微软公司核心技术书库

编写安全的代码

(美) Michael Howard
David LeBlanc 著

程永敬 吴 嵘 庄锦山 等译

机械工业出版社
China Machine Press

随着网络的普及，安全问题已经成了软件业的头等大事。但是，软件安全性的加强，仅仅依靠增加安全功能模块是远远不够的，必须从编码阶段就将安全理念牢记在心。本书是第一本指导程序员从内部加强软件安全的著作。由微软公司的两位安全权威撰写。本书从各种程序代码中可能存在的安全漏洞和安全问题入手，通过大量的实例和代码，详细地分析说明了编写安全的应用程序的方方面面，提供了一些目前尚无其他文档可查的用于安全的设计、安全的编码及测试技术的一些实用技术内幕。本书内容翔实，许多实例代码取材于微软实践过的第一手材料，实用性极强。配套光盘除包含大量实例、工具外，还提供了一份包含本书文字内容的完整的可检索的电子版(电子书)。

本书版权登记号：图字：01-2002-1499

图书在版编目（CIP）数据

编写安全的代码 /（美）霍华德（Howard, M.）等著；程永敬等译. －北京：机械工业出版社，2002.8

(微软公司核心技术书库)

书名原文：Writing Secure Code

ISBN 7-111-10285-1

Ⅰ.编… Ⅱ.①霍… ②程… Ⅲ.电子计算机－安全技术 Ⅳ.TP309

中国版本图书馆CIP数据核字（2002）第030843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）

技术编辑：周明涛

责任编辑：陈 敏 周 睿

北京牛山世兴印刷厂印刷 · 新华书店北京发行所发行

2002年8月第1版第1次印刷

787mm × 1092mm 1/16 · 22.5印张

印数：0 001-5 000册

定价：49.00元 (附光盘)

译 者 序

正如本书前言中所说，教你以安全的方式设计、编写和测试应用程序是本书惟一的目的。本书从各种系统和用户代码中可能存在的安全漏洞和安全问题入手，通过大量的实例和代码，详细地分析了编写安全的应用程序的方方面面。本书覆盖面较广，内容很新（包括了即将推出的Windows .NET Server和刚刚推出的Microsoft Visual Studio .NET中的一些最新的技术和特点），且有一定的深度，因为作者本身就是多年从事安全咨询和安全评审的资深专家。与许多其他安全主题图书不同，本书不仅仅告诉你什么是不安全的应用程序，也不仅仅是抱怨人们没有计划去构建安全的系统，而是完全注重实效地说明了系统是如何被攻击的，错误通常是怎样造成的，尤其重要的是，本书说明了如何构建安全的系统。无论是系统管理员、系统分析员、项目经理，还是程序设计人员，相信都可以从本书的讨论中获益匪浅。这确实是一本关于安全方面的易懂易用的好书！作为译者，我们很荣幸能将其译为中文，献给广大的读者！

参加本书翻译、录入、校对工作有：程永敬、吴嵘、庄锦山、张怀力、刘学来等。周明涛和刘江先生通审了全稿，有效地提高了书稿质量，对此深表感谢。此外，还要感谢所有给予我们帮助和支持的人们！

由于译者的水平有限，本书中可能还存在这样或那样的问题。敬请广大专家和读者指正。我们的Email是freediscovery@263.net。

2002年3月

序

在我们开发Windows 2000的时候，提高安全性是要点之一。那时，我们决定在该系统发布之前，做一个非同寻常的试验，来测试其应付挑战的能力。我们设立了一个名为“Windows 2000test.com”的Windows 2000 Web服务器，把它放在那里，等着看会发生什么。在两小时之内，数以百计的人已经在尝试对它进行攻击。几天之内，有上万人在苦心研究如何攻破Windows 2000系统。

今天，只要新产品一面市，黑客们就会开始花大力气找出并利用其安全漏洞。如果该产品的开发人员不投入同样大的精力在其代码中构建安全性的话，那么黑客们将十有八九会达到目的。产品的安全性和产品的功能同等重要。但不要误会我的意思，谁也不会去购买一个没有杰出功能的产品。但是，虽然开发人员明白如何实现产品的功能，却往往不懂得如何设计和构建安全性。本书将会改变这种现状。

本书提供了用于安全的设计、安全的编码以及测试技术的实用技术，其中许多内容尚无其他文档可查。本书将使你对如何构建安全的应用程序有更为充分的理解。Michael和David分别是微软的Windows安全倡导（Secure Windows Initiative）小组和高信度计算安全性（Trustworthy Computing Security）小组的成员。他们耳闻目睹了许多第一手材料：总是会有一些很基本的编码错误会对安全性造成破坏，而他们的项目为我们提高在产品中实现同Windows 2000和Windows XP一样的安全性提供了大量的帮助。本书将带给你全新的安全技术概念。

Brian Valentine

微软公司Windows部门副总经理

前　言

我们想写这样一本书已经很久了。我们一直致力于说服和教会人们如何加强其应用程序的安全性以避免受到攻击，然而直到最近关心系统安全性的人还不是太多。不要误解我们的意思：软件开发人员谁都想发布卓越的产品，而这里的“卓越”我们认为首先必须是安全的。

本书作者之一——Michael还记得1984年他首次在微软的Windows上编程的情形：那是一个简单的程序，与Kernighan和Ritchie的经典著作《The C Programming Language》（Prentice Hall PTR，1988年第2版）中经典的“Hello，World”程序没什么两样。当程序第一次成功编译、链接并运行的时候，他非常兴奋。我们敢肯定地说，任何在早期版本的Windows上工作过的人都会记得，那时创建一个Windows应用程序是多么的困难。Windows SDK和微软的C编译器的结合学起来并不那么容易，特别是如果你出身于像MS-DOS、PC-DOS或UNIX这些基于文本的背景时，困难就尤其明显了。

回过头来看一下1984年写的那个程序吧，它是否足够安全，可以抵御攻击呢？答案很简单，是的，它可以。它是安全的，首先是因为没有谁将基于Windows 1.x的计算机挂到任何种类的网络上，更不用说是Internet了。它是安全的，还因为那是在1984年，网络犯罪和基于Internet的肆意破坏行为还不是那么猖獗。

时代发生了多么大的变化！当今的Internet环境不可思议地充满着敌意，所有设计程序的人在设计的时候，都必须牢记在心。如果当年运行Windows 1.x的计算机挂在今天的Internet上，那么应用程序肯定会被攻击。这种应用程序在设计时可没有考虑到要运行在如此险象丛生的环境中。老实说，当时设计人员在头脑中想都没有想过安全性，因为Michael那时并不知道怎样进行编写安全的编码。现在，对于安全的代码，几乎没有人完全懂，同时也几乎没有人完全不懂。这里的“安全的代码（secure code）”并不是指安全性代码（security code）或者实现安全性功能的代码。我们指的是那种在设计上能够抵挡得住恶意攻击的代码。安全的代码同时也是健壮的代码（robust code）。

教你以安全的方式设计、编写和测试应用程序是本书惟一的目的。本书中，我们的目标就是进行不断的实践。同时也附带地让你明白代码是会受到攻击的。为了不说得过于生硬，所以让我们来重新解释一下。如果你创建了一个应用程序，这个应用程序运行在一台或多台计算机连接而成的网络上，或者由所有这些计算机及网络形成的大网——Internet上，那么你的代码将会受到攻击。

不安全的系统造成的后果很多，而且多种多样，包括产品的丢失，用户信任度的降低，以及经济上的损失。举例来说，如果有攻击者能够危及你的应用程序，比如使应用程序无法使用，你的客户就可能会流失。使用基于Internet的服务时，绝大多数人所能忍耐的等待时间上限都很短。如果这个服务无法使用，那么许多人会转向他方，而其他公司就会得到他们的光顾并大赚其钱。

对于无数软件开发公司来说，真正的问题在于，安全性看起来并不是开发过程中能产生收入的功能。正因为这样，管理层不想花钱培训开发人员如何编写安全的代码。管理层也会在安全技术上花钱，但这往往是应用程序被成功地攻破之后！而到那时，一切都太晚了，破坏已经造成。对被攻破的应用程序进行修正的代价，无论是从经济上，还是名誉上，都是高昂的。

保护财产免受偷窃和攻击已经是一条经过时间证明了的经验。我们的老祖先就已经制订了法律来惩罚那些偷盗、破坏或侵犯公民所有财产的人。简言之，人们认为某些动产和不动产属于私人，而应该抑制上述的那些行为。同样的道理也适用于数字世界，因此我们程序员的工作之一就是建立应用程序和解决方案来对数字资产进行保护。

本书中包含了一些基础的内容，这些内容应该已经在学校学习设计和构建安全的系统时学过。你可能会认为那种设计是架构设计师(architect)和项目经理的事，这是不错的，但作为开发人员和测试人员，你也应该了解要描述那些设计经得住攻击的系统应该都有哪些步骤。

我们两位作者都为了能写这本书而感到兴奋，这是因为本书是基于实际经验写成的，这些经验来自于我们长期对微软开发小组、外部的合作伙伴，以及需要开发安全的应用程序或者遭受过可怕后果的客户的培训。

谁应该读这本书

如果你正在设计应用程序，或者正在构建、测试解决方案或编写解决方案的文档，那么你需要本书。如果你的应用程序是基于Web的或者是基于Win32的，你需要这本书。如果你现在正在学习或者构建基于微软.NET框架的应用程序，那么你需要本书。简而言之，如果你的工作涉及构建应用程序，那么将会发现，从本书中可以学到很多东西。

本书的组织

本书分为五个部分。第1章和第2章组成第一部分“现代安全”，其中描述了系统为什么应该进行保护以防止攻击，还提纲挈领地介绍了设计此类系统的技术，并对其进行了分析。

本书的实质内容都在第二和第三部分。第二部分“安全的编码技术”包含了第3章到第8章的内容，对一些关键的代码编写技术进行了概括，这些技术可应用于几乎所有的应用程序。

第三部分“基于网络的应用程序”包含了四章内容（从第9章到第12章），重点集中在网络应用程序上，其中包括基于Web的应用程序。

第四部分“特殊话题”包括了三章内容（从第13章到第15章），探讨了一些其他地方讲得较少的话题，包括.NET应用程序的安全性及测试，以及安全的软件安装等。第16章讨论了一些不适合放在任何独立章节中的一些通用的指导原则。

附录部分包括了四个附录，涉及到其他各式各样的问题，包括一些危险的API，以及我们所听到的为不考虑安全性所找的一些站不住脚的借口。

第1、2、4～8以及12～14章由Michael编写。David编写了第3、9、11和15章。而第10章和第16章则由两位作者共同创作。

最后要说明的是，与许多其他安全主题图书不同，我们不仅仅告诉你什么是不安全的应

用程序，也不仅仅是抱怨人们没有计划去构建安全的系统。本书完全面向实践，说明了系统是如何被攻击的、错误通常是怎样造成的，尤其重要的是，本书说明了如何构建安全的系统。

配套光盘

本书所带的光盘中包含了书中讨论的所有示例程序，是本书的一份完整的可检索的电子版本，同时还提供了一些有用的工具。从光盘上的Readme.txt文件中，可以获取关于这些工具的信息。

要对光盘上的内容进行浏览，首先将光盘放到光驱中。如果启动程序没有自动执行，那么在光盘根目录下运行StartCD.exe即可。

安装示例文件

你可以在随书配套的光盘上直接浏览书中的示例文件，也可以将这些文件安装在你的硬盘上，并用来创建自己的应用程序。安装示例文件大约需要1.4MB的磁盘空间。如果运行这些文件时遇到问题，可以参考光盘根目录下的Readme.txt文件，或者参考书中描写这些程序的章节。

工具

在这张光盘上提供了两个工具：Token Master和PPCKey工具。光盘中还给出了关于PPCKey工具的支持文档。从光盘上Tools\PPCKey文件夹中的Readme.html文件中，可以获取关于这个工具的信息。

电子书

这张光盘中包含了本书的一份电子版本。你可以使用这份电子书来浏览本书的文字内容，也可对书中的内容进行检索。关于安装和使用电子书的信息，参见光盘中\eBook文件夹中的Readme.txt文件。

系统配置要求

本书中的绝大多数示例都是用C或者C++编写的，需要微软的Visual C++ 6，并安装了Service Pack 3或者更高版本；此外还有一小部分代码是用微软的Visual Studio .NET创建的，因为它们利用了较新的类特征。而Perl示例已经使用来自www.activestate.com的ActiveState Perl 5.6测试过。微软的VBScript和JScript代码则使用Windows 2000和Windows XP中带的Windows Scripting Host进行了测试。所有的SQL示例都使用SQL Server 2000测试过。最后要说明的是，Visual Basic .NET和Visual C#应用程序是使用Visual Studio .NET beta 2来编写和测试的。

除了两个应用程序以外，本书中所有的其他程序都能在装有Windows 2000的计算机上运行，这台机器还要满足我们推荐的操作系统的配置要求。第5章中的示例CreateSafeProcess和第12章中的示例UTF-8 MultiByteToWideChar需要Windows XP或者Windows .NET服务器才可以正确运

行。编译这些代码需要健壮一点的机器，同时要与推荐使用的编译器兼容。

声明

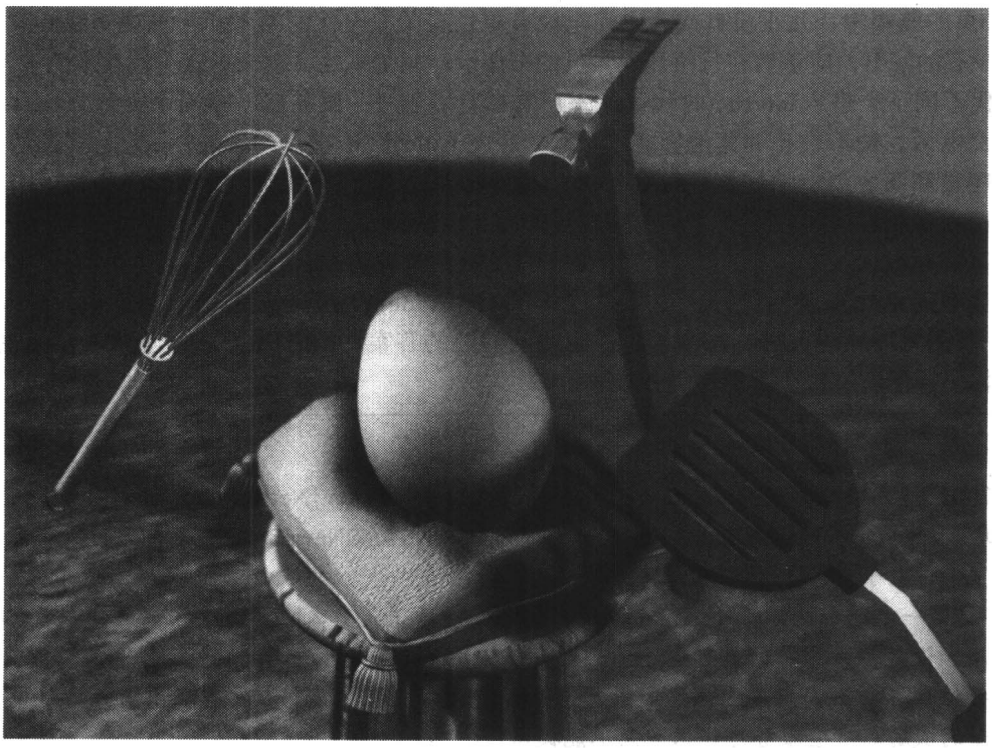
当你阅读本书的时候，书中提到的一些URL或许已经无效了。对于更新的URL，请访问www.microsoft.com/mspress/support/search.asp，在Search（搜索）框中输入0-7356-1588-8，然后点击Go（执行搜索）。

目　录

第四部分 特殊话题

附 录

第一部分　现代安全



第1章 对安全系统的需求

随着Internet重要性的不断增长，应用程序正呈现高度互连的趋势。在“美好的旧时光”，计算机在功能上通常是彼此独立的，如果说彼此间存在交互性的话，也只是微不足道的。在过去，如果你的程序不安全，那也无关紧要（最坏的情况也就是自己攻击自己而已）。而且，只要应用程序能成功执行其任务，绝大多数人是不会关心应用程序安全性的。在20世纪90年代出版的许多经典的“最佳实践（best practices）著作”，都是这种模式。例如，由Steve McConnell所著的优秀图书《Code Complete》（微软出版社1993年出版，中文版即将由机械工业出版社出版——编者注），在其850页的篇幅中，几乎没有或者说完全没有提供有关安全性的参考。不要误会我的意思：这是一本少有的好书，应该摆在每位开发人员的书架上。只是不要拿它来做安全性技术的参考。

时代不同了，进入Internet时代，几乎所有的计算机——服务器、桌面个人计算机，以及更新的蜂窝电话、袖珍设备和其他形式的设备，例如AutoPC和嵌入式系统——都是互连的。尽管这给程序员和商业创造了惊人的机会，但是也意味着这些互连的计算机可能会受到攻击。例如有些应用程序在设计时并不是为了在高度互连的（因而也是非常严酷的）环境中运行的，因此，计算机系统对攻击没有免疫能力。应用程序的开发者没有想到该程序会连在网络上，也没有考虑过它会被恶意攻击者所访问，你听说过“World Wide Web”（万维网）被称作“Wild Wild Web”（混乱无序的网）吗？在本章，你将找到问题的答案。Internet是一个充满了敌意的环境，所以，你必须使所有代码都能经受住攻击。

可这不是在喊“狼来了”

2001年7月13日，星期五，SANS（System Administration, Networking, and Security,）的Web站点www.sans.org被修改了。接下来的一周，SANS向其NewsBytes电子杂志的所有订阅者发送了一份电子邮件，其内容如下：

“这是一个令人吃惊的警示，它提醒我们Internet攻击是多么的具有破坏性。对每一个单独的程序和设置都必须重新进行检查，而在许多情况下，要将其重新进行设计以使其能够安全地进行操作，其出发点不应仅仅是应付今天所受到的攻击，而且还要着眼于未来两年中我们将遭受的威胁。有些服务可能很多天不能提供。”

Internet确实是一个充满了敌意的环境。你可以在www.msnbc.com/news/600122.asp上阅读更多有关遭受攻击的内容。

要点 永远都不要假设你的应用程序将只在很少一些指定环境中运行。如果你的应用程序有机会用在其他一些你还没有定义的设置环境中，这毕竟是一件好事。应当假设你编写的代码将运行在最具敌意的环境中，然后根据这个条件来设计、编写并测试你的代码。

还有重要的一点要记住，那就是安全的系统也是高质量的系统。将安全性作为首要特性来设计和构造的代码要比那些在事后才考虑安全性的代码健壮得多。安全的产品更能避免媒体的

批评，对用户更加具有吸引力，同时在修正和支持上所花的精力也最少。由于安全性是质量的保证，因而你不得不使用与人交往的技巧让开发组内的每一个人都去考虑安全问题。在本章中，我们将对所有这些问题进行讨论，并将给出一些方法，用来确保安全性在你的组织中处于最优先的地位。

如果你对代码的质量感兴趣，那么就请接着往下阅读。

1.1 “Wild Wild Web”上的应用程序

我曾经在Internet网上设立过一台计算机，仅仅为了观察在其身上会发生什么事情。只需几天，这台计算机就会被发现、被探测，并受到攻击。这样的计算机通常称为“蜜罐”（honeypot），是指的是为了吸引黑客而设置的计算机，通过它，你可以观察黑客的活动。

参考信息 要想对有关蜜罐的信息，以及黑客如何攻破系统进行更多的了解的话，请查看Honeynet工程的站点project.honeynet.org。

我还曾看到过黑客攻击的全过程，那是在1999年，我在www.windows2000test.com站点工作的时候。这个站点现在已经关闭了，但在那时它用来在Windows 2000正式发布之前对该系统进行实战测试。我们在星期五悄悄地将这个站点的Web服务器临时放到Internet上，而在星期一的时候该服务器就处于密集的攻击之下了，而我们从没有向任何人讲过它的存在。

Windows 2000的测试站点是怎样被发现的

临时放在Internet上的计算机不会被人发现吗？再想想看，Windows 2000测试站点几乎是立即就被人发现了，而这里要讲的就是其发生的过程。（顺便说一句，如果在这段短文中有你不熟悉的概念，请不要担心。这些概念都将在本章的内容中进行解释。）有人对微软公司所属的外部IP地址进行扫描，找到了一个新的活动IP地址；显然，微软公司设立了一台新的计算机。然后，这个人对各种端口进行探测，看看哪个端口是打开的，这种行为通常称作“端口扫描”。有一个端口打开了，是端口80，这是HTTP（超文本传输协议）服务器的端口。因此这个人就发送一个HTTP包头请求，来查看这个服务器是什么；它是一个IIS 5（Internet Information Service 5）的服务器。当然，IIS 5在那时还没有发布呢。接着这个人就启动了一个Web浏览器，并输入该服务器的IP地址，发现是一台由微软Windows 2000测试小组主办的一个测试站点（域名为www.windows 2000 test.com），除此之外，什么信息都没有。最后这个人在www.slashdot.org上粘贴了一则通告，而在短短的几小时内，这台服务器就被不断探测，并被IP层的攻击所淹没了。

想想看，我们只是将一台服务器挂在了网上！

我们的观点得到了验证：攻击发生了。目前，在这场仍在继续的战斗中，攻击者具有更加高级的手段，这使情况变得更糟糕了。有些攻击者具有非常高明的手段，又非常的聪明。他们具有深厚的计算机知识，同时也有充足的时间。他们有时间也有精力来探测和分析计算机的应用程序，来找出其安全漏洞。坦率地讲，对于其中的某些攻击者，我怀有深深的敬意，尤其是对于那些“白帽子”（white-hat）（指无恶意的人——译者注），又称好伙计（good guy），我本人

就认识很多这样的人。其中最好的白帽子会与软件供应商紧密地合作，这些软件供应商中也包括微软公司，他们的工作就是发现和补救安全问题，然后由软件供应商向用户发布一份安全公告，提示用户采取相应的补救措施。例如应用软件补丁或者更改设置。这种方式有助于使Internet社区摆脱毫无防范能力的处境，而如果这些安全缺陷首先被那些蓄意破坏的人发现的话，他们就将展开广泛的攻击。

许多攻击者只不过是傻乎乎的破坏者，他们被称为“脚本小子”（script kiddyes）。脚本小子几乎没有什么关于安全的知识，他们只是利用一些由知识更渊博的攻击者所写的脚本来攻击那些不安全的系统，而这些知识较为渊博的人找出系统安全的漏洞，编写文档，并自己编写的代码来挖掘这些安全漏洞。挖掘（exploit，也常称为sploit）是指攻破系统的方式[⊖]。

这正是事情变得愈加棘手的原因。试想你发布了一个应用程序后，有一个攻击者发现了一个安全漏洞，而在你有机会纠正这个问题之前，这个攻击者就将其发现和攻击手段公开了。这时脚本小子就来了，他们将攻击Internet上所有运行这个应用程序的机器。有很多次我都处于这样的情况之下，那真是一件可怕的事情，一点儿也不好玩。那时人们都为制作补丁忙得团团转，整天都处于一种混乱嘈杂的气氛中。你最好在第一步就不要将自己陷入这种境地，这就意味着，应设计安全的应用程序来有意识地抵御攻击。

刚刚得出的观点有些主观。我们已经从软件开发人员的角度考察了构造安全系统的原因，如果不能安全地构建系统，不仅将需要做更多的工作，同时将有损公司名声，这些反过来又将随着用户转而选择竞争者中预先认识到这个问题且具有较好安全支持的产品，而使得你的应用程序销量降低。现在让我们来看一下实际的情况：从最终用户的观点来看。

你的最终用户要求应用程序可以像所做的广告宣传那样工作，而每次当他们启动这些应用程序的时候，都会这样期望，而被攻击的应用程序可实现不了这种期望。你的应用程序操作、存储并希望能够对用户和公司的涉密数据进行保护。你的用户可不希望自己信用卡信息被贴在Internet上，不会希望其医疗数据被“黑”，或者系统感染病毒。前两个例子为我们提出了用户的私密性问题，而后面的那个则对宕机时间和数据的丢失方面提出了要求。而这正是你的工作：创建应用程序帮助你的用户从计算机系统中最大限度地得到好处，而无需担心数据丢失和私密信息被侵入。如果你不相信我，可以去问问你的用户。

1.2 让每个人都参与进来

我们已经看到，发布安全软件的需求已经大大高于从前了，因而安全第一就成为共同的格言。用户要求我们构建安全的应用程序，他们认为这是当然的权利，而不是什么特殊要求。而竞争者的销售队伍则会私下告诉潜在的用户，说我们的代码有风险，而且是不安全的。那么，该从哪里着手向整个单位灌输安全性的观念呢？最好的时机就是从开始做起，而这会是一项困难的工作。这之所以困难，是因为需要对公司产生明显的影响，而安全通常被认为是一种“拦路虎”，花了钱而只有很少或没有经济上的回报。向管理层灌输构建安全产品的理念需要机智老练，有时还需要搞一些破坏活动，让我们逐一道来。

⊖ 校注：exploit是黑客常用的行话，用做名词时指安全漏洞，用做动词时指对漏洞的利用。