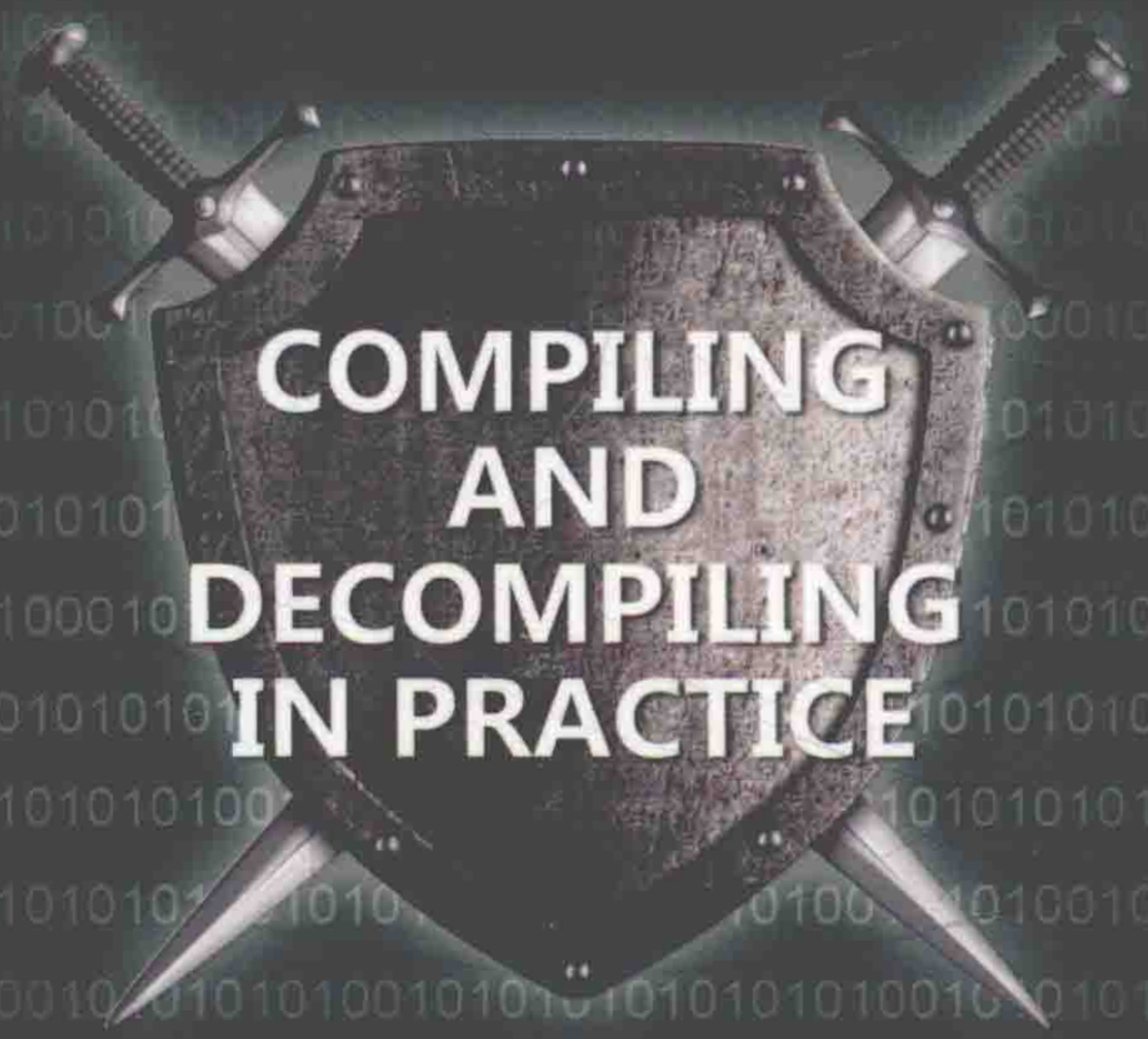
HZ BOOKS
华章教育

信息安全
技术丛书

系统介绍编译与反编译理论、技术与工具
作者多年科研、工程、教学经验的总结与融合

编译与反编译技术实战

庞建民 主编
刘晓楠 陶红伟 岳峰 戴超 编著



机械工业出版社
China Machine Press

信息安全
技术丛书

编译与反编译技术实战

庞建民 主编
刘晓楠 陶红伟 岳峰 戴超 编著

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

编译与反编译技术实战 / 庞建民主编 . —北京：机械工业出版社，2017.5
（信息安全技术丛书）

ISBN 978-7-111-56617-5

I. 编… II. 庞… III. 计算机网络－安全技术 IV. TP393.08

中国版本图书馆 CIP 数据核字（2017）第 070557 号

本书从编译与反编译两个角度出发，按照以实践为主线、以理论为辅助、两者结合的原则展开讲述。在编译部分，从正向角度，按照编译过程的步骤，从词法分析器的设计与实现方面的实践入手，逐步按自上而下语法分析器和自下而上语法分析器的设计与实现等环节展开，而语义分析与处理实践、优化与目标代码生成实践等环节则在对 GCC 和 LLVM 等具体编译器剖析时展开论述；同时，结合信息安全问题对多样化编译实践进行论述；然后，从反向角度介绍反编译实现的相关原理与技术，包括反汇编器的设计与实现、控制流图恢复的实现、关键行为的恢复及其在恶意代码检测中的应用等。

本书可作为软件编程、信息与网络空间安全或者软件逆向分析工作的工程技术人员的参考书，也可作为计算机与信息安全相关专业高年级本科生或研究生的教学参考书。

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号　邮政编码：100037）
责任编辑：佘　洁　　责任校对：殷　虹
印　　刷：北京瑞德印刷有限公司　　版　　次：2017 年 5 月第 1 版第 1 次印刷
开　　本：186mm×240mm　1/16　　印　　张：23
书　　号：ISBN 978-7-111-56617-5　　定　　价：79.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991　88361066　　投稿热线：（010）88379604
购书热线：（010）68326294　88379649　68995259　　读者信箱：hzjsj@hzbook.com

Preface 前　言

“编译技术”是从事软件开发和信息安全相关工作的技术人员必须掌握的基础性技术，也是高等院校计算机科学与技术和软件专业的一门必修专业课，这是理论与实践结合非常强的领域，对提升开发人员的技术水平和大学生科学思维的养成、解决实际问题能力具有重要作用。“反编译技术”则是近几年发展起来的新兴技术，许多计算机软件或信息安全从业者非常关心该技术的发展，但目前这方面的书籍较少，与“编译技术”结合起来讲解的书也很少，从实践角度来剖析的更是少见。本书就是在这种需求以及作者在这两方面的科研实践的驱动下诞生的，目的是为计算机软件和信息安全从业者提供编译与反编译技术方面的知识和实战技巧。

本书的编写得到了解放军信息工程大学和机械工业出版社的大力支持，在此表示诚挚的谢意。本书中的一些材料来自本书主编主持的国家自然科学基金（项目编号：61472447）、国家“863”（项目编号：2006AA01Z408）、国家重大专项某子课题等项目的研究成果，在此对这些课题的支持表示衷心的感谢！

本书是机械工业出版社 2016 年 4 月出版的《编译与反编译技术》（ISBN 978-7-111-53412-9）一书的姊妹篇，配合学习和使用效果更佳。在本书中，作者着力阐述编译与反编译技术及实战方面的相关知识和实战技巧，力图使用通用的语言讲述抽象的原理、技术和实战技能，但限于作者水平，书中难免有错误与欠妥之处，恳请读者批评指正。

作者

2017 年 3 月

目　　录 *Contents*

第1章 Chapter 1

实践的环境与工具

本书致力于通过实践及案例，从正反向两个角度介绍编译系统的一般构造原理和基本实现技术，本章首先对书中内容涉及的环境与工具进行简单介绍，这些工具都是编译与反编译过程中常用的工具。

1.1 实践环境概述

在编译过程中所涉及的环境主要是编译环境及工具链，常用的工具有词法分析生成器、语法分析生成器、编译器、汇编器、链接器等。在反编译过程中主要涉及反汇编器、静态或动态的调试与分析工具。下面对近年来流行的编译与反编译工具逐一进行简单介绍。

1.2 词法分析生成器 LEX

词法分析是编译过程的第一个阶段，其任务就是将输入的各种符号转化成相应的标识符号，转化后的标识符很容易被后续阶段处理。

LEX 是 LEXical compiler 的缩写，是 UNIX 环境下非常著名的工具，主要功能是生成一个词法分析器的 C 源码，描述规则采用正则表达式。描述词法分析器的文件 *.l 经过 LEX 编译后生成一个 lex.yy.c 的文件，然后由 C 编译器编译生成一个词法分析器。

LEX 接收用户输入的正则表达式，识别这些表达式并且将输入流转化为匹配这些表达式的字符串。在这些字符串的分界处，用户提供的程序片段被执行。LEX 代码文件将正则表达式和程序片段关联，将每一条输入对应到由 LEX 生成的程序的表达式，且执行相应的

代码片段。

为了完成任务，除了需要提供匹配的表达式以外，用户还需要提供其他代码，甚至是由其他生成器产生的代码。用户提供一般程序设计语言的代码片段完成程序识别表达式后的工作。因此，用户自由编写动作代码时，并不影响其编写高级语言代码来匹配字符串表达式。这就避免迫使用户使用字符串语言来进行输入分析时，也必须使用同样的方法来编写字符处理程序，而这样做有时是不合适的。LEX 不是完整的语言，它是一个新语言的生成器，可以插入各种不同的被叫作“宿主语言”的程序设计语言中。就像大多数高级语言可以生成在不同计算机硬件上运行的代码一样，LEX 可以生成不同的宿主语言。宿主语言用于 LEX 生成输出代码，也用于用户插入程序片段，这使得 LEX 适用于不同的环境和不同的使用者。每一个应用程序可以是硬件、适用于该任务的宿主语言、用户背景和局部接口属性的直接结合。现在，LEX 唯一支持的宿主语言是 C，过去也支持过其他语言。Fortran LEX 自身一般运行于 UNIX 或 Linux 系统之上，但是 LEX 生成的代码可以在任何适当的编译器上使用。

LEX 将用户输入的表达式和动作代码转换为宿主语言，生成的函数一般名为 yylex。yylex 识别字符流中的表达式，并且当每一个表达式被检测出来后，输出相应的动作。

LEX 的文件结构简单，分为三个部分：

```
declarations
%%
translation rules
%%
auxiliary procedures
```

分别是声明段、规则段和辅助部分。

1）声明段包括变量、符号常量和正则表达式的声明。希望出现在目标 C 源码中的代码，用“%{…%}”括起来。比如：

```
%{
#include <stdio.h>
#include "y.tab.h"
typedef char * YYSTYPE;
char * yylval;
%}
```

2）规则段是由正则表达式和相应的动作组成的。如

```
[0-9]+              printf("Int     : %s/n",yytext);
[0-9]*/.[0-9]+      printf("Float   : %s/n",yytext);
```

“[0-9]+”是描述整数的正则表达式，“[0-9]*/.[0-9]+”是描述浮点的正则表达式，后面的 printf 即为它们对应的动作。

值得注意的是，LEX 依次尝试每一个规则，尽可能地匹配最长的输入流，即规则部分

具有优先级的概念。比如下面的规则部分

```
%%
A      {printf("run");}
AA     {printf("like ");}
AAAA   {printf("I ");}
%%
```

对于内容“AAAAAAA”，LEX 程序会输出“I like run”，首先匹配最长的 4 个“A”，之后在剩下的三个“A”中匹配两个“A”，直到最后的一个“A”。可以看出 LEX 的确按照最长的规则匹配。

3）辅助部分放一些扫描器的其他模块，可以包含用 C 语言编写的子程序，而这些子程序可以用在前面的动作中，这样就可以达到简化编程的目的。辅助部分也可以在另一个程序文件中编写，最后再链接到一起。生成 C 代码后，需用 C 的编译器编译，链接时需要指定链接库。

本书的第 3 章将更加详细地介绍 LEX 及其用法。需要说明的是，对于 GNU/Linux 用户，与 UNIX 环境中 LEX 对应的工具是 Flex，其具体用法和 LEX 相似，这里不再赘述。

1.3　语法分析生成器 YACC

语法分析的任务是分析句子是否符合语法规范。YACC（Yet Another Compiler Compiler）是一个经典的语法分析生成器。YACC 最初是由 AT&T 公司的 Steven C. Johnson 为 UNIX 操作系统开发的，后来一些兼容的程序如 Berkeley YACC、GNU Bison、MKS YACC 和 Abraxas YACC 陆续出现，它们都是在此基础上做了少许改进或者增强，但是基本概念是相同的。现在 YACC 也已普遍移植到 Windows 及其他平台上。

语法分析生成器是一个指定某个格式中的一种语言的语法作为它的输入，并为该语言产生分析过程以作为它的输出的程序。在历史上，语法分析生成器被称作编译 - 编译程序，这是由于按照规律可将所有的编译步骤作为包含在语法分析程序中的动作来执行。现在的观点是将语法分析程序仅考虑为编译处理的一个部分，所以这个术语也就有些过时了。YACC 迄今为止仍是常用的语法分析生成器之一。

作为 YACC 对说明文件中的“%token NUMBER”声明的对应，YACC 坚持定义所有的记号本身，而不是从别的地方引入一个定义，但是却有可能通过在记号声明中的记号名之后书写一个值来指定将赋给记号的数字值。

YACC 的输入是巴科斯范式（BNF）表达的语法规则以及语法归约的处理代码，YACC 输出的是基于表驱动的编译器，包含输入的语法归约的处理代码部分。

由于所产生的解析器需要词法分析器的配合，因此 YACC 经常和词法分析器的产生器（通常就是 LEX）联合使用，把两部分产生的 C 程序一并编译。IEEE POSIX P1003.2 标准

定义了 LEX 和 YACC 的功能和需求。

需要指出的是，本书的第 4 章还有对 YACC 及其用法更加详细的介绍。

1.4 编译器 GCC

GCC（GNU Compiler Collection，GNU 编译器套件）是由 GNU 开发的编程语言编译器。它是以 GPL 许可证发行的自由软件，也是 GNU 计划的关键部分。GCC 原本作为 GNU 操作系统的官方编译器，现已被大多数类 UNIX 操作系统（如 Linux、BSD、Mac OS X 等）采纳为标准的编译器，GCC 同样适用于微软的 Windows。

GCC 原名为 GNU C 语言编译器（GNU C Compiler），因为它原本只能处理 C 语言。随着 GCC 的快速扩展，其可支持 C++，后来又能够支持更多编程语言，如 Fortran、Pascal、Objective-C、Java、Ada、Go 以及各类处理器架构上的汇编语言等，所以改名为 GNU 编译器套件。

（1）前端接口

前端将高级语言源码经过词法分析、语法分析生成与高级语言无关的低级中间层表示 GENERIC，然后经过单一化赋值转化为另一种中间表示层 GIMPLE，在中间层 GIMPLE 组建控制流程图，并在 GIMPLE 上进行一系列优化。然后将其转换为更加便于优化的 RTL 中间表示层。有了与前端无关的中间表示，GCC 的前端将不同的高级编程语言转换成这种中间表示，这就是 GCC 处理器支持多种编程语言的根本原因。

（2）中间接口

中间接口主要在 RTL 中间表示上进行各种优化，GCC 的优化技巧根据版本不同而有很大不同，但都包含了标准的优化算法，如循环优化、公共子表达式删除、指令重排序等。更多的优化方法也在不断地研究中。

（3）后端接口

GCC 后端对每条 RTL 通过模板匹配的方法调用对应的汇编模板生成汇编代码。生成的代码因处理器和结构不同而不同，GCC 后端为不同的平台提供了描述指令的汇编模板文件，这样就可以实现对不同平台的支持。这个阶段非常复杂，因为必须要考虑 GCC 可移植平台的处理器指令集的规格与技术细节，解决指令选择和寄存器分配等问题。

（4）GCC 常用的参数

在使用 GCC 编译器的时候，必须给出一系列必要的调用参数和文件名称。GCC 编译器的调用参数大约有 100 多个，这里只介绍其中最基本、最常用的参数。具体细节内容可参考 GCC Manual。

GCC 最基本的用法是：

```
gcc [options] [filenames]
```

其中 options 就是编译器所需要的参数（也称为编译选项），filenames 给出相关的文件名称。

-c：只编译，不链接成为可执行文件，编译器只是由输入的 .c 等源代码文件生成以 .o 为后缀的目标文件，通常用于编译不包含主程序的子程序文件。

-o output_filename：确定输出文件的名称为 output_filename，同时这个名称不能与源文件同名。如果不给出这个选项，GCC 就给出预设的可执行文件 a.out。

-g：产生符号调试工具（GNU 的 gdb）所必要的符号信息，要想对源代码进行调试，必须加入这个选项。

-O：对程序进行优化编译、链接。采用这个选项，整个源代码会在编译、链接过程中进行优化处理，这样产生的可执行文件的执行效率可以提高，但是编译、链接的速度相应地要慢一些。

-O2：比 -O 更好的优化编译、链接，当然整个编译、链接过程会更慢。

-v：GCC 执行时执行的详细过程、GCC 及其相关程序的版本号。编译程序时加上该选项可以看到 GCC 搜索头文件 / 库文件时使用的搜索路径。

1.5　编译器 LLVM

LLVM 是构架编译器的框架系统，由 C++ 编写而成，用于优化以任意程序语言编写的程序的编译时间、链接时间、运行时间以及空闲时间，对开发者保持开放，并兼容已有脚本。LLVM 计划启动于 2000 年，最初由伊利诺伊大学香槟分校的 Chris Lattner 主持开展。2006 年 Chris Lattner 加盟 Apple 公司并致力于 LLVM 在 Apple 开发体系中的应用。Apple 公司也是 LLVM 计划的主要资助者。

LLVM 的命名最早源自于 Low Level Virtual Machine（底层虚拟机）的缩写，由于命名带来的混乱，目前 LLVM 就是该项目的全称。LLVM 核心库提供了与编译器相关的支持，可以作为多种语言编译器的后台来使用，能够进行程序语言的编译期优化、链接优化、在线编译优化、代码生成。LLVM 的项目是一个模块化和可重复使用的编译器和工具技术的集合。LLVM 提供一个现代化的、基于 SSA 的编译策略，能够同时支持静态和动态的任意编程语言的编译目标。至今为止，LLVM 已被应用到许多商业和开源的项目，并被广泛用于学术研究。

LLVM 荣获 2012 年 ACM 软件系统奖。

对关注编译技术的开发人员，LLVM 提供了很多优点：

1）现代化的设计。LLVM 的设计是高度模块化的，使得其代码更为清晰和便于排查问题所在。

2）语言无关的中间代码。一方面，这使得通过 LLVM 能够将不同的语言相互联结起来，也使得 LLVM 能够紧密地与 IDE 交互和集成。另一方面，发布中间代码而非目标代码能够在目标系统上更好地发挥其潜能而又不影响可调试性（比如，在目标系统上针对本机的硬件环境产生目标代码，但又能够直接通过中间代码来进行行级调试）。

3）可作为工具和函数库。使用 LLVM 提供的工具可以比较容易地实现新的编程语言的优化编译器或虚拟机，或为现有的编程语言引入一些更好的优化 / 调试特性。

1.6 反汇编工具 IDA

IDA 是 IDA PRO 的简称，是英文 Interactive Disassembler 的缩写。它是一个世界顶级的交互式反汇编工具，其使用者覆盖了软件安全专家、军事工业和国家安全信息部门、逆向工程学者、黑客等。它是由 HEX-RAYS 公司开发的，这是一家多年从事将二进制代码反编译到 C 的软件安全公司，其旗舰产品就是著名的 Hex-Rays.Decompiler（IDA 的插件）。

IDA 有两种可用版本。标准版支持 20 多种处理器，高级版支持 50 多种处理器。IDA 不存在任何注册机、注册码或破解版，除了测试版和一个 4.9 的免费版外，网络上能下载的都是包含用户许可证的正版，因为所有的安装包都是 OEM 版，所以 IDA 官网不提供软件下载，并且软件也没有注册的选项（完全可以正常使用，当然这也是一种盗版或侵权的行为，对此 IDA 公司会采取严厉打击措施）。

IDA 的界面比其他反汇编工具界面更加专业，它提供了很多选项并允许用户自己编写插件。它的优点是可以更好地反汇编和进行更深层的分析，而缺点是使用更困难。

1.7 反汇编工具 OllyICE

OllyICE 是一种具有可视化界面的 32 位汇编分析调试器，是一个动态追踪调试工具，利用 IDA 与 SoftICE 结合的思想在 Ring3 级上进行调试，容易上手，已代替 SoftICE 成为当今最流行的调试解密工具，同时还支持插件扩展功能，是目前最强大的动态调试工具。

1.8 仿真与分析工具 QEMU

QEMU 是一套由 Fabrice Bellard 所编写的以 GPL 许可证分发源码的模拟处理器，在 GNU/Linux 平台上使用广泛。Bochs、PearPC 等与其类似，但不具备其许多特性，比如高速度及跨平台的特性，通过 KQEMU 这个闭源的加速器，QEMU 能模拟至接近真实计算机的速度。

目前，0.9.1 及之前版本的 QEMU 可以使用 KQEMU 加速器。在 QEMU 1.0 之后的版本都无法使用 KQEMU，但可利用 qemu-kvm 加速模块，并且加速效果以及稳定性明显好于 KQEMU。

QEMU 有以下两种主要运作模式：

1）进程级模拟模式。在这种模式下，QEMU 能以二进制翻译的方式启动那些为不同处理器编译的 Linux 可执行程序。典型的程序是 Wine 及 Dosemu。

2）系统级模拟模式。QEMU 能模拟整个计算机系统，包括中央处理器及其他周边设备。它使得为跨平台编写的程序进行测试及除错工作变得容易。也能用来在一台主机上虚拟数台不同虚拟计算机。

该软件的优点有：

- 默认支持多种架构。可以模拟 IA 32、AMD 64、MIPS、SPARC、PowerPC、龙芯等多种架构。
- 可扩展，可自定义新的指令集。
- 开源，可移植，仿真速度快。
- 在支持硬件虚拟化的 x86 架构上可以使用 KVM 加速配合内核 ksm 大页面备份内存，速度稳定，远超过 VMware ESX。
- 可以在其他操作系统平台上运行 Linux 程序。
- 可以存储及还原运行状态。
- 可以支持虚拟网卡等多种外设。

该软件的缺点有：

- 对微软视窗及某些主机操作系统的支持不完善（某些模拟的系统仅能运行）。
- 对不常用的架构的支持不完善。
- 除非使用 KQEMU 加速器，否则其模拟速度仍不及其他虚拟软件，如 VMware。
- 比其他模拟软件难安装及使用。

1.9　动态分析工具 TEMU

TEMU 是动态分析工具 BitBlaze 的一个组件，是一个基于系统仿真器 QEMU 开发的动态二进制分析工具，以 QEMU 为基础运行一个完整的系统（包括操作系统和应用程序），并对二进制代码的执行进行跟踪和分析。

TEMU 提供以下功能：

1）动态污点分析。TEMU 能够对整个系统进行动态污点分析，把一些信息标记为污点（如键盘事件、网络输入、内存读写、函数调用、指令等），并在系统内进行污点传播。这个特性可为符号执行提供插件形式的工具。许多分析都需要对二进制代码进行细粒度的分析，而基于 QEMU 的全系统模拟器确保了细粒度的分析。

2）获取操作系统视图。操作系统中提取的信息如进程和文件对很多分析都是很重要的。TEMU 可以使用这些信息决定当前执行的是哪个进程和模块、调用的 API 和参数，以及文件的存取位置。全系统的视图使我们能够分析操作系统内核以及多个进程间的交互，而许多其他的二进制分析工具（如 Valgrind、DynamoRIO、Pin）只提供了一个局部的视图（如单个进程的信息）。这对于分析恶意代码更为重要，因为许多攻击涉及多个进程，而且诸如 rootkits 的内核攻击变得越来越普遍。

3）深度行为分析。TEMU 能够分析二进制文件和操作环境的交互，如 API 调用序列、边界内存位置的访问。通过标记输入为污点，TEMU 能够进行输入和输出之间的关系分析。并且，全系统仿真器有效地隔离了分析组件和待分析代码。因此，待分析代码更难干扰分析结果。

TEMU 由 C 和 C++ 实现。性能要求高的代码由 C 实现，而面向分析的代码由 C++ 编写，以便很好地利用 C++ STL 中的抽象数据类型和类型检查。

1.10 本章小结

本章简要介绍了阅读本书所需要的实践环境，主要有词法分析生成器 LEX、语法分析生成器 YACC、编译器 GCC 和 LLVM、反汇编工具 IDA 和 OllyICE、仿真与分析工具 QEMU、动态分析工具 TEMU 等。在第 1 章简单介绍这些工具，主要是希望读者在开始时就能够在自己的机器上安装这些工具，并能够使用这些工具进行一些简单的实验，某些重要的工具将在后面详细介绍。