能存在着交叉。反编译的分析过程往往是两种分析不断循环重复的过程。如图 1 -2 所示，常见的逆向分析分为文件装载、指令解码、语义映射、相关图构造、过程分析、类型分析、结果输出七个阶段。但逆向分析的过程并不是一个严格的直线过程，而是存在着一些并行性，并且可能需要循环执行分析过程。

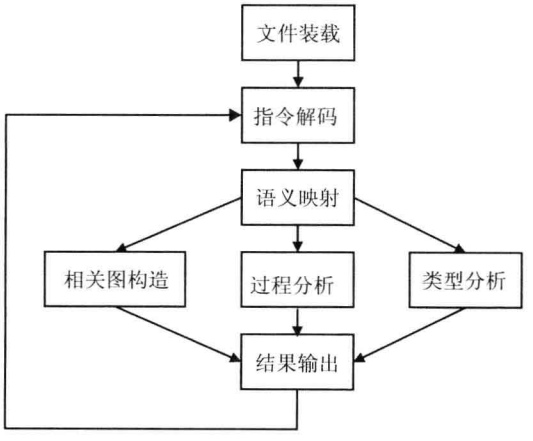

图 1 -2 反编译的各个阶段

文件装载。程序装载阶段主要完成读入目标文件，并进行与目标文件相关的一些初步分析，包括文件格式解析、文件信息搜集、文件性质判定等。几乎所有的操作系统都定义了其使用的可执行文件格式，常见格式有 PE（Portable Executable）格式和 ELF（Executable and Linkable Format）格式。PE 可执行文件格式主要应用在 Windows 系列的操作系统之上，由 Microsoft 公司定义。而 ELF 格式则是 Linux 操作系统上的可执行文件格式。通过文件装载阶段的操作，可以分析出文件执行的入口地址，初步分析文件的数据段和代

码段信息，以及文件运行所依赖的其他文件信息等。

指令解码。指令解码阶段的主要工作是根据目标体系结构的指令编码规则，对目标文件中使用的指令进行识别和翻译。可以将指令解码阶段的工作看作是一个反汇编器，但与常见的反汇编器并不完全等价。根据逆向分析的目的和手段，可以将目标指令映射为汇编指令，也可以映射为某种中间表示形式。使用中间表示的方式有两个优点：一是使用中间表示形式表示的目标程序更加易懂，而且易于进行自动化分析；二是在编译理论中积累的许多优秀算法可以直接应用到以中间表示形式表示的目标程序分析过程中。

语义映射。语义映射就是将二进制指令的执行效果，通过语义描述的方法表示出来，并加以记录。现在通常使用语义描述语言来描述二进制指令的语义。使用语义描述语言描述指令语义是借助一种专门用于描述指令语义的手段，然后在运行过程中动态加载所需体系结构的描述驱动文件，构成指令与语义描述之间的映射关系，从而在分析过程中将二进制指令序列映射为中间表示语句的序列。这种方法的优点是对于新的体系结构无须重新构建软件系统，只需根据需要，增加对应的指令语义描述代码即可。

相关图构造。反编译系统要借助于编译理论中的一些经典算法来完成程序的一些分析工作，其中主要包括图论在程序分析中的应用。常见的程序分析图包括控制流图（Control Flow Graph，CFG）、调用图（Call Graph，CG）、依赖图（Dependence Graph，DG）等。在完成相关分析的基础上便可以进行诸如控制流分析、数据流分析、依赖分析等操作，从而进一步对程序进行切片等高级操作。

过程分析。经过编译器翻译后的程序大多是面向过程式的，即使对于面向对象语言生成的可执行程序来说，编译器依然通过相关技术将其翻译为过程式代码。过程分析阶段的主要目标就是将目标文件中的过程信息恢复过来，包括过程边界分析、过程名（可能并不存在）、参数列表和返回值信息。过程边界信息可以通过相关过程调用指令和返回指令的信息得到，有时也需要借助一些特殊的系统库函数调用（例如 exit 函数）。对于某些经过优化的程序，可能会将调用或返回指令直接编译为跳转指令。这也是自动分析手段需要解决的难点之一。对于过程名来说，由于大多数过程名与程序运行无关，

因此经过优化的程序可能会删除目标文件中与过程名相关的信息。参数和返回值分析则依赖于程序变量的定义—引用信息完成。

类型分析。类型分析阶段的目标在于正确反映原程序中各个存储单元（包括寄存器和内存）所携带的类型信息。该分析主要有两种方式：基于指令语义的方式和基于数据流分析的方式。基于指令语义的方式根据具体指令的执行方式完成类型定义及转换操作，这种方式实现起来比较简单，但无法反映程序指令上下文之间的联系。基于数据流的方式将所有的数据类型进行概括和归纳，并且制订相应的类型推导规则来分析数据类型。

结果输出。结果输出是反编译系统的最终阶段。该阶段将反编译分析阶段生成的使用高级中间表示语言表示的程序，根据转换目标，转化为高级语言的形式（通常为 C 语言）有效地呈现在操作人员面前。

1.4 反编译技术相关研究

尽管早在 20 世纪 60 年代就出现了第一个反编译器，但反编译相关理论研究及成果还是非常有限的，特别是针对二进制可执行程序的反编译来说，现在还不存在无须人工干预就能够反编译所有二进制可执行程序到高级语言并能再次正确编译的工具。但在不同领域，针对反编译的研究一直没有间断过。在 60 年代，反编译主要用来辅助完成第二代计算机上的软件向第三代计算机移植。70 年代到 80 年代，研究主要集中在二进制程序的修改、遗产代码的维护等方面。90 年代后，反编译被用于静态二进制翻译、代码验证等方面，特别是近年来反编译技术还被应用到软件恶意代码检测领域。

1.4.1 早期的反编译

机器代码的反编译已经有较长的历史了。早在 1960 年，D - Neliac 反编译器可由机器码产生 Neliac 语言代码（类似于 Algol 语言）。在早期的反编译中考虑的问题主要有字长度、负整数和浮点数的表示，自修改代码及副作用