Mc Graw Hill Education

安全技术经典译丛

19 Deadly Sins Of Software Security
Programming Flaws and How to Fix Them

一个都不能有

——软件的19个致命安全漏洞

Michael Howard
(美) David LeBlanc 著
John Viega
肖枫涛 杨明军 译

清华大学出版社

一个都不能有

——软件的19个致命安全漏洞

Michael Howard
(美) David LeBlanc 著
John Viega
肖枫涛 杨明军 译

清华大学出版社
北 京

Michael Howard，David LeBlanc，John Viega

19 Deadly Sins of Software Security Programming Flaws and How to Fix Them

EISBN：0-07-226085-8

图书在版编目(CIP)数据

一个都不能有——软件的19个致命安全漏洞/(美)豪沃(Howard,M.)，(美)勒伯兰克(LeBlanc,D.)，(美)维加(Viega,J.)著；肖枫涛，杨明军译. —北京：清华大学出版社，2006.11

书名原文：19 Deadly Sins of Software Security Programming Flaws and How to Fix Them

ISBN 7-302-13804-4

Ⅰ. 软… Ⅱ. ①豪… ②勒… ③维… ④肖… ⑤杨… Ⅲ. 软件可靠性－研究 Ⅳ. TP311.5

中国版本图书馆 CIP 数据核字(2006)第 109367 号

出 版 者：清华大学出版社　　**地　　址**：北京清华大学学研大厦

http://www.tup.com.cn　　**邮　　编**：100084

社 总 机：010-62770175　　**客户服务**：010-62776969

组稿编辑：王　军　　**文稿编辑**：王　翔

封面设计：康　博　　**版式设计**：康　博

印 装 者：清华大学印刷厂

发 行 者：新华书店总店北京发行所

开　　本：185×260　**印张**：17　**字数**：372 千字

版　　次：2006 年 11 月第 1 版　2006 年 11 月第1 次印刷

书　　号：ISBN 7-302-13804-4/TP・8300

印　　数：1～4000

定　　价：38.00 元

译 者 序

随着计算机的普及，人们对于计算机的依赖性越来越高。而其中使计算机得以发挥作用的软件的安全性也越来越受到人们的重视，但是，随着目前软件规模的扩大并趋之复杂，软件出现的漏洞也越来越多，所造成的损失也就越来越大。同时人们意识到仅靠编程语言以及运行平台的安全性，并不能保证软件运行的安全性，关键的还是软件的编写者。

本书本着小巧、易读、实用的原则，涵盖了目前流行的编程语言和运行平台，覆盖了最为常见的与软件安全相关的19个致命漏洞。该书采用理论和实践相结合的方式，对于每个漏洞都给出了详细的描述、产生的原理、查找的方法、弥补的措施等内容，同时，对于每个漏洞，作者还精选了系统中实际出现的例子加以说明，使其更直观，令读者印象更为深刻。另外，在弥补措施中，作者结合多种不同的语言给出具体的代码弥补方案，从而更增强了实用性。

本书虽然小巧，但是非常实用，它的跨平台和跨语言的特性可以为几乎所有的相关从业人员提供良好的建议和实用工具。每一章相对独立而完整，对于时间宝贵的人来说，很有用处。同时，本书出自三位软件安全领域最为著名的专家之手，进一步增加了它的权威性。

在本书翻译过程中，我们本着信、达、雅的原则和认真负责的精神，在努力保持原著风格和充分理解原著思想的基础上，尽量用符合中文表达习惯的语言将本书奉献给各位读者。对于原书中存在的一些错误，我们也进行了修正。

本书由肖枫涛和杨明军翻译，肖国尊负责本书翻译质量和进度的控制与管理，全书最后由肖国尊负责统稿。欢迎各位读者对本书提供反馈意见，从读者的意见中来了解自己的不足，以便在今后译作中更多地、更切实地考虑读者的需要。

希望读者能从本书中受益。

译者

2006年3月

序　言

我们通常期望计算机可以按照我们的指示去执行任务，在现实生活中，我们通过使用软件来完成这些期望。目前计算机及其软件都变得非常的复杂，从我们点击鼠标到看到期望的结果这一过程中，可能经过了多层软件。为了充分利用计算机平台的能力，我们通常要依赖于这些软件层自身执行的正确性。

对于这些软件层来说，每一层都可能出现问题，软件运行的结果并不是作者所需要的，或者至少不是计算机操作者需要的。这些漏洞为我们的系统引入了一定的不确定因素，随之而来的是重大的安全漏洞。这些漏洞有些比较简单，比如软件或系统崩溃(可用于拒绝服务攻击)，或者缓冲区溢出(攻击者可以以此来替换应用程序的代码，从而执行任意的命令)。

只要在软件系统中出现了由于漏洞而导致的不确定性，一些如何保护系统的观点就只能成为空想。我们可以部署防火墙、使用内核级的技术来阻止缓冲区溢出，通常也可以使用权宜之计，但是通过这种方式并不能改变基础设施的安全状况。只有提高软件的质量，同时减少相关的漏洞，才可以在安全方面取得一定的成绩。

在当今的开发环境中，要彻底消除软件中存在的漏洞是不可能的，从安全的角度来说，软件开发方面实在有太多的安全隐患，即使从事这项工作的专业人员也并不能保证认识到所有的漏洞，更别提掌握这些漏洞了。

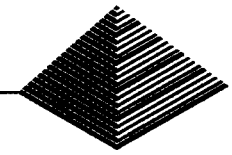
如果我们打算在消除安全漏洞方面有所进展，那么，当开发小组谈到实际的约束条件时，我们就需要帮助他们尽量简单地定位他们软件中存在的安全问题。关于软件安全有好几本很好的书，包括本书作者编写过的一些书，但是我认为在实际中应该消除所有的复杂性，只是为开发小组提供一个小的容易记忆的关键点集合，从而提高其软件的安全性。这意味着在提高软件安全性方面，我们要花费少量的精力来定位大多数的常见问题，而不是去提供一种完美但却不切实际的方案。

当我在国土安全部工作的时候，我请 John Viega 将 19 个编程“漏洞”总结成一个列表。最开始的列表仅仅帮助我们了解一些漏洞信息，也就是向大家暴露可能成为安全漏洞的一些信息，但是这个列表操作性不是很强。本书具有较强的操作性，它提供了一份对于开发人员来说至关重要的安全漏洞的简单列表，以便他们能够有意识地去注意这些漏洞，同时还提供了如何防止这些漏洞的方法。另外，本书还将告诉您如何去发现这些问题(通过代码审查或者软件测试)。这些技术和方法都极具针对性，同时，作者还提供

了一份“要”和“不要”的检查列表。本书作者做了大量的工作，希望能够用一种简单独立的方式来定位目前困扰我们的最常见的安全问题。我希望软件开发社团可以阅读本书，并利用本书中的方法来消除我们目前所使用的软件中存在的大量不确定性因素和安全威胁。

Amit Yoran
美国国土安全部国家网络安全局前局长
Great Falls, Virginia
2005年5月21日

作 者 简 介

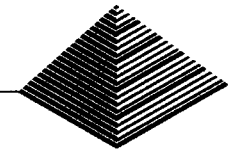


Michael Howard，Microsoft 公司安全工程组的高级安全项目管理经理，同时也是获奖书籍 *Writing Secure Code* 的作者之一。他还是 IEEE Security&Privacy Magazine 杂志“Basic Training”栏目作者之一、美国国家网络安全伙伴任务小组为国土安全部编写的“Processes to Produce Secure Software”一文的作者之一。作为微软“安全开发生命周期”的架构师，Michael 花费了大量的时间来制定和实施最佳安全实践，促进软件开发过程改进，为普通用户提供更为安全的软件。

David LeBlanc 博士目前是 Webroot Software 公司的首席软件架构师。在加入 Webroot 之前，他在微软公司的 Office 部门工作，是可信赖主动计算(Trustworthy computing Initiative)的创始人之一。在微软的网络安全组，他还充当“白客”的角色。David 也是 *Writing Secure Code and Assessing Network Security* 一书的作者之一。

John Viega，*19 deadly programming flaws* 一书的作者，这本书引起了出版社和媒体的极大关注。本书就是以该书为基础的。他是 Secure Software(www.securesoftware.com)的创立者和 CTO。他和其他人共同编写了关于软件安全的第一本书 *Building Secure Software*，同时他也是 *Network Security and Cryptography with OpenSSL* 和 *Secure Programming Cookbook* 的作者之一。他还是 CLASP 过程的第一作者，该过程将安全引入开发生命周期，他还负责了几个开源软件安全工具。John 是 Virginia Tech 的计算机科学副教授，同时也是 Cyberspace Policy Institute 的高级策略研究员。John 也是在软件安全和密码领域知名的研究者之一，并且一直为制定安全网络和软件安全相关的标准而努力。

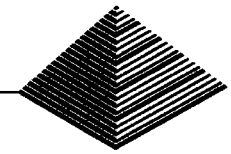
技术编辑简介



Alan Krassowski 是赛门铁克公司首席软件安全工程师。他领导了赛门铁克的产品安全小组，帮助赛门铁克产品团队发布安全技术，以减少风险以及和用户建立信任关系。在过去的 20 年里，Alan 参与了众多的商业软件项目的开发。在加入赛门铁克之前，他是一名开发主管、软件工程师，同时也是许多在行业中领先的公司(比如 Microsoft、IBM、Tektronix、Step Technologies、Screenplay System、Quark 以及 Continetal Insurance)的顾问。他获得了纽约罗彻斯特技术学院的计算机工程系学士学位。

David A. Wheeler 在提高高风险系统(比如大型系统或安全系统)的软件开发实践方面拥有多年的经验。他是 *Software Inspection:An Industry Best Practice* 一书的编者/作者之一，同时还是 *Ada 95:The lovelace Tutorial* 和 *Secure Programming for Linux and Unix HOWTO* 的作者，他也是 IBM developerWorks 的“Secure Programmer”系列栏目的作者。

目　录



第 1 章　缓冲区溢出……1
1.1　漏洞概述……1
1.2　受影响的编程语言……2
1.3　漏洞详细解释……2
1.3.1　受漏洞影响的 C/C++……5
1.3.2　相关漏洞……8
1.4　查找漏洞模式……8
1.5　在代码审查中查找该漏洞……8
1.6　发现该漏洞的测试技巧……9
1.7　漏洞示例……10
1.7.1　CVE-1999-0042……10
1.7.2　CVE-2000-0389—CVE-2000-0392……10
1.7.3　CVE-2002-0842、CVE-2003-0095、CAN-2003-0096……11
1.7.4　CAN-2003-0352……11
1.8　弥补措施……12
1.8.1　替换危险的字符串处理函数……12
1.8.2　审计分配操作……12
1.8.3　检查循环和数组访问……12
1.8.4　使用 C++字符串来替换 C 字符串缓冲区……12
1.8.5　使用 STL 容器替代静态数组……13
1.8.6　使用分析工具……13
1.9　其他防御措施……13
1.9.1　栈保护……14
1.9.2　不可执行的栈和堆……14
1.10　其他资源……14
1.11　本章总结……16

第 2 章　格式化字符串问题……17
2.1　漏洞概述……17
2.2　受影响的编程语言……18
2.3　漏洞详细解释……18
2.3.1　受漏洞影响的 C/C++……20
2.3.2　相关漏洞……20
2.4　查找漏洞模式……21
2.5　在代码审查中查找该漏洞……21
2.6　发现该漏洞的测试技巧……21
2.7　漏洞示例……22
2.7.1　CVE-2000-0573……22
2.7.2　CVE-2000-0844……22
2.8　弥补措施……23
2.9　其他防御措施……23
2.10　其他资源……23
2.11　本章总结……24

第 3 章　整数溢出……25
3.1　漏洞概述……25
3.2　受影响的编程语言……25
3.3　漏洞详细解释……26

3.3.1 受漏洞影响的C和C++ …… 26
3.3.2 受漏洞影响的C# …… 31
3.3.3 受漏洞影响的 Visual Basic 和 Visual Basic.net …… 32
3.3.4 受漏洞影响的Java …… 33
3.3.5 受漏洞影响的Perl …… 34
3.4 查找漏洞模式 …… 35
3.5 在代码审查中查找漏洞 …… 35
3.5.1 C/C++ …… 35
3.5.2 C# …… 37
3.5.3 Java …… 37
3.5.4 Visual Basic 和 Visual Basic.NET …… 38
3.5.5 Perl …… 38
3.6 发现该漏洞的测试技巧 …… 38
3.7 漏洞示例 …… 38
3.7.1 在 Windows 脚本引擎中存在的漏洞可以导致任意代码执行 …… 38
3.7.2 在 SOAPParameter 的对象构造函数中存在整数溢出 … 39
3.7.3 在 HTR 块编码中存在的堆溢出可以导致 Web 服务器遭到入侵 …… 39
3.8 弥补措施 …… 39
3.9 其他防御措施 …… 41
3.10 其他资源 …… 41
3.11 本章总结 …… 42

第4章 SQL注入 …… 43

4.1 漏洞概述 …… 43
4.2 受影响的编程语言 …… 44
4.3 漏洞详细解释 …… 44
4.3.1 受漏洞影响的C# …… 44
4.3.2 受漏洞影响的PHP …… 45
4.3.3 受漏洞影响的Perl/CGI …… 46
4.3.4 受漏洞影响的Java 和 JDBC …… 46
4.3.5 受漏洞影响的SQL …… 47
4.3.6 相关漏洞 …… 48
4.4 查找漏洞模式 …… 49
4.5 在代码审查中查找该漏洞 …… 49
4.6 发现该漏洞的测试技巧 …… 50
4.7 漏洞示例 …… 52
4.7.1 CAN-2004-348 …… 52
4.7.2 CAN-2002-0554 …… 52
4.8 弥补措施 …… 52
4.8.1 验证所有的输入 …… 52
4.8.2 不要使用字符串连接来构造 SQL 语句 …… 53
4.9 其他防御措施 …… 56
4.10 其他资源 …… 56
4.11 本章总结 …… 57

第5章 命令注入 …… 59

5.1 漏洞概述 …… 59
5.2 受影响的编程语言 …… 59
5.3 漏洞详细解释 …… 59
5.4 查找漏洞模式 …… 61
5.5 在代码审查中查找该漏洞 …… 61
5.6 发现该漏洞的测试技巧 …… 63
5.7 漏洞示例 …… 64
5.7.1 CAN-2001-1187 …… 64
5.7.2 CAN-2002-0652 …… 64
5.8 弥补措施 …… 65
5.8.1 数据验证 …… 65
5.8.2 当检查失败时 …… 67
5.9 其他防御措施 …… 67
5.10 其他资源 …… 68
5.11 本章总结 …… 68

第6章 未能处理错误信息 …… 69

6.1 漏洞概述 …… 69

6.2 受影响的编程语言……69
6.3 漏洞详细解释……69
6.3.1 产生太多的信息……70
6.3.2 忽略了错误信息……70
6.3.3 曲解了错误信息……71
6.3.4 使用了无用的错误值……71
6.3.5 处理了错误的异常……71
6.3.6 处理所有的异常……71
6.3.7 受漏洞影响的 C/C++……72
6.3.8 Windows 上受漏洞影响的 C/C++……72
6.3.9 受漏洞影响的 C++……73
6.3.10 受漏洞影响的 C#、VB.NET 及 Java……73
6.3.11 相关漏洞……74
6.4 查找漏洞模式……74
6.5 在代码审查中查找该漏洞……74
6.6 发现该漏洞的测试技巧……75
6.7 漏洞示例……75
6.8 弥补措施……76
6.8.1 C/C++弥补措施……76
6.8.2 C#、VB.NET 和 Java 弥补措施……76
6.9 其他资源……77
6.10 本章总结……77

第 7 章 跨站脚本……79

7.1 漏洞概述……79
7.2 受影响的编程语言……79
7.3 漏洞详细解释……79
7.3.1 受漏洞影响的 C/C++ ISAPI 应用程序或者过滤程序……80
7.3.2 受漏洞影响的 ASP……81
7.3.3 受漏洞影响的 ASP.NET 表单……81
7.3.4 受漏洞影响的 JSP……81
7.3.5 受漏洞影响的 PHP……81
7.3.6 受漏洞影响的使用 Perl 的 CGI 程序……82
7.3.7 受漏洞影响的 mod_perl……82
7.4 查找漏洞模式……82
7.5 在代码审查中查找该漏洞……83
7.6 发现该漏洞的测试技巧……84
7.7 漏洞示例……85
7.7.1 IBM Lotus Domino 跨站脚本和 HTML 注入漏洞……85
7.7.2 Oracle HTTP 服务器的“isqlplus”输入验证漏洞允许远程用户实施跨站脚本攻击……85
7.7.3 CVE-2002-0840……85
7.8 弥补措施……85
7.8.1 ISAPI C/C++弥补措施……86
7.8.2 ASP 弥补措施……86
7.8.3 ASP.NET 表单弥补措施……87
7.8.4 JSP 弥补措施……87
7.8.5 PHP 弥补措施……89
7.8.6 CGI 弥补措施……89
7.8.7 mod_perl 弥补措施……90
7.8.8 关于 HTML 编码的注意事项……90
7.9 其他防御措施……91
7.10 其他资源……91
7.11 本章总结……92

第 8 章 未能保护好网络流量……93

8.1 漏洞概述……93
8.2 受影响的编程语言……94
8.3 漏洞详细解释……94
8.4 查找漏洞模式……96
8.5 在代码审查中查找漏洞……96

8.6 发现该漏洞的测试技巧……99
8.7 漏洞示例……99
8.7.1 TCP/IP……100
8.7.2 电子邮件协议……100
8.7.3 电子商务……100
8.8 弥补措施……101
8.9 其他防御措施……104
8.10 其他资源……104
8.11 本章总结……104
第 9 章 使用 Magic URL 及隐藏表单字段……105
9.1 漏洞概述……105
9.2 受影响的编程语言……105
9.3 漏洞详细解释……105
9.3.1 Magic URL……106
9.3.2 隐藏表单字段……106
9.3.3 相关漏洞……106
9.4 查找漏洞模式……107
9.5 在代码审查中查找该漏洞……107
9.6 发现该漏洞的测试技巧……108
9.7 漏洞示例……109
9.7.1 CAN-2000-1001……109
9.7.2 MaxWebProtal 隐藏表单字段修改……109
9.8 弥补措施……109
9.8.1 攻击者浏览数据……110
9.8.2 攻击者重放数据……110
9.8.3 攻击者预测数据……112
9.8.4 攻击者更改数据……113
9.9 其他防御措施……114
9.10 其他资源……114
9.11 本章总结……114
第 10 章 未能正确使用 SSL 和 TLS……115
10.1 漏洞概述……115
10.2 受影响的编程语言……115
10.3 漏洞详细解释……116
10.4 查找漏洞模式……118
10.5 在代码审查查找该漏洞……119
10.6 发现该漏洞的测试技巧……120
10.7 漏洞示例……121
10.7.1 电子邮件客户端……121
10.7.2 Safari Web 浏览器……121
10.7.3 Stunnel SSL 代理……122
10.8 弥补措施……122
10.8.1 选择协议版本……123
10.8.2 选择加密套件……124
10.8.3 确保证书的有效性……124
10.8.4 验证主机名……126
10.8.5 检查证书撤销……126
10.9 其他防御措施……128
10.10 其他资源……129
10.11 本章总结……129
第 11 章 使用基于弱口令的系统……131
11.1 漏洞概述……131
11.2 受影响的编程语言……131
11.3 漏洞详细解释……131
11.4 查找漏洞模式……133
11.5 在代码审查中查找该漏洞……134
11.5.1 口令内容策略……134
11.5.2 口令修改和重置……134
11.5.3 口令协议……135
11.5.4 口令处理和存储……135
11.6 发现该漏洞的测试技巧……136
11.7 漏洞示例……136
11.7.1 CVE-2005-1505……136
11.7.2 CVE-2005-0432……137
11.7.3 TENEX 漏洞……137
11.7.4 Paris Hilton 劫持……138
11.8 弥补措施……138
11.8.1 多方式认证……138

11.8.2 存储以及检查口令…… 139
11.8.3 选择协议的原则…… 142
11.8.4 口令重置的原则…… 142
11.8.5 口令选择的原则…… 143
11.8.6 其他原则…… 144
11.9 其他防御措施…… 144
11.10 其他资源…… 145
11.11 本章总结…… 145

第 12 章 未能安全地存储和保护数据……147

12.1 漏洞概述…… 147
12.2 受影响的编程语言…… 147
12.3 漏洞详细解释…… 147
12.3.1 采用脆弱的访问控制机制“保护”秘密数据… 148
12.3.2 受漏洞影响的访问控制 149
12.3.3 代码中内嵌秘密数据… 151
12.3.4 相关漏洞…… 151
12.4 查找漏洞模式…… 151
12.5 在代码审查中查找漏洞…… 152
12.6 发现该漏洞的测试技巧…… 153
12.7 漏洞示例…… 155
12.7.1 CVE-2000-0100…… 155
12.7.2 CAN-2002-1590…… 155
12.7.3 CVE-1999-0886…… 155
12.7.4 CAN-2004-0311…… 156
12.7.5 CAN-2004-0391…… 156
12.8 弥补措施…… 156
12.8.1 使用操作系统的安全技术…… 157
12.8.2 C/C++ Windows 2000 以及更新版本的弥补措施 157
12.8.3 ASP.NET 1.1 以及更新版本的弥补措施…… 159
12.8.4 C# .NET Framework 2.0 弥补措施…… 159
12.8.5 C/C++ Mac OS X 10.2 以及更新版本的弥补措施…… 159
12.8.6 无需操作系统支持的弥补措施(不在代码中保存秘密数据)…… 160
12.8.7 Java 和 Java KeyStore 的注意事项…… 162
12.9 其他防御措施…… 163
12.10 其他资源…… 164
12.11 本章总结…… 165

第 13 章 信息泄漏……167

13.1 漏洞概述…… 167
13.2 受影响的编程语言…… 167
13.3 漏洞详细解释…… 168
13.3.1 旁路…… 168
13.3.2 TMI：太多的信息！ …169
13.3.3 信息流安全模型…… 171
13.3.4 受漏洞影响的 C#(以及其他的编程语言)…… 173
13.3.5 相关漏洞…… 173
13.4 查找漏洞模式…… 173
13.5 在代码审查中查找该漏洞… 173
13.6 发现该漏洞的测试技巧…… 175
13.7 漏洞示例…… 175
13.7.1 Dan Bernstein 的 AES 定时攻击…… 175
13.7.2 CAN-2005-1411…… 176
13.7.3 CAN-2005-1133…… 176
13.8 弥补措施…… 177
13.8.1 C#(以及其他编程语言)弥补措施…… 177
13.8.2 本地网络的弥补措施… 178
13.9 其他防御措施…… 178

13.10 其他资源 …… 178
13.11 本章总结 …… 179

第 14 章 不恰当的文件访问 …… 181
14.1 漏洞概述 …… 181
14.2 受影响的编程语言 …… 182
14.3 漏洞详细解释 …… 182
14.3.1 在 Windows 上受漏洞影响的 C/C++ …… 182
14.3.2 受漏洞影响的 C/C++ …… 183
14.3.3 受漏洞影响的 Perl …… 183
14.3.4 受漏洞影响的 Python …… 183
14.3.5 相关漏洞 …… 184
14.4 查找漏洞模式 …… 184
14.5 在代码审查中查找该漏洞 …… 184
14.6 发现该漏洞的测试技巧 …… 185
14.7 漏洞示例 …… 186
14.7.1 CAN-2005-0004 …… 186
14.7.2 CAN-2005-0799 …… 186
14.7.3 CAN-2004-0452 和 CAN-2004-0448 …… 186
14.7.4 CVE-2004-0115 Microsoft 的 Mac 版 Virtual PC …… 186
14.8 弥补措施 …… 187
14.8.1 Perl 弥补措施 …… 187
14.8.2 *nix 上 C/C++的弥补措施 …… 187
14.8.3 Windows 上 C/C++的弥补措施 …… 188
14.8.4 获取用户临时文件的目录 …… 188
14.8.5 .NET 代码弥补措施 …… 188
14.9 其他防御措施 …… 189
14.10 其他资源 …… 189
14.11 本章总结 …… 189

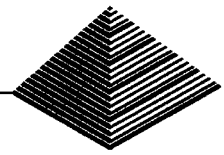
第 15 章 轻信网络域名解析 …… 191
15.1 漏洞概述 …… 191
15.2 受影响的编程语言 …… 191
15.3 漏洞详细解释 …… 192
15.3.1 受漏洞影响的应用程序 …… 194
15.3.2 相关漏洞 …… 194
15.4 查找漏洞模式 …… 195
15.5 在代码审查中查找该漏洞 …… 195
15.6 发现该漏洞的测试技巧 …… 195
15.7 漏洞示例 …… 196
15.7.1 CVE-2002-0676 …… 196
15.7.2 CVE-1999-0024 …… 196
15.8 弥补措施 …… 197
15.9 其他资源 …… 198
15.10 本章总结 …… 198

第 16 章 竞争条件 …… 199
16.1 漏洞概述 …… 199
16.2 受影响的编程语言 …… 199
16.3 漏洞详细解释 …… 199
16.3.1 受漏洞影响的代码 …… 201
16.3.2 相关漏洞 …… 201
16.4 查找漏洞模式 …… 202
16.5 在代码审查中查找该漏洞 …… 202
16.6 发现该漏洞的测试技巧 …… 203
16.7 漏洞示例 …… 204
16.7.1 CVE-2001-1349 …… 204
16.7.2 CAN-2003-1073 …… 204
16.7.3 CVE-2000-0849 …… 204
16.8 弥补措施 …… 205
16.9 其他防御措施 …… 206
16.10 其他资源 …… 206
16.11 本章总结 …… 207

第 17 章 未认证的密钥交换 …… 209
17.1 漏洞概述 …… 209
17.2 受影响的语言 …… 209
17.3 漏洞详细解释 …… 209
17.4 查找漏洞模式 …… 211

17.5 在代码审查中查找该漏洞··211
17.6 发现该漏洞的测试技巧·····212
17.7 漏洞示例·····212
17.7.1 Novell Netware MITM 攻击·····212
17.7.2 CAN-2004-0155·····212
17.8 弥补措施·····213
17.9 其他防御措施·····213
17.10 其他资源·····213
17.11 本章总结·····214
第 18 章 密码学强度随机数·····215
18.1 漏洞概述·····215
18.2 受影响的编程语言·····215
18.3 漏洞详细解释·····215
18.3.1 受漏洞影响的非密码学生成器·····216
18.3.2 受漏洞影响的密码学生成器·····216
18.3.3 受漏洞影响的真随机数生成器·····217
18.3.4 相关漏洞·····218
18.4 查找漏洞模式·····218
18.5 在代码审查中查找该漏洞··218
18.5.1 什么时候应该使用随机数·····218
18.5.2 查找使用 PRNG 的地方·····218
18.5.3 判断 CRNG 是否正确地播种·····219
18.6 发现该漏洞的测试技巧·····220
18.7 漏洞示例·····220
18.7.1 Netscape 浏览器·····220
18.7.2 OpenSSL 问题·····221
18.8 弥补措施·····221
18.8.1 Windows·····221
18.8.2 .NET 代码·····222
18.8.3 UNIX·····222
18.8.4 Java·····223
18.8.5 重放数字流·····223
18.9 其他防御措施·····224
18.10 其他资源·····224
18.11 本章总结·····224
第 19 章 不良可用性·····225
19.1 漏洞概述·····225
19.2 受影响的语言·····225
19.3 漏洞详细解释·····225
19.3.1 谁是您的用户？·····226
19.3.2 雷区：向您的用户呈现安全相关的信息·····227
19.3.3 相关漏洞·····227
19.4 查找漏洞模式·····227
19.5 在代码审查中寻找该漏洞··227
19.6 发现该漏洞的测试技巧·····228
19.7 漏洞示例·····228
19.7.1 SSL/TLS 证书认证·····228
19.7.2 Internet Explorer 4.0 根证书安装·····229
19.8 弥补措施·····230
19.8.1 简化 UI 以便用户参与··230
19.8.2 为用户做出安全决策···230
19.8.3 使有选择的松弛安全策略变得简单·····231
19.8.4 明确指出后果·····232
19.8.5 提供可操作性·····235
19.8.6 提供集中管理·····235
19.9 其他资源·····235
19.10 本章总结·····236
附录 A 19 个致命漏洞与 OWASP 的“前 10 名”漏洞的对应关系·····237
附录 B “要”与“不要”提示总结···239
译者术语表·····247

1



缓冲区溢出

1.1 漏洞概述

长久以来，人们都认为缓冲区溢出是低级语言中所存在的一个问题。这个问题的核心在于：为了照顾到程序的性能，人们通常并不具体区分用户数据指令和程序控制指令，而将它们混合在一起使用，再加上低级语言具有直接访问应用程序内存的能力，这样，低级语言就有可能因为未处理好用户数据而造成缓冲区溢出，并进而修改了程序控制指令，使得系统受到攻击。C 和 C++是受缓冲区溢出影响的两种最常见的编程语言。

严格意义上讲，当一个程序允许输入的数据大于已分配的缓冲区大小时，缓冲区溢出就会产生，但是，也有一些相关的漏洞产生同样的溢出效果，这其中最有趣的一个就是格式化字符串漏洞，我们会在第 2 章讲述。另一个具体的溢出漏洞是：攻击者可以在应用程序中的某个数组空间之外的任意内存地址写入数据。尽管，严格意义上讲这并不是一种典型的缓冲区溢出，但本章还会对该问题加以讲述。

缓冲区溢出造成的后果小到系统崩溃，大到攻击者获取相应应用程序的完全控制权。并且，如果运行该应用程序的用户具有较高的权限(root 权限、管理员权限或者本地系统权限)，攻击者还可以获得系统的完整控制权限，同时，系统中已经登录的用户、即将登录的用户都将处于攻击者的掌握之中。如果出现此漏洞的应用程序是一个网络服务程序，那么将会造成蠕虫的产生并蔓延。第一个著名的 Internet 蠕虫——Robert T. Morris(或者简称为 Morris)蠕虫——就是对 finger 服务器进行攻击的。在 1988 年，一次缓冲区溢出攻击差点使得 Internet 瘫痪，之后，虽然我们似乎已经明白了该如何去避免缓冲区溢出，但是在许多种类的软件中仍旧能常常见到有关缓冲区溢出的报告。

您可能会认为只有那些草率的、粗心的程序员才会编写出有缓冲区溢出漏洞的程序，但实际上，这个问题本身就比较复杂，而且一些解决方法也并不简单。事实上，任何一个编写过很多 C/C++代码的程序员几乎都出现过这个问题。我们——现在教其他的开发人员如何去编写更为安全的代码——在为用户编写软件时，也出现过 off-by-one 溢出的问题。即使是很棒很细心的程序员编写程序时也会出现缓冲区溢出的漏洞，但是最优秀的程序员知道如何有效地避免这种漏洞以及如何进行有效的测试来捕捉这种漏洞。

1.2 受影响的编程语言

C语言是最容易产生缓冲区溢出漏洞的语言，其次是C++语言。因为没有任何安全保护机制，所以采用汇编语言进行编程也很容易产生缓冲区溢出漏洞。尽管C++语言同C语言一样，天生就容易产生缓冲区溢出漏洞，但由于它是C语言的超集，如果在使用STL(Standard Template Library，标准模板库)时小心一些，则会大大地减少对于字符串的不安全操作。另外，日渐严格的C++编译器也会有助于程序员避免这样的漏洞产生。所以，我们建议：即使您是采用纯C语言编写代码，也要使用C++编译器进行编译，因为这样会产生更为整洁而且安全的代码。

最近出现的一些高级语言避免了程序员直接访问内存，当然，这有时候会造成性能上的较大损耗。一些高级语言，比如Java、C#以及Visual Basic，拥有自己的字符串类型、具有边界检查功能的数组，并且通常会避免直接进行内存访问。尽管有些人会说，这样的话那些缓冲区溢出漏洞就不会产生了，但是，更为准确的说法应该是：产生缓冲区溢出漏洞的可能性非常小了。实际上，大部分的高级语言都是由C/C++语言实现的，从而，实现中的缺陷可能会导致缓冲区溢出。高级语言代码产生缓冲区溢出漏洞的另一个原因是：这些代码最终都要和操作系统进行交互，而操作系统基本上都是由C/C++语言编写的。C#可以将直接进行内存访问的操作放入声明为unsafe的代码段中，然而，由于它能够方便地与用C/C++语言编写的操作系统和库进行交互，这样您也可能犯与C/C++语言中同样的错误。如果您主要使用高级语言进行编程，那么防范缓冲区溢出漏洞的主要方法是：验证传递给外部库参数的有效性。否则，这些参数可能会在这些库中产生缓冲区溢出。

虽然我们并不打算提供一份受影响语言的详细列表，但是，事实上，大多数的比较旧的编程语言都可能会产生缓冲区溢出漏洞。

1.3 漏洞详细解释

典型的缓冲区溢出漏洞是“smashing the stack”。对于一个已经编译好的程序来说，栈用来保存控制信息，比如一类特定的参数，这些参数指出程序执行完一个函数后应该返回的地址。由于x86处理器上的寄存器数量很少，常用的寄存器数据会暂时保存在栈中，然而，本地分配的变量也保存在栈中。这些变量有时并不能很准确地引用，因为它们是静态分配的，这一点和动态分配的堆内存正好相反。如果有人在讨论静态缓冲区溢出，我们应该明白的是，他实际上是指栈缓冲区溢出。这种漏洞的根源在于：如果应用程序写入的数据超出了栈上分配的数组的边界，那么攻击者则可以进一步指定控制信息。如果成功的话，那是非常危险的；攻击者很有可能借此将控制信息改为自己的命令信息。