信息科学技术学术著作丛书

可信编译构造理论与关键技术

何炎祥　吴　伟　著

科学出版社

信息科学技术学术著作丛书

可信编译构造理论与关键技术

何炎祥　吴　伟　著

科学出版社
北京

内 容 简 介

编译器是重要的系统软件，除了编译功能，在许多领域都有应用，如绿色计算、嵌入式系统优化、代码分析与验证、可信计算、软件测试等。编译器是否可信直接影响着整个计算机系统的可信性，因此编译器的可信性一直受到人们的广泛关注和深切重视。本书全面系统地介绍了可信编译理论及其关键技术，从保证编译器自身可信性和编译对象可信性两个方面进行了详细阐述。同时，本书还介绍了可信编译在嵌入式软件优化、绿色编译、软件测试以及可信软件开发过程中的应用与实践。

本书许多内容是作者近年来在该领域的最新研究成果，具有较强的原创性。本书取材新颖、内容丰富、概念准确、层次清晰、叙述严谨、图文并茂、系统性强，可作为高等院校和科研院所计算机科学与技术、软件工程、计算机应用技术等相关专业的高年级本科生或研究生的参考书，也可供可信软件及相关领域的研究人员借鉴、学习和参考。

图书在版编目(CIP)数据

可信编译构造理论与关键技术 / 何炎祥，吴伟著. —北京：科学出版社，2013

(信息科学技术学术著作丛书)

ISBN 978-7-03-036420-3

Ⅰ. 可… Ⅱ. ①何… ②吴… Ⅲ. 编译器-研究 Ⅳ. TP314

中国版本图书馆 CIP 数据核字(2012)第 316165 号

责任编辑：魏英杰 杨向萍 / 责任校对：钟 洋

责任印制：张 倩 / 封面设计：陈 敬

科 学 出 版 社出版

北京东黄城根北街 16 号

邮政编码：100717

http://www.sciencep.com

源海印刷有限责任公司 印刷

科学出版社发行 各地新华书店经销

*

2013 年 1 月第 一 版 开本：B5(720×1000)

2013 年 1 月第一次印刷 印张：15

字数：288 000

定价：60.00 元

(如有印装质量问题，我社负责调换)

《信息科学技术学术著作丛书》序

21世纪是信息科学技术发生深刻变革的时代，一场以网络科学、高性能计算和仿真、智能科学、计算思维为特征的信息科学革命正在兴起。信息科学技术正在逐步融入各个应用领域并与生物、纳米、认知等交织在一起，悄然改变着我们的生活方式。信息科学技术已经成为人类社会进步过程中发展最快、交叉渗透性最强、应用面最广的关键技术。

如何进一步推动我国信息科学技术的研究与发展；如何将信息技术发展的新理论、新方法与研究成果转化为社会发展的新动力；如何抓住信息技术深刻发展变革的机遇，提升我国自主创新和可持续发展的能力？这些问题的解答都离不开我国科技工作者和工程技术人员的求索和艰辛付出。为这些科技工作者和工程技术人员提供一个良好的出版环境和平台，将这些科技成就迅速转化为智力成果，将对我国信息科学技术的发展起到重要的推动作用。

《信息科学技术学术著作丛书》是科学出版社在广泛征求专家意见的基础上，经过长期考察、反复论证之后组织出版的。这套丛书旨在传播网络科学和未来网络技术，微电子、光电子和量子信息技术、超级计算机、软件和信息存储技术，数据知识化和基于知识处理的未来信息服务业，低成本信息化和用信息技术提升传统产业，智能与认知科学、生物信息学、社会信息学等前沿交叉科学，信息科学基础理论，信息安全等几个未来信息科学技术重点发展领域的优秀科研成果。丛书力争起点高、内容新、导向性强，具有一定的原创性；体现出科学出版社“高层次、高质量、高水平”的特色和“严肃、严密、严格”的优良作风。

希望这套丛书的出版，能为我国信息科学技术的发展、创新和突破带来一些启迪和帮助。同时，欢迎广大读者提出好的建议，以促进和完善丛书的出版工作。

中国工程院院士
原中国科学院计算技术研究所所长
李国杰

前　言

为了适应大自然，更好地生存下来，人类很早就学会了制造和使用工具。经历了石器时代、青铜时代、铁器时代、蒸汽时代、电气时代，到现在的信息时代，人类利用自己的智慧不断发明更加先进的生产工具。不管在哪个时代，人类对高质量生产工具的追求从未停止，通过改进工艺、使用新材料、反复实验等各种手段以期生产出质量更高的工具。在市场经济作为全球主导经济类型的今天，人类制造出的物品可以作为商品在市场上出售。由于市场的竞争性，人们对质量的要求更加突出。随着工具越来越复杂，功能越来越强大，可信性已成为质量需求中日益重要的一个方面。因为使用不可信的先进工具，人们无法准确预期它的行为进而无法掌控，这将会带来负面作用甚至灾难性后果。计算机作为当今社会最先进工具的象征，提高计算机系统的可信性是目前最迫切、最有现实意义、最具挑战性的研究课题之一。

早在20世纪30年代Babbage的论文《计算机器》中就出现了“可信计算”这一概念，在不同的发展阶段，其研究的内容和重点在不断演变。到目前为止，可信性这一概念还没有形成一个被广泛接受、良好形式化的定义，通常英文翻译包括Dependability、Trustworthiness、High Confidence。美国国家安全局的国家计算机安全中心颁布的官方标准——可信计算机系统评价标准第一次提出可信计算机的概念，并把其作为系统安全的基础。Laprie于1985年正式提出可信性以便与可靠性相区别。可信计算组对可信的定义是：一个实体在实现预定目标时，若其行为总是符合预期，则该实体是可信的。一个软件若是可信的，则必然是正确的，并且能够如预期那样工作。广义地讲，可信性的特征属性包括可用性、可靠性、防危性、安全性、可维护性，其中安全性又可进一步细分为机密性与完整性。

目前，许多安全攸关的系统是结合硬件和软件实现的。这些系统的可信性不仅仅取决于硬件和软件的可信性，还取决于连接硬件层和软件层的编译器的可信性。如果没有一个可信的编译器，任何程序编译后的目标代码都不可信，即使程序已经被验证是正确的。很长一段时间内，可信软件的研究大部分只是关注如何验证高级语言编写的程序可信性，但是高级语言编写的程序与实际运行的目标代码之间存在转换关系，如果转换过程不可信，那么对高级语言的形式化验证就毫无意义。可信编译的研究就是为了解决这一问题。

从20世纪60年代起，工业界和学术界对可信编译的关注就一直未中断过。许多研究者对可信编译的关键理论和实现方法做了大量研究。但是，构造一个商

品化的、优化的可信编译器仍然面临许多挑战。近年来，由于对计算机系统的可信性要求越来越高、程序语言理论的发展及程序验证技术的不断进步，编译器的可信性问题日益突出，可信编译研究逐渐成为当前的研究热点。本书从各个角度对可信编译理论及其关键技术进行了全方位的论述，同时介绍了可信编译器在许多相关领域的典型应用和实践。

本书共 8 章，各章的主要内容组织如下：

第 1 章为绪论，介绍了可信编译的研究背景、相关研究现状以及研究目标和内容。随后介绍了全书的组织结构。

第 2 章介绍了编译器自身可信性相关内容。对可信编译的概念进行了阐述和形式化定义，介绍了编译正确性概念及利用编译器正确性来保证编译器自身可信性的理论和方法。随后，对可信编译器构造方法进行了分析和归类，详细阐述了每类方法的基本原理和步骤。本章最后详细描述了基于形式语义证明编译器正确性的方法和编译优化正确性证明方法。

第 3 章对编译对象的可信性进行了介绍。保证编译对象，即编译器所编译的代码的可信性是可信编译器的最终目标。本章详细介绍了可信编译器采用的 3 种机制来提高编译对象的可信性。这 3 种机制分别是代码可信性验证机制、代码安全性加强机制以及可执行代码安全保护机制。

第 4 章介绍了可信编译在嵌入式软件优化中的应用。首先介绍了一种基于编译的优化框架。通过机器无关和机器相关两个方面概述了传统的编译器优化技术。然后介绍了我们在内存优化和功耗优化方面的最新研究成果。

第 5 章结合绿色计算概念，首先介绍了绿色编译优化相关概念及绿色编译优化框架。然后分别介绍了若干面向混合存储系统和面向总线翻转的绿色编译优化方法。最后介绍了一种绿色评估模型。

第 6 章介绍了可信编译在软件测试中的应用。首先介绍了一种基于编译的测试框架，在此基础上介绍了一种测试需求描述语言。然后详细描述了一种基于编译的错误可跟踪的自动测试方法。

第 7 章以可信软件开发为载体，介绍了可信编译在可信软件开发过程中的应用。首先介绍了基于编译的可信软件开发过程模型。然后详细阐述了基于编译的可信软件开发过程模型中的几个主要部分：可信软件需求建模与验证、可信软件的设计与实现以及可信软件的测试与评估。

第 8 章对本书的内容进行总结。讨论了可信编译理论及其应用有待解决的问题和面临的挑战，展望了可信编译研究的未来趋势和发展方向。

本书的内容是武汉大学计算机学院众多科研人员多年学习、研究和工程实践沉淀的成果。参与相关研究的人员包括刘陶、李清安、刘健博、徐超、胡明昊、董伟、廖希密、刘钱、陈念、吴昊、毋国庆、文卫东、吴黎兵、彭敏、李飞、邵凌霜、贾向阳、严

飞、余发江、李晓红等；参与写作的主要人员有吴伟、陈勇、李清安、刘健博、徐超等。何炎祥具体规划和设计了全书的内容并对全书进行了统稿，文卫东、吴黎兵、彭敏对本书初稿提出了很多建设性意见，在此对他们的积极参与和热心帮助表示衷心的感谢。

本书是国内第一本针对可信编译展开系统研究的学术著作，对相关领域的研究人员具有一定的借鉴意义和参考价值。本书的出版得到国家自然科学基金可信软件基础研究重大研究计划项目“可信编译理论与实现方法研究”(90818018)、国家自然科学基金可信软件基础研究重大研究计划项目“基于编译的高可信嵌入式软件开发与验证方法研究”(91018009)、国家自然科学基金“基于编译的嵌入式系统优化研究”(61170022)、国家自然科学基金可信软件基础研究重大研究计划重点项目“可信软件构造理论与方法研究”(9111800)，以及湖北省自然科学基金计划重点项目(2008CDA007)等项目的资助，在此一并表示感谢。

可信编译理论及其应用是当前处于科学前沿的课题之一，相关的理论和技术还在发展之中，许多新的思想、理论和方法还需要进一步完善和验证。由于作者的水平和经验有限，不妥之处在所难免，恳请读者批评指正，共同推进可信编译理论研究的进步和发展。

作　者

2012 年 6 月于武汉

目　录

《信息科学技术学术著作丛书》序
前言
第 1 章　绪论 …… 1
1.1　研究背景及意义 …… 1
1.2　相关研究现状 …… 2
1.2.1　编译器自身的安全性问题研究 …… 3
1.2.2　编译所生成程序代码的安全性问题研究 …… 3
1.2.3　安全程序设计语言及其相应编译系统构造方法研究 …… 5
1.3　目标及内容 …… 7
1.3.1　编译器自身的可信性研究 …… 8
1.3.2　编译所得可执行代码的可信性研究 …… 9
1.4　本书组织结构 …… 11
1.5　本章小结 …… 12
参考文献 …… 13
第 2 章　编译器自身可信性 …… 15
2.1　引言 …… 15
2.2　可信编译概念 …… 16
2.3　编译正确性证明 …… 17
2.3.1　编译器正确性 …… 18
2.3.2　形式语义与语义保持 …… 21
2.3.3　保证编译器自身可信性方法 …… 22
2.4　可信编译器构造 …… 26
2.4.1　基于编译器正确性的构造方法 …… 27
2.4.2　基于携带证明代码的构造方法 …… 29
2.4.3　基于转换检验的构造方法 …… 31
2.4.4　基于可证明微型编译器的构造方法 …… 33
2.4.5　可信编译器构造方法对比分析 …… 38
2.5　基于形式语义的编译器正确性证明 …… 39
2.5.1　源语言 …… 39
2.5.2　目标语言 …… 42

2.5.3 编译转换及正确性证明 …… 43
2.6 编译优化正确性证明 …… 49
2.6.1 中间语言语法和迁移系统 …… 50
2.6.2 模拟关系及语义一致性验证 …… 52
2.6.3 编译优化实现正确性验证过程 …… 52
2.6.4 应用示例 …… 55
2.6.5 结论 …… 56
2.7 本章小结 …… 57
参考文献 …… 57
第 3 章 编译对象的可信性 …… 63
3.1 引言 …… 63
3.2 代码可信性验证机制 …… 63
3.2.1 基于分离逻辑的程序验证 …… 64
3.2.2 基于模型检测的程序验证方法 …… 67
3.3 代码安全性加强机制 …… 80
3.3.1 基于有色 Petri 网的缓冲区溢出检测 …… 80
3.3.2 基于 SMT 求解器的程序验证方法 …… 86
3.4 可执行代码安全保护机制 …… 93
3.4.1 基于压缩的低开销代码保护策略 …… 93
3.4.2 基于程序流敏感的自修改代码混淆方法 …… 101
3.4.3 程序动态完整性保护 …… 109
3.5 本章小结 …… 118
参考文献 …… 119
第 4 章 基于编译嵌入式软件优化 …… 121
4.1 引言 …… 121
4.2 基于编译的优化框架 …… 122
4.2.1 基于编译的嵌入式系统优化框架结构设计 …… 122
4.2.2 面向嵌入式平台编译优化的中间语言设计 …… 123
4.2.3 目标机器特征描述语言研究 …… 124
4.3 传统的编译器优化技术概述 …… 124
4.3.1 机器无关的编译器优化技术 …… 124
4.3.2 机器相关的编译器优化技术 …… 128
4.4 内存优化 …… 128
4.4.1 背景 …… 129
4.4.2 分析 …… 131

4.4.3 方法 …… 132
4.4.4 一个例子 …… 136
4.4.5 实验结果 …… 136
4.5 功耗优化 …… 138
4.5.1 功耗模型 …… 139
4.5.2 基于编译的功耗优化方法 …… 141
4.5.3 基于总线的功耗优化实例 …… 143
4.5.4 基于片上缓存的功耗优化 …… 154
4.6 本章小结 …… 165
参考文献 …… 166
第5章 绿色编译优化 …… 168
5.1 引言 …… 168
5.2 绿色编译器概念及优化框架 …… 169
5.2.1 绿色编译器的概念 …… 169
5.2.2 绿色编译优化框架 …… 170
5.3 面向移动嵌入式系统的绿色编译优化方法研究 …… 172
5.3.1 基于计算博弈论的混合存储系统绿色编译优化 …… 174
5.3.2 基于总线翻转编码和多维度集成学习的指令选择和调度方法研究 …… 176
5.3.3 基于编译的多核环境下的并行程序绿色优化 …… 177
5.4 绿色评估模型 …… 181
5.5 本章小结 …… 183
参考文献 …… 183
第6章 基于编译的软件测试 …… 185
6.1 引言 …… 185
6.2 基于编译的测试框架 …… 186
6.3 测试需求描述语言 …… 188
6.4 基于编译的错误可跟踪的自动测试方法 …… 190
6.4.1 用于测试用例生成的扩展语法及对应语义 …… 192
6.4.2 应用举例 …… 194
6.5 本章小结 …… 197
参考文献 …… 197
第7章 基于编译的可信软件构造及关键技术 …… 200
7.1 引言 …… 200
7.2 基于编译的可信软件开发过程模型 …… 201
7.2.1 可信软件开发过程模型 …… 201

7.2.2 可信软件开发过程可信保障机制 …… 202
7.3 可信软件需求建模与验证 …… 205
7.3.1 基于软件行为的需求建模方法 …… 205
7.3.2 面向可信软件需求模型的验证方法 …… 206
7.4 可信软件的设计与实现 …… 208
7.4.1 可信软件体系结构设计 …… 208
7.4.2 体系结构驱动的构件化可信软件组装 …… 210
7.4.3 基于可信编译的代码生成与分析 …… 212
7.5 可信软件的测试与评估 …… 220
7.5.1 可信软件的测试 …… 220
7.5.2 可信软件的评估 …… 221
7.6 本章小结 …… 222
参考文献 …… 223
第 8 章 总结及展望 …… 226
8.1 总结 …… 226
8.2 展望 …… 227

第1章 绪 论

1.1 研究背景及意义

近年来随着软件规模的不断扩大，其内部结构越来越复杂、应用环境越来越开放，这些因素使得人们更加关注软件的可信性问题。软件的可信是系统稳定运行的关键，可信软件旨在保证和提高计算机系统的可信性。因此，在人们对计算系统极为依赖的今天，可信软件的研究对于信息安全、系统安全乃至整个国家和社会的安全都极为重要。

可信性是指正确性、可靠性及安全性。影响软件可信性的主要因素包括来自软件内部的代码缺陷、代码错误、程序故障及来自软件外部的病毒、恶意代码等。由此可知，软件可信性在很大程度上依赖于程序代码的可信性，因此从代码角度来保证软件的可信性是实现可信软件的重要途径之一。

众所周知，软件程序一般需要经过编译器编译后方能执行，如果我们能在编译过程中对上述各种问题进行防范，则可以大大提升软件的可信性程度。另一方面，如果编译器本身不可信，则无法保证所生成代码的可信性，因为非可信编译器在对程序源代码进行编译的过程中，很可能篡改其原本的语义，生成不安全的目标代码。例如，目前已提出的利用编译器的可信攻击(trusting trust)就是通过修改编译器，使之在编译过程中向重要程序中插入恶意代码，从而对整个系统实行攻击。目前各商用编译器的发展虽然已较为成熟，但是这些编译器本身仍存在许多问题。例如，Unix 上 C、C++的编译器 GNU C(GCC)一直以来都存在问题，以至于编译研究者和程序员专门成立了一个 Usenet 新闻组来讨论 GCC 中的缺陷。Sun 公司的 Java 开发包在发布他们的编译器时，就知道里面存在许多错误并开始准备发布后续的补丁。Borland 公司的 Delphi 可视化编程环境也一直存在许多错误。不管这些编译器存在的缺陷有没有导致严重的软件错误，它们都是一直存在的，因此由其编译生成的程序代码很难被认为是可信的，可以说目前编译器本身的安全隐患对软件的可信性已造成了严重威胁。

通过研究与分析，我们认为针对可信编译理论与实现方法的研究将在可信软件系统的构建中起到非常重要的作用。其原因可总结为如下几点：

① 编译器在整个计算机软件系统中所处的“位置”非常特别，几乎所有的可执行代码都是通过编译器编译后产生的。因此，如果能在编译器中加入一些有效的可信技术，它们会在编译的过程中自动地对代码进行可信处理，无需编程人员干

预。这将大大减轻软件开发者开发可信软件的工作量。

② 编译器的主要作用就是将高级语言程序转换成等价的可执行代码，在编译转换过程中，编译器会提取大量的程序结构信息，对程序进行分析。这些分析结果比其他一般的程序分析工具得到的信息更加详细、全面。这些信息将为我们在代码编译过程中，实现其安全加强和可信性验证提供有益的支持。

③ 在编译过程中，编译器会对程序代码进行各种转换，在这些转换过程中加入一些代码安全性加强措施比较方便。目前这一技术已成为许多软件安全防范的有效方法之一。

④ 编译优化技术已经研究多年，拥有许多非常成熟的理论和方法，这为可信编译理论和方法的研究提供了非常好的基础。相对其他的编译器组成部分而言，编译优化能更好地在安全和效能之间找到平衡点。

⑤ 有些开源编译器，如 GCC 等，既可以支持多语种又可以支持多目标机。如果我们能对其中间代码转换或其他共有阶段实施可信代码保证技术，则可使所得可信编译器同样支持多语种和多目标机。

⑥ 后端编译技术与硬件平台关系密切，目前许多编译技术可以利用系统的体系结构特征，如并行编译技术等。同样，在可信编译技术的研究中，我们也可以考虑利用系统硬件支持，通过硬件与编译器软件相结合的方法实现可信编译的目标。

⑦ 最重要的是，由于编译理论与方法研究是一个非常完整的体系，编译理论研究者们一般对编译系统都有较完整的了解和掌握，从高层的语言构造到比较底层的硬件结构特征都有全面的研究，这使得我们能够深入的、自顶向下来看待各种安全目标。这为在此基础上进行可信编译理论和实现方法的研究提供了坚实的研究基础。

基于以上分析可知，可信编译技术对于可信软件系统的实现来说不仅是一种非常重要的手段，而且是其基础、前提和关键。我们对可信编译相关理论及其实现方法进行了研究，实现了对编译器自身的可信性验证，并在编译过程中对所编译的程序代码的安全性进行加强、可信性进行验证，保证了系统平台上最终运行软件的安全和可靠，从而提高了整个计算系统的可信性。

1.2 相关研究现状

近年来，基于编译技术的软件安全性、可靠性问题在国外已逐渐引起相关研究者们的高度重视。安全编译器、验证编译器、证明编译器等相继被提出，编译器的安全性、可靠性问题正在逐渐成为继编译优化技术之后编译领域的另一个研究热点。对编译相关安全问题的研究目前主要从 3 个角度出发：第一类是对编译器自身的安全性问题进行研究，此类研究主要针对可信类恶意攻击进行防范，即防止攻

击者利用编译器对所编译程序进行破坏，但不涉及编译器所编译程序自身的安全性问题；第二类是对编译器所编译程序的安全性问题进行研究，力图通过编译过程保证最终生成可执行程序代码的安全性，此类研究主要涉及程序代码安全性证明以及代码防篡改、泄漏等技术；第三类是对安全程序设计语言设计及其相应编译系统的构造方法进行研究，力图通过程序设计语言自身的安全特性保证使用其所编写程序代码的安全性。

1.2.1 编译器自身的安全性问题研究

此类工作主要针对编译器编译操作的安全性进行研究，防止攻击者通过编译操作，在编译过程中对程序进行恶意篡改。针对该问题，研究者已经提出一些解决方法，例如关闭所有编译优化操作并对编译生成汇编代码进行重复检查等。由于这些方法大多数存在可靠性不高、代价较大、影响程序性能等缺陷，因而实用性较差。Leroy 等提出了一种借助 Coq 辅助定理证明工具对编译过程进行形式化验证的方法，确保源代码与编译生成目标代码之间的语义一致性，使得目标代码能够保持源代码具有的一些安全属性。虽然以前已有许多方法用于证明编译器操作的正确性，但大多数只是针对编译的某一个阶段操作，如代码优化或数据流分析等。Leroy 注重的是整个编译过程的正确性证明，尤其注重传统方法所忽视的源代码到汇编代码生成这一主要编译过程，且此方法的验证过程可由机器自动执行，效率较高、代价较小，因此具有实用意义。目前此方法的研究尚不完善，只能证明编译器对 C 语言一个很小子集编译过程的正确性，还无法满足对普通程序编译过程进行验证的需要。

1.2.2 编译所生成程序代码的安全性问题研究

此类研究又可大致分为代码安全漏洞防范、代码安全属性证明、可执行代码安全性保护 3 个方面。

1. 基于编译器的代码安全漏洞防范技术研究

此类研究主要针对代码中某种常见的安全漏洞，通过在编译过程中对代码进行修改，来防范安全漏洞可能引发的恶意攻击。此类研究较具有代表性的就是针对最常见的缓冲区溢出攻击的 StackGuard 和 StackShield 方法。StackGuard 方法通过编译过程对程序进行修改，使其每次在调用函数时，将返回地址压栈后，再将一个随机产生字压栈。当函数调用完毕后，查看此随机字是否被修改，若改动，说明有溢出攻击发生则报警并停止程序运行。StackShield 方法使用了另一种技术，它更改程序使其在运行时创建一个特别的堆栈用来存储函数返回地址的一份拷贝，并在受保护的函数开头和结尾分别增加一段代码，开头处代码用来将函数返

回地址拷贝到这个特殊堆栈中，而结尾处代码则用来将返回地址从特殊堆栈中拷贝回原程序运行堆栈。目前提出的此类方法比较多，但一般只是用于防范某种或几种特定的安全漏洞，针对问题过于单一，无法保证程序代码的完全安全。

2. 编译过程中代码可信属性证明方法研究

这方面研究主要针对编译过程中程序的可信性证明问题，目前主要集中在代码的安全属性证明方面。所谓可信的代码是指运行时行为与所期待行为一致的代码，但是完整精确地指明所期待的代码行为对于目前大多数复杂程序来说是不切实际的。因此，研究者提出使用安全策略对程序的行为进行规范，指明哪些行为是可以接受的，哪些是不允许的。如果代码符合这些安全策略则认为此代码是安全的。

这方面研究最具代表性的就是由卡耐基梅隆大学的 Necula 和 Lee 提出的携带证明代码(proof carrying code，PCC)。该方法主要用于验证代码是否遵循预先定义的某些安全条件。携带证明代码提出了一种程序验证框架，在代码发送方，根据用户提出的安全条件，对代码进行分析，产生一种安全形式化证明信息，并将此信息证书附加于原始代码上形成携带证明的代码。在代码接收方，根据相同安全条件，参照携带证明代码中的证书，对代码进行验证，如果通过验证则认为此代码是安全的，允许执行。携带证明代码中安全条件均采用一阶逻辑谓词进行描述，证明过程采用逻辑推理的方法，代码验证通过类型检查机制实现。携带证明代码主要可用于移动通信中代码的验证。携带证明代码对源代码没有做任何修改，只是将安全性证明信息附加在代码中，供代码使用者验证所用。但是携带证明代码存在证书代码量过大的问题，一般为待验证代码的 3～7 倍，严重影响了验证效率。

与携带证明代码相似，康奈尔大学 Kozen 的研究小组提出了有效代码验证(efficient code certification，ECC)。这是一种简单、有效的程序验证方法，可以看做是一种携带证明的代码验证方法，只是它的证书不是完整的形式化证明过程，而只是一些证明的引导信息。有效代码验证可以保证通过验证的代码具有控制流安全、内存安全、堆栈安全等特性。最重要的是有效代码验证可以采用对已有编译器进行修改的方式实现。虽然有效代码验证对代码安全性的描述能力没有携带证明代码强，但是其证明过程更加简单，验证效率更高。

目前这类方法的研究在代码安全性证明方面有一定进展，但仍存在可证明安全级别较低、验证所需附加信息量过大、验证效率不高等缺陷。

3. 编译生成可执行代码安全性保护技术研究

这方面研究技术主要用于防止攻击者对编译后可执行代码进行读取分析和篡改。所有攻击者对程序代码的攻击都是按两个关键步骤进行的，首先对程序代码

进行读取分析;然后根据程序分析结果,对代码进行恶意篡改。因此,要防止攻击者对程序代码进行攻击,就必须阻止攻击的这两个关键步骤。目前主要方法包括如下几种:

① 版权、水印(copyright notice、watermark),即在代码中嵌入版权信息,然后在运行时检测其正确性。由于插入的文本信息很容易被发现,所以一般由编译器变换代码的相关信息产生水印,如基本块和内存中函数的顺序模式都可以用来编码水印。水印在跟踪和识别代码、防范代码的非法复制方面非常有用。但它并不能阻止攻击,因为这需要附加代码对水印进行周期性检查。另外,有效的版权检查仍不能阻止未授权的代码篡改。

② 混淆代码(obfuscation),即故意生成顺序混乱的代码。该代码仍然是正确的且可以正常执行,但很难被攻击者理解。Collberg 等已给出了目前各种混淆变换算法的分类。该技术主要通过重新排列或变换代码,使得攻击者难以分析程序的控制流图。Wang 已证明分析混淆程序是 NP 难问题。混淆技术简单实用,不涉及任何加解密算法就可防止代码被攻击者分析。混淆技术具有很高的执行效率,但其可保证的安全级别较低。

③ 签名(signature),主要用来验证代码是否被更改,防止攻击者的恶意篡改行为。Horne 等提出了一种防篡改的自校验技术,该方法通过计算代码段的哈希值,然后与正确值比较,确定代码是否被篡改。通过比较哈希后的摘要值,可以检测出代码和数据的任何改变,并且不需花费太多的计算资源。该方法的可靠性依赖于校验和算法的保密程度,如果攻击者破解了校验和算法,那么验证系统将面临严重威胁。

④ 加解密(cryptographic architecture),通过编译器对程序代码进行加密,在安全协处理器中,程序可以加密的形式运行。这样代码就不会暴露于任何不可信的主存中,防止攻击者分析或窃取,如 IBM 的 Citadel、Dyad 等系统。这种基于密码体系结构的技术需要硬件平台的支持,编译器生成代码时就立即对其加密。但安全协处理器造价昂贵限制了这类技术的推广。比较折中的方案是给一般处理器增加现场可编程门阵列(field programmable gate array,FPGA)芯片,用以加速加密和解密的计算过程。

1.2.3 安全程序设计语言及其相应编译系统构造方法研究

目前大多数常用的程序设计语言所能提供的安全保证都是非常脆弱的。一般来说,从程序设计语言角度来解决代码安全性问题是一种有效的方法。许多研究者都开始从事此方面的研究,并取得一些研究成果,如类型安全汇编语言(type-safe assembly language, TAL)、语言修正(language modification)、引用监视(reference monitor)等。这方面研究的主要思路是,首先设计一种能够满足安全

要求的语言，所有的安全要求都由此语言自身的安全属性来保证，一般认为通过此语言表述的程序代码都是满足安全性要求的。然后，构造相应的编译转换系统，将高级语言程序转换成这种安全语言所表述的程序。

以哈佛大学的 Morrisett 等提出的类型安全汇编语言为例，它是一种新型强类型汇编语言，对传统非类型汇编语言进行了扩展，加入了类型注解、内存管理原语以及一套完善的类型规则，其中类型规则保证了类型安全汇编语言程序代码的内存安全、控制流安全和类型安全。类型安全汇编语言作为一种静态类型中间语言，可在高级语言程序的安全编译中起到重要作用。基于类型安全汇编语言的安全编译过程，先将高级语言程序中的类型信息通过一系列转换依附在目标代码中，形成带类型注解的类型安全汇编语言代码。程序验证过程中，类型检查器对类型安全汇编语言代码进行静态检查，通过判断程序语义与施加的类型限制是否一致来确定目标代码是否安全，因此在某种程度上说类型安全汇编语言也是一种携带证明代码。与携带证明代码相比，虽然类型安全汇编语言不如携带证明代码表述得清楚，但是已可以处理所有可用类型系统术语表示的安全策略，其中包括了内存安全、栈安全和控制流安全等基本安全策略。类型安全汇编语言在面对编译优化时非常健壮，因为它的类型注解是与代码一起变换的，现已证明类型安全汇编语言完全适合于大规模工业应用。

现在最大的问题就是该方法适用的语言都是非常简单的，还无法满足常用高级程序设计语言与其之间进行转换的要求，而且其自身能实现的安全要求也非常有限。

在国内，如何通过编译的方式来加强程序代码安全性问题一直是编译领域一个备受关注的内容，也已取得了一些初步成果，开发了一些安全编译器。但是，大多数安全编译器所做的工作都是针对某一特定种类的攻击方式的。由于程序的安全漏洞类型多种多样，针对这些漏洞的攻击方式也将是层出不穷的，所以这些安全编译器并不能从根本上解决程序的安全性问题。这些安全编译器缺乏形式化验证机制，所能保证代码的安全级别较低，无法满足当前可信平台的要求，还不能称为可信编译器。另外，对于程序的安全性证明方面的研究也逐步引起了国内研究人员的关注，中国科学技术大学陈意云教授带领的研究小组，在代码安全性检查、代码迷惑等方面做了有益的研究工作。

虽然这些理论和方法还存在一定的缺陷，但却为我们今后可信编译理论和实现方法的研究奠定了良好的基础。例如，我们可以借助现有的一些安全编译技术在编译过程中对相应的程序漏洞进行防范；在现有的程序证明理论基础上对其进行扩展，使其满足更加复杂的高级语言程序证明的需要；根据现有的可执行程序代码安全性保护技术，对其进行改进，并在编译后端实现，保证可信编译最终所得代码的可靠性等。