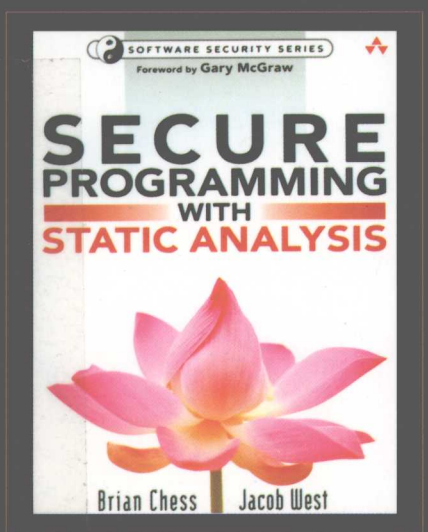
HZ BOOKS
PEARSON Addison Wesley
华章程序员书库

# 安全编程

## 代码静态分析

## Secure Programming with Static Analysis




（美）Brian Chess
Jacob West 著

董启雄 韩平 程永敬 等译

“本书向您展示如何应用高级静态分析技术来创建更为安全、更为可靠的软件。”

—— Bill Joy，Java语言的发明人之一

附赠光盘

机械工业出版社
China Machine Press

华章程序员书库

# 安全编程

## 代码静态分析

## Secure Programming with Static Analysis

（美）Brian Chess
Jacob West 著

董启雄 韩平 程永敬 等译

机械工业出版社
China Machine Press

本书介绍应用静态分析技术创建安全软件的方法，共分为4个部分。第一部分“软件安全和静态分析”，讲述软件安全静态分析概述性的内容，即软件安全问题；静态分析帮助改善软件安全性的方法；以及将静态分析集成到软件开发过程等。第二部分“常见问题”，探讨当前最为常见的安全问题和安全缺陷类型，并用来自安全实践的Java和C代码实例阐明了：如何发现编码错误；如何防止出现编码错误；以及通过静态分析如何能够快速找出类似的错误等。第三部分“特性与特色”，处理影响常见的各种程序以及特殊软件功能的相关安全问题。第四部分“静态分析实践”，给出一组展现静态分析如何能够改进软件安全的动手实践。

本书适合于软件开发人员、软件安全工程师、软件分析师、软件测试人员以及其他关注构建更加安全的软件的人员。



**本书版权登记号：图字：01-2007-4205**

**图书在版编目（CIP）数据**

安全编程：代码静态分析/（美）奇思（Chess, B.），（美）韦斯特（West, J.）著；董启雄等译. —北京：机械工业出版社，2008.3

（华章程序员书库）

书名原文：Secure Programming with Static Analysis

ISBN 978-7-111-23321-3

Ⅰ. 安…　Ⅱ. ①奇…　②韦…　③董…　Ⅲ. 软件开发—安全技术　Ⅳ. TP311.52

中国版本图书馆CIP数据核字（2008）第006785号

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：周茂辉

北京京北制版厂印刷 · 新华书店北京发行所发行

2008年3月第1版第1次印刷

186mm × 240mm · 24印张

标准书号：ISBN 978-7-111-23321-3

ISBN 978-7-89482-547-6（光盘）

定价：56.00元（附光盘）

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线（010）68326294

# 本书的赞誉

“为了能够对Java进行静态分析，我们对其进行了设计。本书向你展示了如何应用高级静态分析技术来创建更为安全、更为可靠的软件。”

——Bill Joy

Sun Microsystems（Sun微系统公司）的共同创始人、Java编程语言的共同发明人

“如果你想知道新的优秀代码扫描工具如何能够提高你的软件安全性，那么本书正好适合你阅读。作为此类图书的首部作品，本书写得非常好，它讲述了你需要知道的内容，但没有太过于陷入细节。在这方面，本书堪称典范。”

——David Wagner

加利福尼亚大学伯克利分校副教授

“Brian和Jacob能从大量实践的视角进行有关软件安全的著述。他们编写的著作充满了有益的忠告。”

——Marcus Ranum

防火墙发明者，Tenable Security公司首席科学家

“过去的几年中，我们已经看到有多本软件安全方面的图书在书店成功销售，这其中也包括我本人的书。这些书都针对优秀软件安全实践提出了他们自己的见解，而本书则填补了以往图书未曾涉及的空白。作者完成了一项伟大的工作：详细描述了如何使用当前所有可用的工具和技术来进行代码静态分析。值得称道的是，本书以对此主题清晰的认识将开发人员武装起来，同时提供了大量将这些认识付诸于实践的实用指导。当今每个从事软件开发的人都应将本书列入自己的必读图书清单。”

——Kenneth R. van Wyk

KRvW Associates，LLC.公司总裁兼首席咨询师

“软件开发人员是其代码安全的第一道也是最佳的一道防线。本书为他们提供了安全开发知识和相关的工具，以便在软件成为最终产品而被利用之前消除安全漏洞。”

——Howard A. Schmidt

前白宫计算机安全顾问

“现代的人造物品是通过计算机辅助来建造的。没有这些最为复杂、最为先进的工具，修建大桥、隧道或者飞机将是无法想象的事情。可是，出于某些原因，许多程序员却在没有优秀静态分析工具辅助的情况下开发他们的软件。这是很多软件系统充满Bug（程序缺陷）的主要原因，

而这些Bug原本是可以避免的。在这本杰出的图书中，Brian Chess和Jacob West为程序员提供了一份非常宝贵的资源。拥有了本书提供的实用指导，开发人员最终将可能全面利用技术进步来创作出更好的代码。阅读本书是任何严肃编程的首要之选。”

——Avi Rubin博士
约翰斯·霍普金斯大学 计算机科学教授
独立安全评估（Independent Security Evaluators）公司总裁、共同创始人

“如果世上有后悔药卖的话，应用程序安全绝对是当今必需的后悔药。坏蛋们将发现如何滥用你的软件，其破坏方式你也可想象到——花掉老板的钱还要败坏其声誉。Brain Chess和Jacob West提供了适时而卓越的指导，使你能从一开始就将安全性和弹性加入到应用程序的设计中。现在就买这本书，今晚就开始阅读吧。”

——Steve Riley
微软公司可信赖计算计划高级安全战略家

“通过详尽的代码示例，本书提供了具体的、技术方面的详细资料，这些都是你现在开始编写安全的软件所需要的东西。安全缺陷可能难以找出并修复，所以，Chess和West向我们展示了如何使用静态分析工具来可靠地找出缺陷，并且提供了代码示例演示修复这些缺陷的最佳途径。本书是一本适合于所有软件工程师阅读的优秀图书，同时，也是软件安全课程中，用于McGraw出版社面向方法的软件安全系列图书之面向代码的理想配套图书。”

——James Walden
北肯塔基州大学（Northern Kentucky University）计算机科学助理教授

“Brian和Jacob从独特的视角——静态源代码分析出发，描述了当前大多数严重安全问题的许多根本原因。

本书使用了大量真实的源代码实例，并结合易懂的理论分析和评估，它以如此简单而实用的方式为软件开发人员解释代码漏洞，这是我所读过的同类书中最好的一本。”

——Gang Cheng博士

“基于他们在软件行业和理论研究方面广博的经验，本书作者以一致的模式通过实例阐明了一些有独到见解的软件安全实践。本书倡导了实用的静态分析方法，这使得它从众多同类图书中脱颖而出。我相信，本书所倡导的这种方法必将对提高软件安全性产生重大的影响。”

——Hao Chen博士
UC Davis大学计算机科学助理教授

# 译 者 序

在我以前工作过的一个团队中，在解决软件质量和安全性问题的“摸着石头过河”的过程中，我们从理论上知道，除了继续在做的二进制代码的所谓“黑盒”测试外，还应该尝试开展“白盒测试”。于是，有一个小组引入了一位“白盒”测试工程师。在我的印象中，那位工程师整日在看代码，屏幕上总是很多代码在翻翻滚滚；他采用人工的方式遍历代码，以期找出问题。

过了没有多久，大概也就是我基本上熟悉这位工程师的时候，突然有一段时间看不到他了。问起他们的小组负责人，才知道已经辞职了。原因双方都有，我记得比较清楚的是老板不知道他做了些什么，或者说，工作成绩看不到，工作效果不明显。我们首次进行代码静态分析的尝试就这样失败了。

翻译本书的过程中，我实际上是带着对静态分析的怀疑和疑问的：代码分析可能有效吗？究竟能提供多大的帮助呢？代码分析人员和程序员怎么沟通呢？等等。毕竟，曾经的尝试给我留下了比较深的负面印象。

采用自动化方式进行代码静态分析，是本书介绍的静态分析技术和方法的核心，其应用目标重点放在了代码安全问题的查找上。读了这本书，我对代码静态分析有了新的认识。这确实是关于代码静态分析的一本很好的入门书！尽管本书的重点放在了针对安全问题的代码静态分析上，但是，书中讲的基本原理也适用于一般的代码静态分析。读了这本书，我想，我们的尝试之所以失败，除了其他的各种主客观因素外，至少在技术思路上有两个问题：一是采用人工的方式，效率很低，很难让静态分析的效果快速、量化显现出来；二是在结果的表达上，缺少像Fortify的Audit Workbench给出来的那样一份详尽而易懂的报告。就像书中所说，结果的表示和审计非常重要！

编写安全代码的重要性我们已经一再阐述，相信读者也已经认同。我们之前已经为大家推荐了程序员角度的《编写安全的代码》和安全测试人员角度的《软件安全测试艺术》，这次，我们为大家推荐这本书。这本书是面向开发人员和安全测试人员的一种较高级别的书。说它较高级别，是因为静态分析在实践上要求的起点比较高，熟练使用静态分析工具开展安全测试，尽管同样需要安全方面的知识，但还需要更多的关于软件开发方面的知识。比如，读者应该了解编译原理，知道基本的数理逻辑方面的概念，还应当有基本的编程语言方面的知识等。所以说，作为“白盒”测试的一种方式，静态分析对安全测试人员提出了较高的要求，甚至可能需要开发团队的人员来参与这项工作。

不过，好在有这本书，它从基础讲起，让我们快速地建立起静态分析技术的基本概念，并从实践的角度了解静态分析工具如何用于我们的安全测试中，特别是能够理解结果应该如何表达，如何使用。通过本书，相信大家能全面地了解静态分析技术和静态分析工具，并为进一步

实际开展静态分析工作打下基础。

本书的翻译工作由程永敬组织，主要由董启雄、韩平、程永敬负责翻译，参加翻译的还有高晓玲、李波、张宝玲、叶伟、卢敏、汪顶武、安志琦、费玮、张旭等。

由于时间有限，加之译者的精力和能力所限，尽管我们尽了自己最大努力，但书中仍会有不少不尽人意之处，敬请读者不吝指正！您可以发送邮件到：chengyongjing@126.com。

程永敬

2007年11月

# 序

## 使用静态分析工具进行软件安全审查和代码审查

建筑工程师在开课的第一天往往会学习关键的一课："关注并弄清楚桥梁修建细节，否则你建起来的桥就有可能坍塌"。这一课还通过播放Tacoma Narrows大桥把自己晃塌（http://www.enm.bris.ac.uk/anm/tacoma/tacoma.html）的一段视频非常有力地进行了例证。图1展示了1940年这座大桥600英尺的一段坍塌落入水中的情景。与此相反，在软件工程课程的第一天，那些初入门的开发人员得到的教导却是他们能构建所有能想到的东西。通常他们都是从"hello world"开始的。

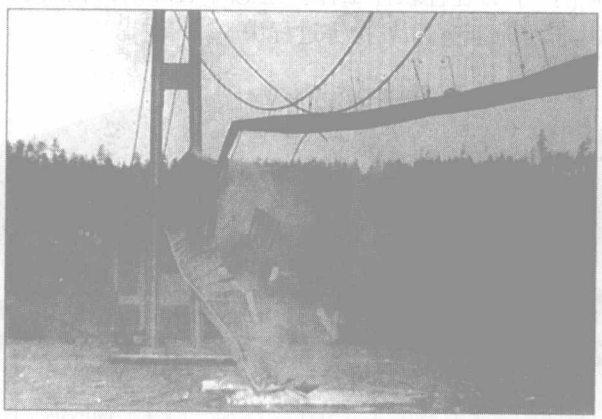


图1 Tacoma Narrows大桥600英尺的一段坍塌落入普吉特湾（Puget Sound），就像这座桥自己扭曲转动到坍塌一样。建筑工程师们早期就得到这方面的警告：如果他们不能苦练为一个优秀的工程师的话，就会发生这样的事故

过度乐观的软件开发方式无疑已导致创建了很多难以想像的东西，然而，从安全的视角来看，这同样也让我们自己陷入了困境。简单地说，我们没能想到：如果发生有目的的恶意攻击，我们的软件会出现什么情况？

许多当前的软件是非常脆弱的，只有当其环境是纯净而可预知的时候，其功能才能正常运转。如果将我们这脆弱的软件所运行的环境换作一种好斗的、恶意的环境（就像Internet这样的实际环境），软件的崩溃将非常地惊人，它将轰然落入类似的普吉特湾。

当今计算机安全中存在的最大问题就是大多数系统在构建的时候就没有安全意识。诸如防火墙之类的反应式网络技术能帮我们减轻"脚本小子们"对服务器的攻击，但这类技术并不能解决真正的安全问题：质量低劣的软件。如果我们想要解决这类计算机安全问题，就需要在构建安全的软件上下更多的功夫。

软件安全是使得软件能够在恶意攻击的情况下安全而正常运转的一种软件构建实践。本书主要讲解的就是关于软件安全中最重要的实践之一：使用静态分析工具进行代码审查。

随着业界人士对软件安全的重要性愈来愈重视，人们逐步采纳并开展了一系列的最佳实践来致力于解决这一问题。微软公司的可信赖计算计划（Trustworthy Computing Initiative）取得了显著的成就。许多Cigital公司的用户正处于企业范围的软件安全计划之中。当前，在实践中应用的绝大多数方法都是围绕着对开发人员、测试人员以及架构师进行培训，软件工件的分析和审计，开展安全工程。将软件安全尽可能深入地引进到开发过程，并利用软件从业人员过去几年中学过的这些安全工程课程，舍此无它。

在我的《Software Security》那本书中，我引入了一组称作“接触点（touchpoint）”的7大最佳实践。实现软件安全要求对大多数组织的软件构建方式进行某些更改。好在这些更改并不是基础性的、地震式的或者是开销巨大的更改。事实上，采用一系列简单明了的最佳工程实践，将安全交叉插入到现有开发过程并以此方式开展软件设计，所要做的更改通常就只是这些。

图2详细说明了这些软件安全接触点（touchpoint），并展示了软件从业人员如何将这些接触点应用到软件开发过程中产生的各种软件工件。这意味着理解如何将安全工程引入到需求、架构、设计、编码、测试、检验、度量和维护等过程中。

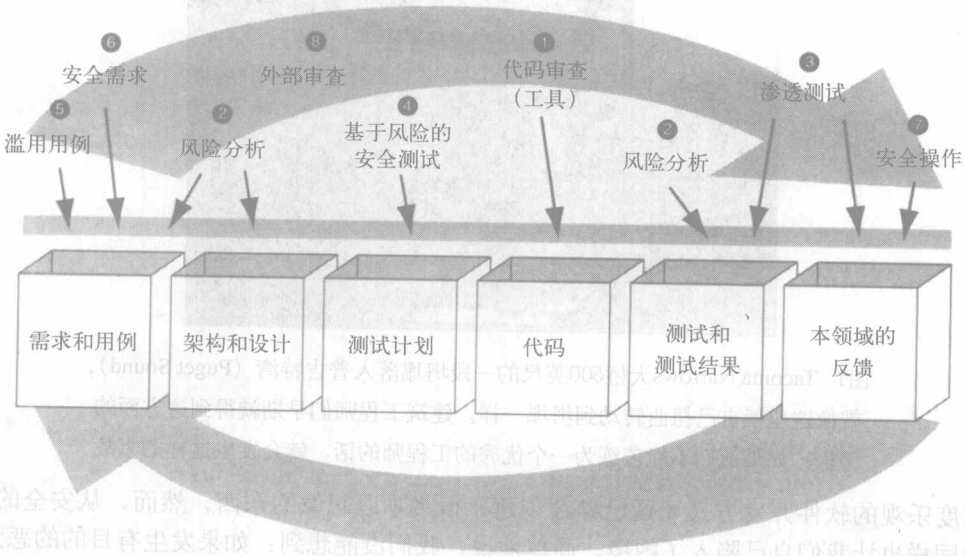



图2 在《Software Security: Building Security In》一书中引入和充实的软件安全接触点

某些接触点本身就比其他接触点影响力要大。首先采用这些最具影响力的接触点不失为一种稳健的做法。排在前面的两个接触点是使用静态分析工具进行代码评审和对架构的风险分析。本书全部内容将围绕前者展开。

所有的软件工程都至少会产生一个工件：代码。这一事实使得我们将代码审查放在我们列表中第一号的位置。在代码级别上，我们的关注点将集中在发现缺陷上，特别是那些用来扫描代码中常见漏洞的静态分析工具所能找出来的缺陷。现在，有好几家工具厂商都致力于这一领

域，其中包括Fortify Software公司——Brian和Jacob就供职于此。

发现缺陷（Bug）不仅数量众多，而且也很常见（就像维吉尼亚乡村的臭虫一样），其中包括了由于使用或误用存在漏洞的API（如C语言中的gets()、strcpy()等）而带来的一些很难处理的东西——例如臭名昭著的缓冲区溢出。无论是人工的还是（甚至更为重要的）使用静态分析工具的自动化方法，代码审查总是试图在软件发布之前识别出安全缺陷。

当然，哪种单一的技术都不是万能药。对于实现安全的软件来说，代码审查是一项必要而非充分的例行工作。安全缺陷（尤其是C语言和C++语言中的安全缺陷）是一种实际存在的问题，而架构的缺陷则是一个大问题。开展代码审查是一项极其有用的活动，但是，就算是这样，这种审查只能识别缺陷。最好的情况下，代码审查能揭示出50%左右的安全问题。而架构的问题很难（并且很大程度上不可能）通过紧盯代码的方式找出来。对于由无数代码行组成的现代系统来说，更是这样。而一种综合的软件安全方式包括了对代码审查和架构分析的全面整合运用。

就其本质来讲，代码审查要求编码方面的知识。在代码审查期间，一个几乎没有软件编写和编译经验的信息安全从业者没有太大用处。代码审查步骤最好还是交给开发组织的成员来完成，尤其是当他们配备了某种现代源代码分析工具的时候，更应该让他们进行这项工作。除了那些在编程语言和代码级漏洞解决方案方面具有丰富经验的信息安全人员以外，代码审查阶段的工作本质上并不适合网络安全专家来做。对于那些目前正在尝试通过信息安全部门在企业实行软件安全的组织来说，可能会对这种说法非常惊讶。即使这些组织关于安全性实施的这种想法是严谨的，但是，只有满足代码审查对在编码方面有具体的实践经验这一需求的时候，安全性才能在代码级别上成功实施。

如何查找缺陷？或者，找到缺陷后该怎么办？大多数开发人员对于这些问题都没有什么概念，这是个问题。这正是本书的切入点。你手上的这本书，是关于安全性静态分析和代码审查的所有已出版著作中水平最高的一部。本书不仅会告诉你缺陷是什么（我有时候称之为软件安全的“缺陷检阅”方法），还将教你如何使用现代静态分析工具来找出这些缺陷，此外，更为重要的是，本书还教你如何来改正这些缺陷。通过将书中的教导付诸实践，你将在解决软件安全问题的道路上迈出一大步。

Gary McGraw博士

于维吉尼亚州贝利维尔（Berryville，Virginia）

2007年3月6日

公司：www.cigital.com

播客（Podcast）：www.cigital.com/silverbullet

博客（Blog）：www.cigital.com/justiceleague

图书著作：www.swsec.com

# 前　言

跟随太阳的指引，我们告别了旧世界。

——克里斯托夫·哥伦布

我们生活在一个经济空前增长的时代，而计算机和通信技术则不断地推动着这种增长。我们使用软件实现工厂自动化、贸易现代化，并将信息传递到那些依赖信息的人手中。我们生活在信息时代，而软件是使信息为我们所用的主要手段。

没有足够的安全性，我们将无法充分发挥数字时代的潜力。但令人奇怪的是，许多打着计算机安全旗号进行的活动事实上根本就不是在解决安全问题，而是清理安全问题产生的混乱局面而已。病毒扫描程序、防火墙、补丁管理以及入侵检测系统，是我们弥补软件安全不足所采用的所有手段。而软件行业将更多的精力放在了对安全缺陷的修补上，而不是致力于一开始就创建安全的软件。这并不是说我们认为这些缺陷修补的机制没有价值。就如同每艘海轮上都应有救生艇，对于我们软件行业来说，能建立一种快速修补最新发现漏洞的机制，这既是有益的、也是健全的。但是，即使这样，软件的安全状态仍然堪忧。每天都会发现新的漏洞。在某种意义上，我们成了盼望着每次轮船出海的时候都需要使用救生艇了。

要想改变软件安全的这种状态，就要改变软件构建的方式。这并不是件容易的事情。毕竟，程序员会犯无数错误！虽说潜在的错误可能会是无限的，但实际上，程序编写整体上却倾向于重复相同的错误。缓冲区溢出漏洞存在了大约20年，这一事实就是此论点很好的例证。1998年，莫里斯（Morris）病毒使得Internet编程团体注意到缓冲区溢出会导致对安全性的破坏，但是，直到2004年，缓冲区溢出仍然被国际漏洞公布组织（Common Vulnerabilities and Exposures，CVE）列为引发安全问题的头号诱因[CWE，2006]。这种熟知错误的大量重复提示我们：当前遇到的许多安全问题是可预防的，而软件团体拥有避免这些错误出现所必需的经验技巧。

21世纪初，我们激动地投身到软件构建之中，这使人联想到拓荒年代投身造船事业的情景。哥伦布到达美洲的时候，拓荒是经济扩张背后的驱动力，而轮船则是拓荒者周游世界的工具。在哥伦布的年代，成为经济强国需要成为一个海军强国，这是因为，在轮船能够安全航行于新的贸易航线之后，才能使发现一块新大陆带来价值。同样，要使信息技术能带来好处，人们所使用的计算机系统就必须是安全的。一些批评家对即将到来的“网络战役”发出了警告，但是，既然我们知道软件安全作为一种主要因素左右着愿意投身网络技术的信任人群的数量，那么，我们就不会害怕像这种电子毁灭的预言。

我们认为，确保自己的作品是安全的，这是软件构建人员的责任。软件安全问题不能留给系统管理员或最终用户去处理。网络安全、明智的管理员以及谨慎的使用都很重要，但是，从长远来看，如果软件天生就有漏洞，这些努力最终是徒劳的。尽管有时候安全看起来会像是一种黑色艺术，或者是一种碰运气的事情，但是，我们希望证实不是这么回事。把安全说得让人

感觉是不可能的或者是神秘的，这就有点言过其实了。有了正确的知识和正确的工具，通过将安全构建到软件开发过程中，软件安全是可以实现的。

我们经常会遇到这样一些程序员，他们质疑软件安全是否是一种有价值的目标。毕竟，如果昨天没有人攻击你的软件，你为什么要相信明天就会有人攻击呢？安全要求付出某些额外的思考、更多的注意力和更大的努力。在过去的数10年间，这些额外的工作并不显得那么太重要，而那些没有经历过安全问题的程序员则用他们的好运气来判断未来，从而忽视了安全性。在对“挑战者”号航天飞机失事原因的调查中，Richard Feynman发现，NASA的风险评估是基于以前的航天飞机发射都是成功的这一事实得出的[Feynman，1986]。他们知道过去已经出现过不正常的运行状态，但他们用还没有发生过灾难来作为理由，从而相信永远都不会发生灾难。由此导致的安全系数的削弱使得失事的发生几乎是不可避免的。Feynman写到：“当玩俄罗斯轮盘赌的时候，第一枪安全逃过并不能在面对下一枪时用于自我安慰。”⊖

## 安全编程：使用静态分析

有两条交织的主线贯穿于全书：软件安全和静态源代码分析。我们将广泛讨论引起安全问题的常见代码错误，对每一种错误的安全后果进行解释，并对安全路线的制定给出建议。我们最常见的建议最终将归结到本书的主题：在代码错误被利用之前，使用静态分析工具识别出这些代码错误。我们主要关注商业软件，这些软件既可用于企业商用，也可用于普通消费者，但我们的重点将放在商用系统上。我们不会陷于那些构建有特殊安全需求的软件所必需的细节。尽管在构建一个操作系统或者一个电子投票机的特殊安全需求方面，需要注意的方面很多，但是，在我们遇到的程序员中，更多的人需要了解如何构建一个安全的网站或者如何构建一个安全的企业应用程序。

此外，我们首先希望提供一些实用的、并且立即就可以实行的建议，以消除软件安全缺陷。我们用了许多存在漏洞的实际代码示例来演示我们所讨论的安全缺陷，在本书随赠的光盘中还有一套静态源代码分析工具，这样，读者可以使用我们讲过的检测技术来进行实验。

本书不是一本安全功能特性、安全框架、或者安全API使用的指导书。我们不讨论Java安全管理器（Java Security Manager），不讨论密码技术，也不讨论身份管理的正确方法。无疑，这些都是重要的话题。这些内容太重要了，实际上，都有专门的书来探讨这些话题。而我们的目标是关注那些与安全功能特性无关的东西，一旦它们出错，将会致软件于危险的境地。

多数情况下，恶魔隐藏于细节之中。安全法则（以及违反安全法则）必须映射到其在源代码中的表现。我们将重点放在用C语言和Java语言编写的程序上，因为这两者是当前我们最常遇到的编程语言。我们还将大量关注其他语言。现在很多安全敏感的项目都是用C#、Visual Basic、PHP、Perl、Python、Ruby以及COBOL完成的，但是，即使是蜻蜓点水，在一本书里写到所有的语言也是很困难的。

某些情况下，我们所讨论的许多问题都是与编程语言无关的，我们希望读者在阅读这些示例的时候，能够越过示例的语法，理解其在你所使用的编程语言上的衍生结果。

⊖ 俄罗斯轮盘赌：逞能冒险举动，赌时只在左轮手枪中装一发子弹，然后转动旋转弹膛，将枪口对准自己的头并扳动扳机。——译者注

## 读者对象

本书为那些已经下决心优先考虑软件安全的人而写。我们希望程序员、项目经理以及软件架构师都能从阅读本书之中获益。尽管我们的宗旨并非关注有关软件安全和静态分析的详细知识，但本书涵盖了足够深入的这类主题内容，我们希望专业的代码审查人员和渗透测试人员也能够从中获益。我们假设读者能够轻松使用C语言或者Java语言编写程序，而且阅读用这两种语言写的简短示例都不会太吃力。某些章节倾向于更多地使用其中的某一种语言。例如，在关于缓冲区溢出的章节中，示例是用C语言编写的。

## 本书的组织结构

本书分为4个部分。第一部分“软件安全和静态分析”，讲述了一些概述性的内容：软件安全问题、静态分析帮助改善软件安全性的方法、以及将静态分析集成为软件开发过程的一部分的观点。第二部分“常见问题”，我们将关注一些普遍存在的安全问题，无论软件的功能是什么，都会受这些问题的影响。而在第三部分“特性与特色”中，我们将着手处理影响常见的各种程序以及特殊的软件功能的安全相关问题。第四部分“静态分析实践”，我们给出一组展现静态分析如何能够改进软件安全的实际动手练习，这些练习将第一部分、第二部分和第三部分的知识融会起来。

第1章，“软件安全问题”。该章从程序员的视角勾画出软件安全的困境：为什么软件总是容易出问题？为什么用来捕获程序Bug的典型方法用在查找安全问题上就不那么灵了？

第2章，“静态分析简介”。该章讨论静态分析所能解决的各种问题，包括结构、质量，当然，还有安全问题。我们将对一些开源的和商业的静态分析工具进行一个快速的概览。

第3章，“作为代码审查过程组成部分的静态分析”。该章将探讨静态分析工具如何能作为安全审查过程的组成部分付诸实施。我们分析，软件开发组织的决心是这些工具能得到有效使用的基础。此外，我们还将讨论基于静态分析输出的度量标准。

第4章，“静态分析技术内幕”。该章将深入探讨静态分析工具是如何工作的。我们将探究构建一套静态分析工具相关的基础组件，并考虑静态分析工具在实现良好的精确度和仍然能够扩展到对数百万行代码进行分析这两者之间的平衡。

第二部分对软件中普遍存在的安全问题进行了概述。整个这部分的章节及后续的章节中，我们将对安全的编程给出积极的指导，并随后使用专门的代码示例（其中许多代码来自实际的程序）来演示要避免出现的缺陷。顺着这种思路，我们将指出静态分析能够对软件安全提供帮助的地方。

第5章，“输入的处理”。该章讨论程序员曾面对的最头痛的软件安全话题，这同时也是未来他们最可能要面对的问题：对大量输入窗体以及非信任输入衍生的数据进行处理。

第6章，“缓冲区溢出”和第7章，“缓冲区溢出伴随的问题”。这两章将讨论一个输入带来的软件安全问题，这个问题已经伴随了我们数十年，即缓冲区溢出。第6章以一种战术性的方法开始：如何找出最有可能引起可被利用的缓冲区溢出的特殊代码结构部分。第7章将探究缓冲区溢出的间接原因，例如整数回绕（wrap-around）。然后，我们将进一步展开并对缓冲区溢出及解决

这一问题的可能方法进行一种更为战略性的研究。

第8章，“错误和异常”。该章主要探讨特殊情况下程序员考虑问题的方法。尽管错误和异常是相当少见的引发安全漏洞的直接原因，但却以一种非直接的方法与漏洞相关。意外条件和安全问题之间有着紧密的联系，因此，错误处理和恢复将是一个永远的话题。最后，该章还将讨论常见的日志记录和程序调试方法，这些通常都是作为一个整体与错误处理联系在一起的。

第三部分采用一样的风格进行积极的指导并给出专门的代码示例，以解决在常见类型的程序中发现的、以及和某些特殊软件功能特性相关的涉及安全的问题。

第9章，“Web应用程序”。该章将探讨当前最流行的安全话题：万维网。我们将探讨Web和HTTP协议所特有的安全问题。

第10章，“XML与Web服务”。该章将研究一个日渐显现的安全挑战：使用XML和Web服务在分布式组件之外构建应用程序。

尽管安全功能并不是我们所关注的焦点，但某些安全功能有明显的错误倾向，因而应当对其特殊对待。第11章，“隐私与秘密”，该章将对一些程序进行探讨，这些程序需要对私有信息提供保护，或者，更为常见的情况下，需要保存秘密。第12章，“具有特权的程序”，该章将探讨一些特殊的安全需求，当编写的程序需要操作与调用该程序的用户不同的权限集时，必须考虑这些安全需求。

第四部分是有关获得静态分析体验的内容。本书的随赠光盘中包含了一个静态分析工具，这是我们公司——Fortify Software公司免费赠送的版本，光盘中还有许多示例工程的源代码。

第13章，“Java语言源代码分析练习”，是从Java语言角度出发的一个涵盖静态分析的实验指导。第14章，“C语言源代码分析练习”的内容和第13章一样，只不过其中的例子和练习是用C语言编写的。

## 本书使用的一些约定

讨论安全错误会使人陷入一种消极的心态，或者抱有一种悲观的态度。我们力图通过将注意力放在纠正安全问题所需要做的事情上，从而坚持积极的立场。详细说明很重要，所以，当我们讨论编程错误的时候，我们试图给出一个有指导意义的实例来演示某种编程错误在详细审查之下的情形。当针对一个特殊问题的解决方案与我们的原始实例相比较发生较大变化的时候[⊖]，我们也会给出一份纠正这个问题后重写的代码版本。为直截了当地标示这些例子，我们用  这个图标表示在给出的代码中故意包含了一个缺点，用  这个图标表示给出的代码中已经纠正了那个缺点。

## 致谢

我们的编辑——Addison-Wesley出版社的Jessica Goldstein为我们做的事情远远不止帮我们熟悉出版流程；在RSA 2005上和她的一番谈话使得这个项目得以启动。Addison-Wesley出版社

⊖ 即修改前和修改后的代码相去甚远时。——译者注

的其他员工也为我们提供了巨大的帮助（并且非常耐心），他们中包括：Kristin Weinberger、Chris Zahn、Romny French以及Karen Gettman。

第1、2、3章的部分内容源于我们过去几年中写的一些技术论文和杂志上发表的文章。我们要感谢这些论文和文章的共同执笔人：Gary McGraw、Yekaterina Tsipenyuk O'Neil、Pravir Chandra以及John Steven。

对于一些实在是粗糙又粗糙的草稿，我们的审稿人真是受苦了，而他们总是返回一些有建设性的反馈意见。要多多感谢Gary McGraw、David Wagner、Geoff Morrison、Gary Hardy、Sean Fay、Richard Bejtlich、James Walden、Gang Cheng、Fredrick Lee、Steve Riley和Hao Chen。我们也受到了来自Fortify公司技术咨询部的很多必要的鼓励，他们有：Gary McGraw、Marcus Ranum、Avi Rubin、Fred Schneider、Matt Bishop、Li Gong、David Wagner、Greg Morrisett、Bill Pugh以及Bill Joy。

Fortify公司的每一个人都大力支持我们的工作，他们的大量工作都体现在了本书的随赠光盘上。我们非常感激他们给予我们的这些支持。我们还应当大力感谢Greg Nelson，他修正了我们对于静态分析的一些观点。

我们要特别感谢我们的家人：Brian家的Sally和Simon以及Jacob家的Jonathan。和在硅谷软件公司工作的人住在一起需要有巨大的耐心，而同时要容忍他写软件和写书那简直是圣人都比不过了。最后，还要感谢我们的父母。你们给我们指引了这条路，我们不打算转投他处。

# 作 者 简 介

Brian Chess是Fortify Software公司的创始人。目前，他是Fortify公司的首席科学家，他的工作重点是创建安全系统的实用方法。Brian在圣克鲁兹的加州大学（University of California）获得计算机工程博士学位，在那里他研究在代码中查找安全相关缺陷问题的静态分析。在走上安全这条路之前，Brian花了十年时间在硅谷大大小小的公司中工作。他的研究主题涉猎很广，范围从集成电路设计一直到将软件作为服务交付。他住在加州的山景城（Mountain View, California）。

Jacob West管理着Fortify Software公司的安全研究小组，该小组负责将安全知识构建到Fortify公司的产品中。Jacob带来了各种编程语言、框架和风格的专家意见，同时还带来了有关现实中的系统为何会失败的知识。在加入Fortify公司之前，Jacob在加州大学伯克利分校和David Wagner教授一起开发安全属性模型检查计划（MOdel Checking Programs for Security properties, MOPS），MOPS是一种用于揭示C语言程序中安全漏洞的静态分析工具。在告别编程之后，Jacob花了大量的时间在各种研讨会上发言，并和客户一起工作以提高他们对软件安全的理解。他住在加州的旧金山（San Francisco，California）。

# 目　录