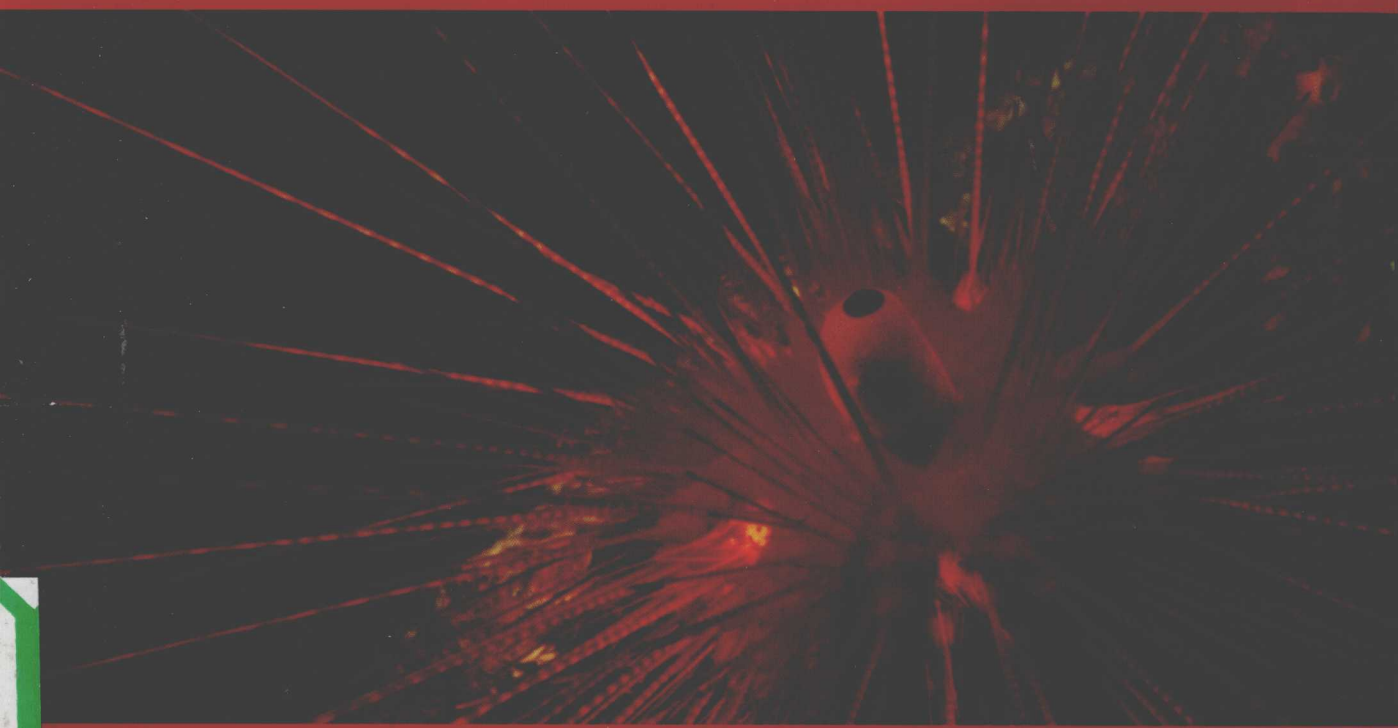
HZ BOOKS

Modern C++ Design
C++
设计新思维

C 和 C++ 安全编码

Secure Coding in C and C++



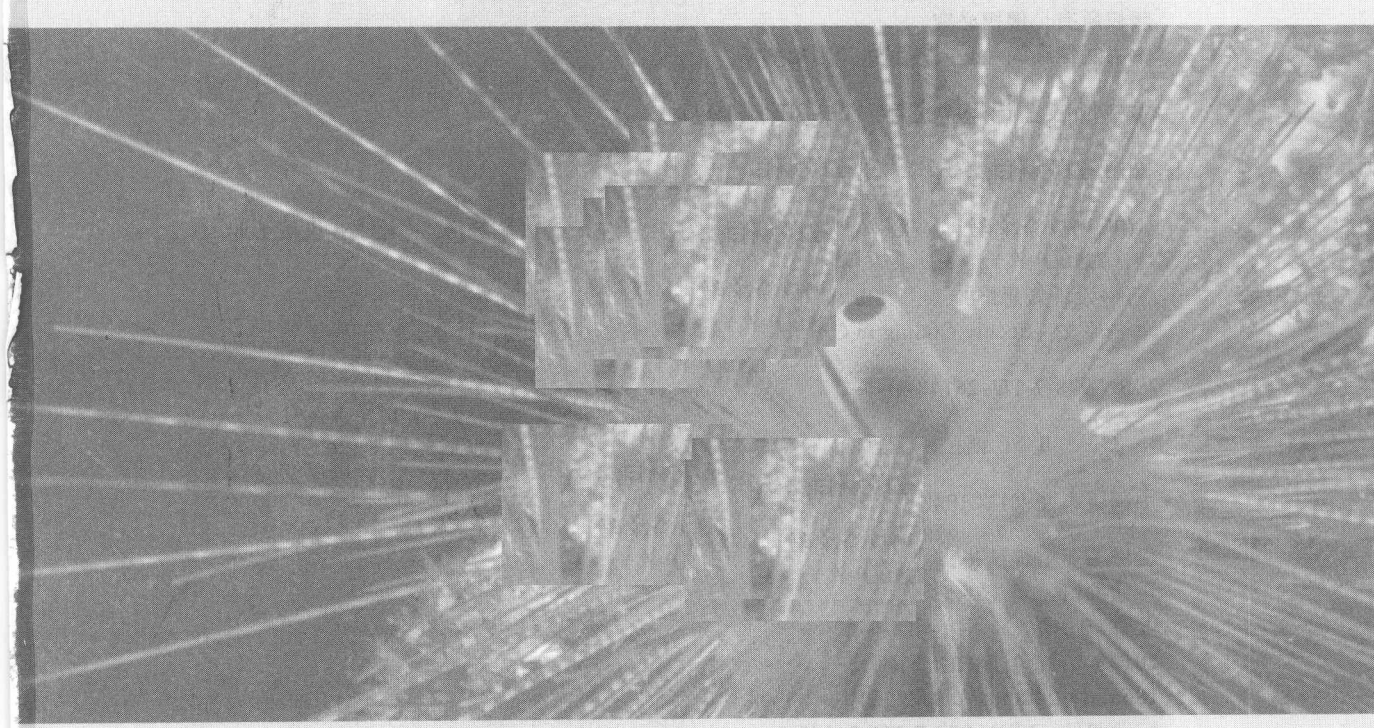
(美) Robert C. Seacord 著
荣耀 罗翼 译

机械工业出版社
China Machine Press

C++ 设计新思维

C 和 C++ 安全编码

Secure Coding in C and C++



(美) Robert C. Seacord 著
荣耀 罗翼 译

机械工业出版社
China Machine Press

本书是关于C和C++安全编码的著作。本书介绍了C和C++程序中导致危险的、破坏性的基本编程错误，包括在字符串、指针、动态内存管理、整数、格式化输出、文件I/O等中的漏洞或缺陷。本书还提供了对这些编程错误的深入剖析，并给出缓解策略，以减少或消除恶意利用漏洞的风险。

本书适合C/C++程序员、软件安全工程师参考。

本书版权登记号：图字：01-2006-1947

图书在版编目（CIP）数据

C和C++安全编码 /（美）西科德（Seacord, R. C.）著；荣耀等译. —北京：机械工业出版社，2010.1

（C++设计新思维）

书名原文：Secure Coding in C and C++

ISBN 978-7-111-26148-3

Ⅰ. C…　Ⅱ. ① 西…　② 荣…　Ⅲ. C语言—程序设计　Ⅳ. TP312

中国版本图书馆CIP数据核字（2009）第015137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

北京京师印务有限公司印刷

2010年1月第1版第1次印刷

186mm × 240mm · 15.25印张

标准书号：ISBN 978-7-111-26148-3

定价：45.00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

购书热线：（010）68326294；88379649；68995259

投稿热线：（010）88379604

读者信箱：hzjsj@hzbook.com

译者序

现代社会离不开计算机软件系统的顺畅运行。然而总有一些黑客或骇客为了政治、经济或纯粹的个人满足目的攻击软件系统，其中大部分得逞的攻击是因为软件系统自身存在漏洞。尽管事后的补救措施可以在一定程度上减少损失，然而如果在软件创建之初就避免引入漏洞，攻击者就无法轻易获得下手的机会。避免软件引入漏洞首先是程序员的责任。C和C++语言的天性导致程序员必须小心谨慎才能编写出没有漏洞的程序。

Robert Seacord是SEI CERT/CC资深漏洞分析师，具有二十多年的编程经验，洞悉C和C++在软件项目中的应用状况，这本书可以说专门为C和C++程序员而写（尽管其他语言的开发人员或技术管理者也可以从阅读本书中深深受益），并不要求读者事先掌握多少计算机安全背景知识。作者显然善于讲解复杂的技术，深入浅出地阐述了软件中的各种安全漏洞及成因，给出了很多著名的安全漏洞例子，并揭示黑客如何利用程序中的漏洞侵入并破坏系统。

保障软件系统的安全主要有两种方式：配置安全和编程安全，前者面向系统管理员和系统用户，后者则要求程序员掌握安全的编程技术。也许完全消除漏洞是难以达到的目标（世界上根本不存在没有bug的软件），但程序员可以组合使用多种方式尽可能消除软件缺陷。作者展示了许多不安全的编程习惯和相应的改进措施，推荐了有效的软件工具和经过验证的方法，有助于我们编写出更健壮、更不容易遭受攻击的软件系统。

每一位开发商业和项目软件系统的C或C++程序员都应该阅读本书，尤其是那些编写会被很多用户通过互联网下载安装的软件的程序员，那些编写控制电力、电信、金融、军事、航空航天等关键设施的软件系统的程序员，以及其他所有认为自己编写的软件对他人很重要的程序员。对于有漏洞的软件系统，即使没有遭受黑客的恶意攻击，出乎意料的输入或操作也可能会导致灾难性的后果。由于每一个安全漏洞也可能是导致程序崩溃或产生错误输出的缺陷，因此，即便没有黑客或骇客盯上我们的软件，程序员也有责任保证软件系统没有漏洞。

使用Java或C#之类的语言编写软件无疑可以少引入安全漏洞，然而C和C++语言的地位无法取代。为何使用C和C++语言编程？因为对效率和灵活控制底层的苛求。如果你是一名C或C++程序员，希望本书在带给你宝贵的技术和良好阅读体验的同时，也能为你带来危机感——现在就复审以前的代码是否存在安全漏洞，并避免在未来的代码中引入安全漏洞。毕竟编写无漏洞的软件需要的不单是技术，更有高度的责任感。

感谢读者朋友给予的一贯支持，你们的批评和赞扬对我来说同样重要。感谢初译者罗翼先生作出的贡献。感谢机械工业出版社华章公司的陈冀康先生、周茂辉编辑以及所有其他为本书

的出版作出贡献的不知名的人们。感谢导师王建东老师的理解与支持。感谢朱艳和荣珅，你们是我生活的全部。

祝各位阅读快乐！

荣耀
2009年5月
南京师范大学中北学院
www.royaloo.com

序　言

现代社会对网络化软件系统的依赖与日俱增，以这些系统为目标的攻击数量亦与日俱增。形形色色的针对政府、公司、教育机构以及个人系统的攻击，已经导致诸多严重的后果：敏感数据丢失或被破坏，系统受损，生产效率降低，以及财务损失，等等。

尽管今天互联网上的攻击行为大多不过是恶作剧，然而越来越多的证据表明，罪犯、恐怖分子以及其他心怀叵测的人已将软件系统中的漏洞视作达成其不可告人目的的手段。近年来，每年发现的软件漏洞逾4000个。这些漏洞的成因包括：在设计和实现方面对如何保护系统考虑不周；在开发实践方面对消除会导致安全缺陷的实现瑕疵关注不够。

伴随着软件漏洞日益增加，攻击手法也与时俱进，变得越发老练、有效。入侵者在发现软件产品中的漏洞后，可以迅速开发出利用漏洞的脚本，继而使用这些脚本威胁计算机的安全。更糟糕的是，还将这些脚本与其他攻击者共享。这些脚本与一些“自动扫描网络以窥探脆弱系统”的程序联合起来，攻击那些脆弱系统，危害其安全，甚至利用它们进一步传播攻击。

由于每年都会发现大量的漏洞，系统管理员为了给既有系统打补丁日益疲于奔命。打补丁有时并非易事，并可能会导致意想不到的副作用。在厂商发布一个安全补丁后，可能需耗时数月乃至数年的时间，才能有90%～95%的受影响计算机完成修补工作。

互联网用户已经严重依赖于这样一种解决方式：互联网社群作为一个整体，对发生的安全攻击快速作出反应，以确保将损害降至最低，并将攻击迅速击溃。然而，显而易见，如今的“反应式解决方案”已经达到其效力的极限。虽然个体响应组织均为将漏洞处理过程精简化和自动化而努力工作，然而今天商业软件产品中的漏洞数目，已经多至只有财力雄厚的组织才能应对漏洞修复工作，其他组织实质上已心有余而力不足。

没有任何证据表明大多数产品中的安全问题得到了改善，很多软件开发者尚未很好地理解从各种漏洞成因中获得的教训，或者尚未运用能够起作用的缓解策略，CERT/CC不断在同一软件产品的更新版中发现早期版本即已存在的同类漏洞就是明证。

这些因素交织在一起，预示着我们完全可以相信，很多攻击甚至会在我们所实际希望的最短响应时间内造成巨大的经济损失和服务崩溃。

虽然积极且协调的反应仍然不可或缺，但我们还必须构建出不易被破坏的更安全的系统。

关于本书

本书致力于解决C和C++中已经导致危险的、破坏性的常见软件漏洞的基本编程错误。这些漏洞自CERT 1988年创立以来就记录在案。针对导致这些漏洞的编程错误，本书出色地给出了深度工程分析和缓解策略，可以富有成效地减少或消除漏洞被恶意利用的风险。

自Robert于1987年4月加入SEI以来，我就一直与他共事。Robert是一位知识渊博、技术熟练

的软件工程师，在软件漏洞细节分析和表达洞察成果方面卓尔不凡。作为研究结果，本书细致而精确地分析了软件开发者面临的常见问题，并提供了实用的解决方案。Robert在软件开发方面具备的宽广背景，也使得他擅长在性能、易用性以及其他在开发安全代码过程中必须考虑的质量特性之间作出权衡。除了Robert外，本书还凝聚了CERT积累和提炼的知识，以及CERT/CC漏洞分析小组、CERT操作人员、SEI编辑和后勤人员的杰出工作。

Richard D.Pethia

CERT主管

前　言

1988年11月爆发的Morris蠕虫事件造成当时全球十分之一的互联网系统陷入瘫痪，作为对该事件的响应，当月美国国防部高级研究计划局（Defense Advanced Research Projects Agency，DARPA）成立了CERT协调中心（CERT Coordination Center，CERT/CC）。CERT/CC位于宾夕法尼亚州匹兹堡市的软件工程研究院（Software Engineering Institute，SEI）内，这是一个由美国国防部发起的研发中心，受联邦政府资助。

CERT/CC最初的工作重点是对各种网络事件作出快速响应和分析。这里所说的事件既包括得逞的攻击（如系统受损与拒绝服务等），也包括未得逞的攻击企图、探测和扫描。自1988年以来，CERT/CC共已接到逾22 665个报告计算机安全事件或咨询有关信息的热线电话，已处理总共逾319 992起计算机安全事件，而且每年报告的事件数目呈持续增长的态势。

虽然对事件作出及时响应必不可少，然而这还不足以保护互联网和互联的信息系统的安全。分析表明，大部分计算机安全事件是由于特洛伊木马、社会工程学（social engineering）以及软件漏洞利用（exploitation）所造成的，包括软件缺陷、设计决策、配置决策以及非预期的系统间交互等。CERT/CC监控漏洞信息的公共来源并经常性地接到漏洞报告。自1995年以来，CERT已经收到超过16 726份漏洞报告。每当收到一份报告，CERT/CC就会分析报告所述的可能的漏洞，并与软件制造者协作，通知其产品中存在安全缺陷，促进并追踪其对问题的响应[⊖]。

和事件报告相似，漏洞报告也以惊人的速度持续增长[⊖]。虽然对漏洞的管理抑制了这一进程的发展，然而对于解决互联网和信息系统的安全问题来说，这同样远远不够。为了解决日益增加的漏洞和事件问题，必须采取相应的措施：在源头予以有效地控制它们，即必须在软件的开发阶段和随后的维护工作中就避免引入软件漏洞。对现有漏洞的分析表明，大部分漏洞都是由少数根本原因所导致。本书的目标就在于告诉开发者有关这些根本原因的知识，并介绍避免引入漏洞的措施。

读者

对任何采用C和C++开发或维护软件的人来说，本书都是非常有价值的。

- 对C/C++程序员而言，本书将教你如何识别会导致软件漏洞的常见编程错误，理解这些错误是如何被利用的，并以安全的方式实现解决方案。
- 对软件项目管理者而言，本书将教你识别软件漏洞的风险及导致的后果，对进行安全的软件开发给出指导。

⊖ CERT/CC与超过700名软件、硬件开发者保持定期互动交流。

⊖ 参见http://www.cert.org/stats/cert_stats.html以获得最新统计信息。

- 对计算机专业学生而言，本书将教你有用的编程实践，帮助你避免不良的开发习惯，使你能够在职业生涯中开发出安全的程序。
- 对安全分析师而言，本书对常见的漏洞提供了细致的描述，并提供了检测漏洞的方法和实用的规避措施。

本书组织和内容

本书提供了C和C++编程中关于安全实践方面的实用指导。安全的程序需要安全的设计，然而，如果开发者不了解C和C++编程中许多固有的安全陷阱，那么即便是最佳设计也可能最终导致不安全的程序。本书对C和C++中常见的编程错误提供了详细的解释，并描述了这些错误是如何导致有漏洞的代码的。本书专注于讨论C和C++编程语言以及相关库固有的安全问题，而不强调和外部系统（如数据库服务器和Web服务器等）的交互中牵涉的安全问题。这些方面话题丰富，应单独讨论。本书的用意在于，对于涉及开发安全的C和C++程序的任何人都有所裨益，而不管具体开发何种应用。

本书围绕着软件工程师经常实现的、可能会导致安全问题的功能（例如格式化输出和算术运算等）组织内容。每一章都描述了一些会导致漏洞的不安全编程实践和常见的错误，这些编程缺陷又会被如何利用，这种恶意利用所导致的后果，以及替代的安全做法等。对于软件漏洞的根源问题，例如缓冲区溢出、整型范围错误和无效的格式字符串等，都有详细的解释。每一章都包含检测既有代码中漏洞的技术，以及如何安全地实现所需功能的措施。

本书包含以下各章：

- **第1章**：概述问题，介绍安全术语和概念，并对为何C和C++程序中存在如此多的漏洞发表看法。
- **第2章**：描述C和C++中的字符串操作、常见的安全缺陷以及由此导致的漏洞（包括缓冲区溢出攻击和栈粉碎攻击），还介绍了代码注入（code injection）和弧注入（arc injection）两种漏洞利用方式。
- **第3章**：介绍了任意内存写（arbitrary memory write）漏洞利用方式，它允许攻击者对内存中任意位置的一个地址进行写操作。本章描述了这些漏洞利用方式如何用于在受害机器上执行任意的代码。由“任意内存写”导致的漏洞在稍后的章节中讨论。
- **第4章**：描述动态内存管理，讨论了动态分配的缓冲区溢出、写入已释放内存，以及重复释放（double-free）漏洞。
- **第5章**：讨论整数安全问题（即与整数操作相关的安全议题），包括整数溢出、符号错误以及截断错误等。
- **第6章**：描述格式化输出函数的正确和错误的用法。因对这些函数的错误使用所导致的格式字符串和缓冲区溢出漏洞都有讨论。
- **第7章**：描述了和文件I/O相关的常见漏洞，包括竞争条件（race condition）和TOCTOU（time of creation, time of use）漏洞。
- **第8章**：推荐了一些可以整体改善C/C++应用程序安全性的具体开发实践。这些建议是对

每一章中用于解决特定漏洞问题的推荐做法的补充。

本书包含了数百个安全和不安全的代码示例以及漏洞利用示例。尽管其中有一些涉及与其他语言的比较，然而几乎所有例子都是C和C++的。这些例子都在Windows和Linux操作系统上验证过。除非另有说明，微软Windows程序示例均使用Visual C++ .NET编译，在运行Intel Pentium 4处理器的Windows 2000 专业版平台上测试。Linux程序示例则使用GNU gcc/g++编译，在运行Intel Pentium 4处理器的Red Hat Linux 9[⊖]平台测试。

尽管具体的示例程序通常已在一个或多个环境下编译和测试过，然而漏洞都被予以评估，以决定它们是具体相关于以下因素，或者具有包含以下因素的普遍性：编译器版本、操作系统、微处理器、适用的C或C++标准、小端表示法（即：低位在前、高位在后）或大端表示法（即：高位在前、低位在后）架构，以及执行栈架构等。

本书集中讨论最常导致软件漏洞的C和C++编程缺陷。尽管如此，限于篇幅，本书并未涵盖漏洞的每一个可能来源。与本书有关的附加信息、更新信息、事件以及新闻参见http://www.cert.org/books/secure-coding/。书中讨论的漏洞也是参照引用来自US-CERT漏洞数据库（参见http://www.kb.cert.org/vuls/）的实际例子。

致谢

在此我要感谢为本书问世作出贡献的每一个人。首先要感谢Noopur Davis、Chad Dougherty、Fred Long、Rob Murawski、Nancy Mead、Robert Mead、Dan Plakosh、Jason Rafail以及David Riley，他们为本书作出了直接的贡献。还要感谢研究人员的贡献，他们是Omar Alhazmi、Mathew Conover、Jeffery S.Gennari、Oded Horovitz、Poul-Henning Kamp、Doug Lea、Yashwant Malaiya以及John Robert。

我要感谢SEI和CERT/CC的管理层，感谢他们的鼓励以及对我劳动成果的支持。他们是Jeffrey Carpenter、Shawn Hernan、Jeffrey Havrilla和Rich Pethia。

还要感谢编辑Peter Gordon和Addison-Wesley的其他一些职员，他们是Jennifer Andrews、Kim Boedigheimer、Chanda Leary-Coutu、John Fuller、Eric Garulay、Marie McKinley以及Elizabeth Ryan。

我还要感谢审稿人给予的深思熟虑的评论和富有洞察力的见解，他们是Tad Anderson、William Bulley、Corey Cohen、Will Dormann、Robin Eric Fredericksen、Amit Kalani、John Lambert、Jeffrey Lanza、David LeBlanc、William Fithen、Gary McGraw、Michael Howard、Ken MacInnis、Randy Meyers、Dave Mundie、Patrick Mueller、Craig Partridge、Brad Rubbo、Time Shimeall以及Katie Washok。

我要感谢CERT/CC小组其他成员的支持和协助，没有他们我永远都不可能完成本书。最后（但并非最不重要），感谢单位内部的编辑和图书管理员，他们的帮助使得本书的最终问世成为可能，他们是Rachel Callison、Pamela Curtis、Len Estrin、Eric Hayes、Karen Riley、 Sheila Rosenthal以及Barbara White。

⊖ 访问http://www.linux.org/dist/list.html以获得Linux发行版的分类清单。

作译者简介

作者简介

Robert C.Seacord是宾夕法尼亚州匹兹堡市SEI（Software Engineering Institute，软件工程研究院）的CERT/CC（CERT/Coordination Center，CERT协调中心）高级漏洞分析师。CERT/CC定期对软件漏洞报告进行分析，并且评估互联网及其他关键的基础设施可能遭受的风险，此外还从事其他一些与安全有关的研究活动。作为一名涉猎广泛的技术专家，Robert还是《Building Systems from Commercial Components》（Addison-Wesley，2002）和《Modernizing Legacy Systems》（Addison-Wesley，2003）的合著者，并发表了40多篇论文，领域涉及软件安全、基于组件的软件工程、基于Web的系统设计、遗留系统的现代化改造、组件仓库与搜索引擎以及用户界面设计与开发等。Robert于1982年起在IBM开始职业编程生涯，从事通信和操作系统软件研发、处理器开发以及软件工程。Robert还为X 协会（X Consortium）工作，为CDE（Common Desktop Environment，公共桌面环境）和X Window 系统开发和维护代码。他还积极参与JTC1/SC22/WG14的C语言国际标准化工作组工作。

译者简介

荣耀 是南京师范大学副教授，主研方向为企业级应用架构与软件工程，已出版著译作品包括《ASP.NET 2.0实战起步》、《.NET大局观》、《C++ Templates全览》、《Imperfect C++中文版》、《C++必知必会》、《C++编程你也行》以及《Windows Forms 程序设计》等，并在各类期刊发表技术文章约30篇，他的个人网站是www.royaloo.com。

罗翼 现供职于某搜索互联网企业工程技术部门。从事系统程序（以及各种古怪脚本）的开发、调试、优化等Linux/UNIX平台上C/C++程序员的工作。业余时间主要贡献给了 Slackware系统的各个犄角旮旯以及各种新奇、古怪、败家的消费电子硬件。他曾与人合著过《加密与解密》等计算机科学科普读物。你可以通过luoyi.ly@gmail.com 与他联系。

目　录

第1章 夹缝求生

只有上帝的身边没有邪恶。

——Sophocles, Fragments, 1.683

计算机系统并不易受攻击，易受攻击的是我们人类，只不过遭受的攻击通过计算机系统完成罢了。

2003年8月11日爆发的W32.Blaster.Worm蠕虫病毒很好地例证了软件中的安全缺陷是如何让我们变得易受攻击的。Blaster可以在毫无用户参与的情况下感染任何一台连接到互联网上未打补丁的计算机系统。微软提供的数据显示，至少有800万个Windows系统被该蠕虫感染[Lemos 04]。Blaster的主要破坏力在于使用户无法正常使用自己的机器，并且能够渗透整个局域网，感染的用户必须设法删除该蠕虫并且升级系统才能正常工作。

Blaster蠕虫作恶前后的一系列事件揭示了软件厂商、安全研究人员、发布漏洞利用代码者以及恶意攻击者之间错综复杂的关系。

首先，LSD（Last Stage of Delirium）研究小组发现了RPC[一]（可用于通过TCP/IP交换信息）中存在一个缓冲区溢出漏洞，成因在于对“畸形”信息的错误处理。该漏洞影响监听RPC端口的一个分布式组件对象模型（DCOM）接口，后者处理客户机发送给服务器的对象激活请求。对该漏洞的成功利用允许攻击者在受感染的系统上以本地权限执行任意的代码。

在这种情况下，LSD小组遵循了负责任的披露原则，在公开该漏洞前与软件厂商进行了合作，寻求解决问题的办法。2003年7月16日，微软发布了微软安全公告MS03-026[二]，LSD发布了一个特别报告[三]，CERT/CC则发布了一个漏洞说明VU#568148[四]，详细描述了该漏洞，并提供相应的补丁和应急方案的信息。第二天，CERT/CC还发布了名为“Buffer Overflow in Microsoft RPC”（微软RPC缓冲区溢出）的CERT公告CA-2003-16[五]。

9天后，也就是7月25日，一个名为Xfocus的安全研究小组发布了该漏洞的利用代码。Xfocus自称是“一个在1998年创立于中国的非营利和自由技术组织，致力于研究和展示网络服务和通信安全方面的弱点”。实质上，Xfocus对微软提供的补丁程序进行了逆向工程，研究了该漏洞的原理，并且开发了相应的攻击该漏洞的方法，并将其公之于众[Charney 03]。

HD Moore（Metasploit项目的创始人）改进了Xfocus的代码，使其可以对更多版本的操作系

[一] RPC（Remote Procedure Call，远程过程调用）是一种进程间通信机制，允许一台计算机上运行的程序执行另一台远程系统上的代码。微软的实现以OSF（Open Software Foundation，开放软件基金会）的RPC协议为基础，并添加了微软独有的一些扩展特性。

[二] 参见http://www.microsoft.com/technet/security/bulletin/MS03-026.mspx。

[三] 参见http://lsd-pl.net/special.html。

[四] 参见http://www.kb.cert.org/vuls/id/568148。

[五] 参见http://www.cert.org/advisories/CA-2003-16.html。

统生效。很快就有漏洞利用工具发布了，使得黑客可以通过IRC网络发送命令。8月2日就已经发现了这些攻击的蛛丝马迹[de Kere 03]。

由于DefCon黑客大会将于8月2日至3日召开，因此大家普遍预计将会有针对该漏洞的蠕虫发布（并不是说参加DefCon的人会这么做，而是由于该大会会引起人们对黑客技术的关注）。DHS（Department of Homeland Security，美国国土安全部）于8月1号发布了一个预警，FedCIRC（Federal Computer Incident Response Center，联邦计算机事故响应中心）、NCS（National Communications System，美国国家通信系统）以及NIPC（National Infrastructure Protection Center，美国国家基础设施保护中心）也在对漏洞利用行为积极进行监测。8月11日，也就是补丁程序发布后的第26天，Blaster蠕虫首次被发现在互联网上传播。24小时后，Blaster感染了336 000台计算机[Pethia 03a]。截至8月14日，Blaster已经感染了超过100万台计算机，在高峰期，它每小时感染100 000个系统[de Kere 03]。

Blaster是一个通过TCP/IP进行传播的贪婪的蠕虫，它利用了Windows的DCOM RPC接口中的一个漏洞。当Blaster执行的时候，它首先检测计算机是否已感染以及蠕虫是否正在运行，如果是这样，它就不会重复感染该计算机，否则，Blaster将：

```
"windows auto update" = "msblast.exe"
```

加入注册表的

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
```

键中，这样，每当启动Windows，该蠕虫程序都会运行。接下来，Blaster产生一个随机的IP地址，并试图感染具有该地址的计算机。该蠕虫利用Windows 2000或Windows XP上的DCOM RPC漏洞通过TCP 135端口发送数据。Blaster监听UDP端口69，接收可以通过DCOM RPC 漏洞利用连接的机器的请求。一旦接收到请求，随即将`msblast.exe`文件发送到对方计算机并且执行该蠕虫[Hoogstraten 03]。

蠕虫利用`cmd.exe`建立一个远程shell进程“后门”，监听TCP端口4444，从而允许攻击者在受危害的系统上发出远程命令。Blaster同时还试图对Windows Update网站发起DoS（denial-of-service，拒绝服务）攻击，以阻止用户下载补丁。蠕虫会在一个特定的日期以SYN洪水（SYN flood）[⊖]的形式对`windowsupdate.com`的80端口发起DoS攻击。

即便Blaster不能成功地感染目标系统，DCOM RPC缓冲区漏洞利用蠕虫也会杀死其扫描到的Windows NT、Windows 2000、Windows XP以及Windows 2003系统上运行的`svchost.exe`进程。对于Windows 2000和Windows NT而言，这会导致系统不稳定甚至挂起；而对于Windows XP和Windows 2003，默认情况下会造成系统重起。

Blaster的爆发并不令人惊讶。在Blaster利用的漏洞被公布前的一个月，CERT/CC的主管Richard Pethia于2003年6月25日在主题为“网络安全、科学、研究和发展”的美国国土安全小组委员会的特委会会议[Pethia 03a]上明确表示：

当今理应关注互联网的安全问题。与互联网有关的漏洞将用户置于危险的境地。适应于组

⊖ SYN洪水（SYN flooding）是一种恶意客户程序用户所使用的对计算机服务器进行拒绝服务攻击（DoS）的方法。恶意程序利用伪造的IP地址重复地对服务器的每一个端口发送SYN（同步）包。