<<C和C++安全编码>>

图书基本信息

书名：<<C和C++安全编码>>

13位ISBN编号：9787111261483

10位ISBN编号：7111261488

出版时间：2010-1

出版时间：机械工业出版社

作者：西科德

页数：227

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e。

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。

<<C和C++安全编码>>

内容概要

本书是关于C和C++安全编码的著作。
本书介绍了C和C++程序中已经导致危险的、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。
洞悉软件漏洞的成因，熟知规避之道　　通常而言，可利用的软件漏洞都由本可避免的软件缺陷所导致。
在分析了过去10年中近18000份漏洞报告后，CERT/CC发现少量的根本原因导致了这些漏洞的产生。
本书识别并解释了这些原因，而且展示了预防利用漏洞的步骤。
此外，本书还鼓励程序员采用最佳安全实践，并培养安全的开发理念，这不但有助于保护软件免遭当前的攻击，更可使它们免遭将来可能发生的攻击。
基于CERT/CC的报告和总结，Robert Seacord系统地揭示了最可能导致安全缺陷的编程错误，展示了这些缺陷的利用方式，介绍了可能导致的后果，并提供了安全的替代做法。
本书特别讨论了如下技术细节：　　改善任何C/C++应用程序的整体安全性。
抵御利用不安全的字符串操作逻辑的缓冲区溢出和栈粉碎攻击。
避免因对动态内存管理函数的不当使用而导致的漏洞和安全缺陷。
消除与整数相关的问题，包括整数溢出、符号错误以及截断错误等。
正确地使用格式化输出函数，避免引入格式字符串漏洞。
避免I/O漏洞，包括竞争条件等。
本书提供了许多针对Windows和Linux的安全代码、不安全代码以及利用程序的例子。
如果你负责创建安全的C或C++软件，或者需要保持这类软件的安全性，本书为你提供了详尽的专家级协助。
在这方面，其他任何书籍都望尘莫及。

作者简介

Robert C.Seaco rd是宾夕法尼亚州匹兹堡市SEI（SoftwareEngineering 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

<<C和C++安全编码>>

书籍目录

章节摘录

2．3．2缓冲区溢出　　当向为某特定数据结构分配的内存空间边界之外写入数据时，即会发生缓冲区溢出。
c和c++都容易发生缓冲区溢出问题，因为这两种语言具有以下共同之处：a）将字符串定义为以空字符结尾的字符数组；b）未进行隐式的边界检查；c）提供了未强制性边界检查的标准字符串函数调用。
　　取决于内存的位置以及溢出的规模，缓冲区溢出可能不会被侦测到，但可能会破坏数据，导致程序出现奇怪的行为甚至非正常中止。
　　缓冲区溢出是一个令人头痛的问题，因为在软件的开发和测试阶段并非总能发现该问题。
c和C++编译器在编译过程中并非总能识别可能引发缓冲区溢出的安全缺陷，在运行时也不会报告越界写问题。
不过一旦测试数据可以引发一个可侦测的溢出，我们就可以使用动态分析工具来发现缓冲区溢出问题。
　　并非所有的缓冲区溢出都会造成软件漏洞。
然而，如果攻击者能够操纵用户控制的输入来利用安全缺陷，那么缓冲区溢出就会导致漏洞了。
例如，有一些广为人知的技术可以用于覆写栈帧以执行任意的代码。
缓冲区溢出也可以在堆或静态内存区域被利用，做法是通过覆写邻接内存的数据结构。
　　在我们深入了解这些利用行为之前，理解进程内存是如何组织和管理的是非常必要的。
如果你对进程内存组织、执行栈以及堆管理等主题已经非常熟悉，可以直接跳到第2．5节。

媒体关注与评论

“信息系统的安全性并未随受攻击的数目和程度的增长同步得到改善。
为了改变这一现状，必须改进用于创建系统的策略和技术。
具体来说，必须从一开始就将安全性构建于系统之内而不是将其作为补救措施附加进来，这正是本书的要旨。
这本书向软件开发者详尽展示了如何构建高质量的系统，它们具有更少的漏洞，不易遭受代价高昂或后果严重的攻击。
任何开发者在进行重大的项目开发之前都应该阅读本书。
”　　——Frank Abagnale，作家、讲师、欺诈预防和安全文献领域资深顾问

<<C和C++安全编码>>

版权说明