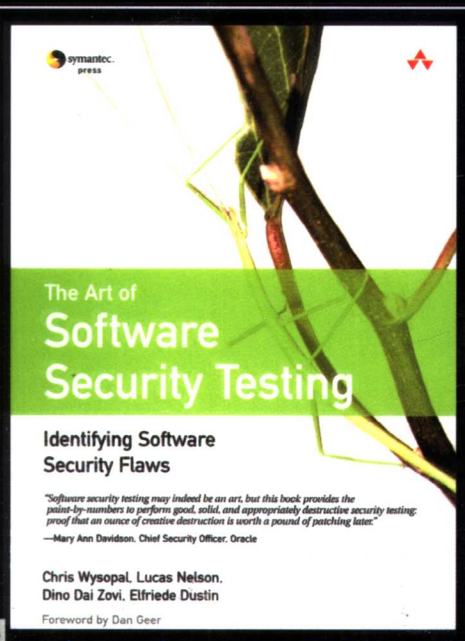
HZ BOOKS
PEARSON Addison Wesley
华章程序员书库

软件安全测试艺术

The Art of Software Security Testing

Identifying Software Security Flaws



（美）Chris Wysopal
Lucas Nelson
Dino Dai Zovi
Elfriede Dustin
著　程永敬 等译

软件安全测试真的是一门艺术，本书用大量数字形象地描述优秀的、原汁原味的、恰到好处的破坏性安全测试：证明了“一盎司”的创造性破坏活动需要后期付出“一磅”的修补工作来弥补。

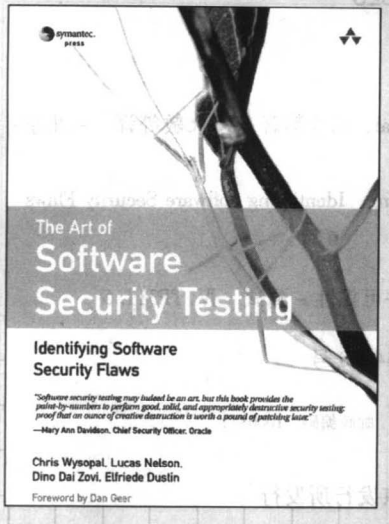
——Mary Ann Davidson
Oracle公司首席安全官

机械工业出版社
China Machine Press

软件安全测试艺术

The Art of Software Security Testing

Identifying Software Security Flaws



（美） Chris Wysopal
Lucas Nelson
Dino Dai Zovi 著　程永敬 等译
Elfriede Dustin

机械工业出版社
China Machine Press

本书囊括了应用程序和网络安全分析和测试方面的内容。分为三部分，第一部分讨论一些现实情况，主要介绍漏洞的产生、安全的软件开发生命周期、基于风险的安全测试和分析：白盒、黑盒和灰盒测试；第二部分主要分析网络上发现应用程序并对其进行攻击的方法；第三部分介绍如何判定漏洞的可利用性。

本书内容涉及广泛，叙述详尽，适合作为安全工程师、软件测试工程师及软件开发人员等的参考用书。

本书版权登记号：图字：01-2007-1829

图书在版编目（CIP）数据

软件安全测试艺术/（美）威斯波尔（Wysopal，C.）等著；程永敬等译．－北京：机械工业出版社，2007.8

书名原文：The Art of Software Security Testing，Identifying Software Security Flaws

ISBN 978-7-111-21973-6

Ⅰ.软…　Ⅱ.①威…　②程…　Ⅲ.软件可靠性－测试　Ⅳ.TP311.5

中国版本图书馆 CIP 数据核字（2007）第 114758 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：吴　怡　　王春华

北京牛山世兴印刷厂印刷·新华书店北京发行所发行

2007 年 8 月第 1 版第 1 次印刷

186mm×240mm·15 印张

定价：32.00 元

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“美誉”

作为本书的重点，基于风险的安全测试，是在我的那本 *Software Security: Building Security In* 书中引入的七个软件安全评测点之一。本书循序渐进地阐述了这一基本思想。本书由软件漏洞利用(exploit)大师所著，以非常精练的语言阐述了软件安全测试是如何有别于常规的软件测试的，这些均已为各个 QA(质量评价)小组所实践。它将 Michael Howard、David Litchfield、Greg Hoglund 和我的相关思想统一为一个简洁的介绍性篇章。马上阅读本书，改进你的安全测试吧！

——Gary McGraw 博士，Cigital 首席技术官；
Software Security，Exploiting Software，
Building Secure Software，和 Software Fault Injection 的作者；
个人主页：http://www.cigital.com/~gem

截至 2006 年底，我们将看到有超过 5000 个软件安全漏洞向公众公布。许多已发现或将发现的这类安全漏洞均来自企业应用程序，而这些企业应用程序往往出自那些配备了大量专业 QA 团队的公司。那么，这些安全缺陷又怎么能够一再地逃脱如此组织完善、勤勉有加的测试团队呢？在某种程度上说，问题的答案在于这种测试在验证安全缺陷的存在性时，基本上一直只浮于表面。幸而，本书有望将对安全测试领域的理解带到一个更为全面的层次，并且，随着时间的过去，能让我们的软件都更加安全一些。

——Alfred Huger，Symantec 公司高级研发主管

软件安全测试或许真的是一门艺术，本书用大量数字形象地描述优秀的、原汁原味的、恰到好处的破坏性安全测试，证明了“一盎司”的创造性破坏活动需要后期付出“一磅”的修补工作来弥补。如果能明白花一倍的工作就可以攻破软件，而每个程序员要相应付出 12 倍的工作来编写可防御的、安全的、健壮的软件，那么就知道其中用于软件安全测试的部分至少要占 2～6 倍的工作。

——Mary Ann Davidson，Oracle 公司首席安全官

在过去的几年中，出版了一些优秀的图书，这些图书通过描述常见安全故障模式，来指导程

序员如何编写更为安全的软件。然而，这些书都不是面向测试人员的，而测试人员的职责就是在软件从研发部门出来进入到用户手中之前找出其中的安全性问题。本书填补了这一空白。本书作者均在安全测试领域具有广泛的经验，他们阐明了如何使用自由软件工具来找出软件中的问题，还给出了大量的例子说明软件缺陷在这些测试工具中看起来是什么样子。读者将认识到安全缺陷为什么有别于其他类型的程序错误(bug)(我们不仅想知道“程序完成了哪些其设定的工作”，还要知道“程序不会做哪些其未设定的工作”)，并将了解如何使用相关工具来找出这些安全缺陷。书中示例主要基于C语言代码，但一些Java、C#以及脚本语言的描述也为那些使用这些开发环境的读者提供了帮助。作者在书中涵盖了Windows平台和基于UNIX平台的测试工具，同时提供了大量的屏幕截图来展示将会呈现的内容。无论是查找安全问题的软件复查人员，还是主要从事功能性测试的初级QA人员，每个从事软件QA测试工作的人员都应该阅读本书。

——Jeremy Epstein，WebMethods公司

译 者 序

尽管“软件质量”这个概念已经为大多数业内人士所熟知，国内的软件企业和软件开发团体也已经意识到软件质量保证在成本/效益等各方面的重要性，软件测试也逐步开展了起来，但是，很多企业和团体，特别是中小企业和自发组织的开发团体，在对软件质量保证及其实施的理解上，在业务流程、专职人员、专业知识等保障上还有很长的路要走。从这种意义上讲，我们的软件质量之路还刚刚起步。

软件安全，作为网络时代软件必然的需求，其范畴既与软件质量相关，又有别于软件质量，它是一种对软件质量的更高要求。在实际工作中，我们接触过很多中小企业和软件开发团体的成员，可以说，随着安全形势越来越严峻，大部分人的头脑中还是有安全意识的，但是，绝大多数的人却不知道如何实现安全的软件。《编写安全的代码》是一本指导软件开发团队遵循安全的编码原则来编写安全代码的权威指导教材。从这里开始，我们有了构建安全软件的系统知识。但是，作为产品的软件，其安全保证和质量保证一样，不能仅仅依靠开发团队来完成。安全测试是在攻击者之前发现软件安全缺陷，并及时修补软件安全漏洞的重要环节。近年来，国外已经出版了几部涉及这一领域的专著。本书是这些优秀著作之一，为我们做好软件安全测试、把好软件质量关提供了从方法学到具体实践的全面指导。

本书倡导的以攻击者的思维进行思考，以及通过威胁建模排定测试优先级等思维方式和方法论，为我们定义安全测试工作流程、设置安全测试工作岗位、制定安全测试工作计划，从而规范地开展安全测试，提供了非常有用的指导。书中介绍的各种安全攻击和安全测试技术，尤其是很多用于安全测试的开源或免费工具的用途和用法，更是为在这方面预算紧张的软件企业，特别是中小软件企业开展软件安全测试指明了可行之路。

很荣幸能将这本书的中文版带给大家！及早阅读本书，及早开展软件安全测试，可助您在降低软件风险、降低软件总体成本、提高软件安全性，乃至提高软件全面质量方面迈出一大步！

参加本书翻译工作的有程永敬、卢敏、安志琦、董启雄、韩平、费玮、汪顶武和李波等。在本书的翻译过程中，得到了机械工业出版社华章分社的编辑们大力支持，在此表示衷心的感谢！

由于时间以及译者经验和水平所限，尽管我们尽了最大努力，但书中难免还有不尽如人意的地方，敬请读者不吝指正！您可以发送邮件到：chengyongjing@126.com。对您的批评和指正，我们表示真诚的感谢！

程永敬

2007 年 6 月

序　言

在使自己信赖软件测试之前，谁能够议论测试的事情呢？谁都不能议论，也不会议论。因此，如果没有开展测试，原因会是别的什么，但都不是令人信服的拒绝测试的理由。这些原因不外有三种：负担不起测试，不测试也可以做得很好，不知道怎么测试。

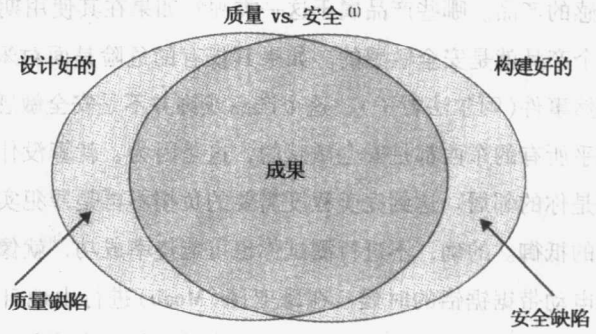
- **负担不起测试**——考虑到节俭的人不同意超过花费的限度，所有的开销都要事先做出选择。如果进行测试的话，有什么事情就不做了呢？如果不做的事情是给软件增加别的功能特性的话，或许你值得庆贺你选定了一个较简单的产品。较简单的产品事实上比较容易测试(并且还有一个好的理由：安全的主要敌人是复杂性，而没有什么比逐步增加功能特性更能带来复杂性的了)。如果不进行测试的话，通常给出的理由是“要按时把产品弄出来”。如果不是使性子的话，这个理由是不够充分的。本书所讲的这种测试致力于软件的未来，其重要性甚至要超过按时把产品弄出来。只有那些醉心于财富之梦的 CEO 们才阻止测试而不去考虑软件的未来。测试是努力让你控制未来，而不是让它控制你。测试让不可避免的未来产品故障提前在当前出现。当威廉·吉布森说出名言“未来已经来临，只是尚未广为流传而已”的时候，他并不关心我们这里讲的测试。你显然想做的事情就是把未来不对等地发布，以便能在你的用户(和竞争对手)之前看到产品的未来。读了这一段以后，你极可能在头脑中已有测试的框架，所以，我们停止这个论题，继续看下一个问题。
- **不测试也可以做得很好**——某些产品或许不需要太多的测试。这些产品和创新无关，它们是不易损坏的日用品，或者是相当无聊的东西。这不是我们关心的东西，我们这里所说的是要保护安全敏感的产品。哪些产品属于这一类呢？如果在其使用期间将面对感觉灵敏的对手，那么，这个产品就是安全敏感的。如果其仅有的危险是面对笨蛋(“嗨，看这个!”)或者无规则的偶然事件(阿尔法粒子)，这个产品或许并不是安全敏感的。但是，作为软件和网络来说，几乎所有的东西都是安全敏感的，这是因为，就算没什么危险，起码每个极富攻击性的人都是你的邻居。达到完美程度需要的负担不再是罪犯实施完美的犯罪，而是防卫者实行完美的抵御。的确，不进行测试你也可能侥幸成功，就像你不穿戴任何保护装置，也可能在用电动带锯锯铝的时候、在摩亚国(Moab)进行山地自行车赛的时候或者在清洗 P3 级防放射泄露装置实验室时侥幸地不会受到伤害。总是有人能侥幸地逃过这些伤

害，甚至是比这更危险的情况。但这种理由在这里并不适用。为什么？因为你的产品越成功，传播越广，那些攻击的人就越多，那些感觉灵敏的对手也会越多，他们将选中你的产品作为特别项目。只需要问问微软就知道了。如果你想要让你的产品得到广泛的使用，就需要对软件进行测试。唯一的问题就是“由谁来测试?”

- **不知道怎么测试**——这又让我们回到了本书的宗旨。你已经准备好了，也愿意开展测试，可是你不会。或者，你想要确保你不落后于你的对手。抑或，你需要比这里罗列的内容更多的“极限运动”的原始材料。你到这个地方就来对了(这里不是“独一无二”的地方)。这是(让我们清楚地说)一个完全正确的地方。作者久经考验，而技术是当前最新的。尽管安全相关的技术具有从未“完备”的极大优点，但你所要做的不会超过这个范围。如果你愿意，就可以成为本书的读者。同时，掌握 Chris Wysopal、Lucas Nelson、Dino Dai Zovi 和 Elfriede Dustin 教给你的东西，并将其付诸实践。像这样的技能集合不会无限增长，而你的对手的技能发展也不会比这些东西发展得更多。

从目录中你可以看到，测试是一种思考方法，而不是一个待按的按钮，也不是一个待批准的预算。你极可能只有接受这种思考方法，并且，除了工具的常规使用和具体化的技术之外，没有什么东西更能强化思考的方法。这不是开玩笑。外部的攻击者技能非常熟练且日益专业，同时拥有工具和思考模式。恶意软件，特别是那些把好公民变成无心的坏公民的恶意软件，已经使得长期存在的对安全威胁者的假定变为现实：真正的威胁来自内部。

问：外部攻击者度量成功的第一个依据是什么？答：获得内部的信任、访问和授权。如果这个攻击者计划通过利用目标内部所运行软件的漏洞来达到这个目的，那么只有你的设计和测试能够阻止攻击者的行动。如下图所示，其中思想不是“产品在做其设定的工作吗?”而是“产品没有做它未设定的工作吧?”这个问题比质量保证问题更为严峻，因为这是安全性内在的一个口子。这种问题并不能由开发本身全部处理，它必须是经过测试保证的，更适合由受过专门训练的测试人员来执行，而不是和软件构建过程搅到一起。



这同样是本书的着眼点。本书将讲述如何施加专家级的压力，进行加快故障出现时间的测试。你将学会如何高效地完成这样的测试，以使你乐于开展测试并摒弃不经过测试而侥幸成功的想法。换句话说，这就像在灌木丛中打猎。你自学也能学会怎么做，但跟随一个专家级的追踪者比艰难地学习每一件事情更快。掌握本书所提供的东西，你不仅可以成为一个强大的猎手，而且还有自己做一个追踪者的机会。别忘了，所有的技能都是实践的结果。这些作者都是经验丰富的；现在轮到你了，他们已经帮了你一把。

Daniel E. Geer, Jr.，理学博士

2006 年 7 月 24 日

附注

〔1〕Herbert H. Thompson 和 James A. Whittaker 著.《Testing for Software Security》(软件安全测试). *Dr. Dobb's Journal*, 27(11): 24-32, November 2002。

前　言

随着针对计算机系统的威胁的逐步增长，许多组织和机构都在寻求一些解决方案，用以保护品牌资产价值和消费者的信心，并将维护费用减少到最低限度。

由于威胁来源的范围不断扩大，这种挑战也在不断增长。攻击者的目标始于基于 UNIX 系统的 Internet 服务，而后转向了基于 Windows 系统的 PC。现在，他们开始瞄准了 Apple 公司的 MacOS X 系统、网络视频游戏控制台、无线手持设备甚至手机。只要一有软件漏洞曝光，通常用不了一个星期，黑客很快就会利用这个漏洞。而即使是脆弱期得以缩短（“window of vulnerability”这里译作“脆弱期”。这是来自军事领域的一个术语，是指一个期限，在这段时间内，防御措施或者缺失，或者被削弱或损坏。——译者注）从漏洞被曝光到软件供应商发布相应的安全补丁之前，这段时间仍然需要大约 6 个星期的时间[1]。

来自 Symantec 公司的报告显示，针对诸如防火墙和路由器这类传统安全设施展开的大型、多目标攻击正逐步淡出攻击者的关注重点。根据 Symantec 公司统计，2005 年下半年报告的漏洞中有 69% 出自 Web 应用程序。

不同于以往，目前，攻击者的注意力集中于区域性目标、个人桌面计算机以及 Web 应用程序，这些目标都潜在存在着被攻击者窃取公司信息、个人信息、财务信息或机密信息的可能性。Symantec 公司一份新的 Internet 安全威胁报告中列举了几种增长中的攻击者行为趋势，即：使用 bot[2] 网络；针对 Web 应用程序和 Web 浏览器的攻击，以及模块化恶意代码。

安全问题往往被转换为一个组织的收益或名誉损失。这些问题也会带来市场份额的丢失，对于具体组织的远景而言，这或许是至关重要的。据安全公司 Counterpane Internet Security 和 MessageLabs 估算，一段中度感染率的恶意软件会给一个小公司一年带来 83 000 美元的损失，而对于一个大公司来说，这笔损失可能达到 100 万美元或更多。他们补充说，这还只是直接损失，尚未包含诸如名誉损失之类的间接损失。[3]

Dave Cullinane 是位于西雅图的财务公司 Washington Mutual 公司的首席信息安全官，他说：“如果你向 Internet 开放一个应用程序，供人们来理财，这个应用程序就将被探测（probe）”。Cullinane 认为，这样的应用程序被攻破的后果不仅仅是财务损失，还将损坏公司的声誉。“声誉

危机完全会导致破产。如果你报告出一起安全失效事件，就会有 20% ~45% 的客户从你这里离开。"[4] 处理信用卡交易的 CardSystems 系统，最近由于该软件让罪犯进入偷走了数百万客户的个人财务数据而停止了使用。那些购买了 CardSystems 系统的公司已被 FTC(联邦贸易委员会)指令在未来 20 年中接受独立审计。

本书将探讨如何预防这种对敏感数据的不安全处理。具体而言，针对这种想定情况的测试将在第 6 章和第 7 章中讨论。

此外，如果这个公司把安全测试作为其软件研发过程的一部分，并学会如何检测这种情况的话，或许可以防止这场 CardSystems 灾难的发生。那时，检测出问题之后，该公司就可以采取本书所述的隐患消除步骤。切记，存储太多秘密完全是违反 VISA 信用卡的 PCI(支付卡行业)规则的行为。这是应该贯彻的信用卡安全标准和信用卡安全策略要求，详细内容将在第 3 章中讨论。

据 Gartner 公司的研究表明，这种威胁是巨大的，70% 的业务安全漏洞存在于应用层。而据 Microsoft 公司所做的研究，其中 64% 的自用商务软件开发人员承认他们对于编写安全的应用程序缺乏信心，这使得问题更加复杂。

及早开展安全测试，将有助于在很大范围内阻止来自现在和未来的威胁。保护客户数据(举例来说，包括在线交易中包含的个人信息)的安全性和完整性，对于保持客户的信赖是至关重要的。

本书致力于解决当前软件安全工程师、软件项目经理以及其他负责应用程序安全的软件从业人员所面临的这种亟待解决的问题。

这些专业人员从事系统的开发和部署工作。他们顶着完成安全的开发工作和升级合并工作的压力，以及在敌人破坏之前维护系统安全的压力。

本书的目标是指导从事软件构建和测试的工程师来为他们的客户提供保护，他们必须将安全测试贯穿整个开发过程。如果安全不能贯彻到整个开发的生命周期中的话，软件将不可避免地出现安全问题，这些问题将可能导致财务信息被窃、数据丢失、性能下降，并危及私人数据的安全。过去，软件供应商可能没有在安全测试上花太多精力而侥幸成功。他们只是简单地等待外部的研究人员或客户发现安全问题，而后报告给他们，然后他们修复这个问题并发布一个补丁程序。面对数百个不时要在数千台机器上测试和安装的补丁，企业客户都快被压垮了。

通过将安全测试环节补充到他们的开发过程中，软件开发人员能够将不良公众信息和非信任用户阻止在外。每家软件发行商都有一个质量保障过程，在这个过程中，他们的 QA(质量保障)从业人员创建并执行测试，以验证软件的功能性。安全测试可以扩展这个过程，进而验证软件是否存在常见的各类安全漏洞。过去，一些较差的软件提供商因为总把他们的客户当作测试人员来

用而受到责难。现在，那些没有开展安全测试的公司将他们的客户置于危险的境地，他们总在等外人自愿告诉他们其软件的安全问题。遗憾的是，不是每个人都是正直的人。有些人发现了软件中的安全问题却不会告诉该软件的供应商，反而可能会利用这些漏洞来窃取信息。

2005 年 12 月，罪犯在俄罗斯利用 IE 浏览器中存在的一个叫做 WMF 的漏洞来攻破那些毫无戒心的 Web 冲浪者的计算机，并在其中安装 bot 软件。这影响到数千台计算机的攻击，其发生的主要原因之一就是这个漏洞发现后，在 Microsoft 公司制作出修补程序之前，漏洞就被大肆利用了。对于使用一份具有漏洞的软件的客户来说，脆弱期(漏洞从被发现到厂商提供出修补程序之前的这段时间)是最危险的时间。安全测试旨在消除这些软件安全漏洞。通过负责任的漏洞公布，脆弱期能够缩短。由于漏洞是被“坏蛋们”发现的，我们不能阻止脆弱期的存在，但我们可以预防漏洞本身的出现。

Internet 上的攻击活动在近几年发生了变化，变得在本质上更加罪恶、更加阴险。三四年以前，利用软件漏洞的人大多热衷于借助攻击别人来为自己扬名，其手段通常是编写并发布病毒和蠕虫，而不是图财。当前的发展趋势是利用漏洞来为罪犯们收集银行、公司和政府的认证信息，然后用于犯罪。现在，当家庭计算机被攻破，它们往往被用于“钓鱼(phishing)”攻击或者传递垃圾信息。我们使用的软件并没有在安全质量相关方面得到极大的改进，而当前攻击者的动机却比以往更加明确，那就是谋财。

作为一名消费者，你应该要求你的软件供应商将安全性集成到他们的开发过程中，要求了解他们的过程和工具。如果这个软件是用于保护财务数据或其他敏感信息的，要求出示第三方对该供应商的软件安全质量评测结果。如果业界不能主动接受安全质量标准，或者不能通过政府强制执行安全质量标准的话，消费者就要考虑要求出示特定安全质量等级的证明。直到建筑检测人员确认该建筑物符合规范，人们才会住进这栋楼。汽车以及其他具有潜在危险性的产品同样也要求必须进行独立的安全测试。现在轮到软件消费者来进行这种测试了，更有可能是请可信的第三方来进行测试。然而，大多数公司只进行了非常有限的第三方应用程序安全测试。从前，这种测试是由联邦组织来完成的，但现在已经扩展到一些较大的金融机构或企业。但即便是这样，在这些机构或企业中，也缺少受过充分的黑盒安全测试训练的人员，从而并不能完全实施安全测试。由于软件厂商自身工作做得不够，就需要消费者来进行他们自己的安全测试(或雇佣第三方来测试)。

计算机用户习惯于把广告软件和间谍软件仅仅当作低级的、用来提供近期大减价信息的令人讨厌的东西。据 Reuters 所说[5]，一个加利福尼亚人被联邦法院指控创建了一个由被劫持计算机组成的类机器人网络(亦即 bot 网络——译者注)，通过这个网络强制安装讨厌的广告软件从而帮助

他和两个同伙从中获利 10 万美元。

广告软件的背后操纵者总是从低级的令人讨厌的东西那里赚钱。然而，键盘记录间谍软件却有更加险恶的威胁。如果说广告软件只是些讨厌的东西，而键盘记录间谍软件却被黑客用来得到你的金融站点登录名称和密码，并在线转走你的钱，这可就是个比较大的问题了。

Symantec 公司每 6 个月发布一次的 Internet 安全威胁报告对 Internet 安全活动进行分析和讨论。其范围包括 Internet 攻击、漏洞、恶意代码以及未来的安全趋势等内容。编写本书的时候最新一期的安全威胁报告中涵盖了 2005 年前 6 个月的情况，其中标识出了 Internet 安全威胁领域内的发展变化轨迹[6]。攻击者已经从对网络周边的大型的多目标攻击转移到针对客户端目标展开的较小的更集中的攻击，这些客户端目标如 Web 浏览器、媒体播放程序以及即时通信程序等。新的威胁领域很可能被那些新出现的威胁所占领，如 bot 网络、可定制的模块化恶意代码以及针对 Web 站点的锁定目标攻击等。和传统的攻击活动不同的是，当前的许多威胁动机非常明确，就是为了获利。攻击者通常企图进行犯罪活动，比如窃取身份资料、敲诈或欺骗等等。

众多安全分析员正努力工作，以便能把所有可能被利用的安全漏洞及时地告知公众。例如，Symantec 公司的 Deepsight 威胁管理系统[7]（Deepsight Threat Management System）提供了这种情报，其中涵盖了完整的安全威胁生命周期，即从最初发现威胁到公开漏洞，再到攻击活跃起来的整个过程。

BugTraq[8]是一个大型、完全公开的邮件列表，用来详尽讨论并公布计算机安全漏洞。BugTraq 是整个 Internet 上安全团体的基石。

SecurityFocus 安全漏洞数据库为安全从业人员提供了各种平台和服务方面安全漏洞的最新信息。

这还仅仅是专门跟踪匿名流量的众多资源中的一少部分。对海量信息（恶意代码、安全风险、域名失效警报等）进行监控，在某种程度上是势在必行的。

在理想的情况下，各 Web 站点都应该能够抵御来自 Internet 使用者的恶意攻击，保护站点和其用户机密数据的安全。Web 应用程序的最终用户都应该对该程序的安全性有信心，即他们可以放心使用这个系统，而不用担心非授权用户能够访问到存储在这个 Web 站点上的交易信息或个人信息。

素材的覆盖范围和本书的组织结构

本书囊括了应用程序和网络安全分析与测试方面的内容，提供了读者急需的技术深度。

许多其他图书所述内容依赖于专利权软件的使用，这些软件的价格高达上千美元，而本书中

所讲的工具都是可以免费得到的。本书为 QA 从业人员提供了相关工具的综述，向进行大量烦琐工作的测试人员介绍了一些很有价值的策略。

本书将展示如何针对那些最有意义、最重要和/或最具风险的方面进行测试，从而避免将耗费不菲的测试资源用于那些较小可能的攻击想定。

通常，大多数人的意见似乎仍然认为需要有专家来指导安全测试。“安全测试任务只能由经过专门培训的人员来完成，因为一般的质量保障测试人员没有得到完成这项工作所需的训练”，在 2006 年 2 月召开的 RSA 会议上，参加座谈的一组公司安全主管、学术界人士和专业的软件开发人员得出了这样的结论[9]。

关于安全测试是谁的职责的问题，存在很多混乱的认识，但是，很多人都负有这种责任。

本书概括了安全的软件开发生命周期(Secure Software Development Lifecycle，SSDL)的概念。与软件开发生命周期相对应，SSDL 由 6 个阶段构成。本书的每章将详细讲述 SSDL 的各个阶段(第 3 章将概述 SSDL)，并将讨论在此生命周期中，及早归并和提出安全问题的重要性。本书将谈到每个阶段的安全任务、角色以及每个团队成员的职责，从而有助于澄清任务到底是什么，究竟谁真正对安全测试负责。

必须尽早定义并/或理解有关的安全原则(guideline)、安全规则(rule)和安全规章(regulation)。人们认为需要遵循的安全原则包罗众多。无论你的软件开发安全标准是遵循某种标准(如 VISA 标准、HIPPA 标准或 SOX 标准等)，还是自己内部制定的标准，都应该有一个安全策略，作为 SSDL 的基准来使用。

在业务分析和需求整理阶段，安全需求往往被忽略，然而，应该在这个阶段对其进行定义。在原型/设计/架构阶段，安全小组通过对系统架构复审和威胁模型构建(将在第 4 章进行讨论)进行指导，从而对安全提供支持。通过完成这些工作，指出所有潜在的安全漏洞，并确定出这个应用程序风险最高的部分。

安全的编码原则有助于防止代码中出现纰漏(将在第 2 章进行讨论)，这些原则需要在软件开发阶段坚持遵守。白盒/黑盒/灰盒测试技术(将在第 5 章进行讨论)用于集成和测试阶段。

本书第二部分的重点放在在网络上发现应用程序并对其进行攻击。然后，将讨论各种攻击方法，包括针对 Web 站点授权功能模块的攻击。首先我们将学习如何执行 SQL 注入攻击，然后将看到针对 Web 服务器的其他常见攻击(第 6 章、第 7 章、第 8 章)，最后我们将学习如何使用各种代理来测试应用程序的各种安全漏洞(第 9 章和第 10 章)。第二部分的每一章都会对相应的测试攻击模式进行总结，QA 人员或测试工程师可以以此为起点制定其测试大纲或测试清单。

判定可利用性(在第三部分中讨论)是在整个生命周期中都需要做的事情，而结果要在系统成

品开发和维护阶段才会定案。这时，应做好补丁管理计划。在整个测试的生命周期中，应对安全计划复审和评估活动提供指导，以便进行持续的改进活动。随着安全测试的开展，使用整个过程中收集的度量标准进行的最终复审和评估需要得到指导，以便得出满意的、明智的决策。

本书分为三个主要部分。

第一部分

第一部分讨论了一些现实情况，即安全测试需要一种范式来替换传统的测试方法，并需要指导安全测试人员学会像攻击者那样思考。本部分将讨论安全漏洞是怎样藏到软件中来的，还将讨论威胁建模及白盒、黑盒和灰盒测试是怎样用于找出这些安全漏洞的。

第一部分的第3章展示了构成安全的软件开发生命周期(SSDL)的各部分是怎样贯穿全书来描述的。

- 第1章，设身处地：从传统软件测试转变的一个范式。本章讨论通过设法攻破软件来促进软件安全的有关话题。这要求测试人员像侦探那样工作，并迫使测试的范式从传统测试中转变。本章提供了一些高级安全测试策略。安全测试需要一种独特的测试思想。安全测试人员不仅应该把自己当作一名验证者，还应该把自己当作一名攻击者，也就是必须进行一些探测工作，以判定应用程序最为薄弱的部分，并设计相应的攻击。因为并不是所有的攻击都是破坏性的，所以测试人员必须进行评价并排定安全测试想定执行的优先顺序，并纠正所有未公开的潜在安全漏洞。
- 第2章，漏洞是怎样藏到软件中的。本章将对各种安全漏洞进行讨论，并探讨如何防止这些漏洞偷偷藏到你的程序中(如同设计漏洞与实现漏洞那样)。设计漏洞是一种设计错误，它使得程序不可能安全运转，而无论代码编写人员实现得多么好都无济于事。与之不同，实现漏洞是由软件实际编码中存在的安全错误(bug)引起的。本章将讨论如何能够避免各类安全错误在不知不觉之中“溜进”你的项目。
- 第3章，安全的软件开发生命周期。本章将讨论在安全的软件开发生命周期中，及早介入安全测试和提出安全问题的重要性。在传统的软件开发生命周期中，安全测试往往只是一种事后反应。然而，将安全测试工作延迟到软件开发完成之后才进行，这是一种坏习惯。本章内容将涉及及早开展软件安全测试工作，并将讨论如何将安全性需要引入到软件开发生命周期中，且从最早的阶段就开始引入这种要求。
- 第4章，基于风险的安全测试：通过威胁建模排定安全测试的优先顺序。本章将讨论服务于软件安全测试的威胁模型的使用。将讨论威胁模型建立的各个步骤，如信息采集(包括与架构师会谈或指导运行时审查)、贯彻建模过程、排定优先级以及使用缓解策略。

- 第5章，白盒、黑盒和灰盒测试。本章将讨论各种安全测试方法及各自的优缺点。其中包括关于如何搭建一个安全测试实验环境的内容，以及所需的软件安全测试工具软件的信息。

第二部分

本部分开始讲述能使测试人员像攻击者一样思考的一些技术，详细讨论如何进行攻击。诸如常见的网络故障注入、应用程序攻击、代理服务器以及本地错误注入等攻击都将论及。

- 第6章，常见的网络故障注入。本章重点内容是在网络上发现应用程序并对其展开攻击。首先，我们将学习如何找出应用程序的网络痕迹，然后将讨论各种攻击方法，最后将讨论中介随机侦测程序的使用。
- 第7章，Web应用程序：会话攻击。本章重点介绍针对Web站点的会话(session)管理功能展开的攻击。我们将学习密码的暴力攻击、cookie分析以及跨站点执行脚本攻击。
- 第8章，Web应用程序：常见问题。本章将向你展示如何执行SQL注入攻击。然后，本章还将介绍针对Web服务器的其他一些常见攻击。
- 第9章，Web代理：使用WebScarab。本章介绍免费的Web代理软件WebScarab的使用及其特点。
- 第10章，实现定制的侦探工具。本章介绍如何使用快速原型开发语言来实现一个定制的侦探工具。
- 第11章，本地故障注入。本章详细介绍一些专用于测试本地应用程序攻击面相关的技术，这些攻击面包括ActiveX接口、文件格式、命令行参数以及共享内存段等。

第三部分

本书的最后这部分内容将提供一种分析方法，并展示如何判定漏洞的可利用性。

- 第12章，判定可利用性。本章描述如何基于漏洞对应用程序的潜在影响来确定其影响范围和严重程度。你将看到安全漏洞是怎样在处理器指令层上触发的，这样，你就可以辨别应用程序的崩溃是否会导致代码的执行。

读者对象

软件安全和测试工程师需要更快更彻底地开展安全测试，针对这种需要，本书讲述了一些实用技巧，并提供了开展工作所需的知识。这些技巧同样可以用于那些负责安全测试(除了单元测试和集成测试之外)的软件开发人员。本书还对软件项目经理的需要提供了支持，这些项目经理负责整个项目的安全测试。本书为他们提供了诸如安全测试的目标和结果、如何确定在哪里和是