Building Secure Software

How to Avoid Security Problems the Right Way

构建安全的软件

避免产生软件安全问题的正确方法

[美] John Viega
Gary McGraw 著
钟向群 王鹏 译

由 Bruce Schneier 作序

Addison Wesley

ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

Pearson Education

清华大学出版社

构建安全的软件

避免产生软件安全问题的正确方法

Building Secure Software：How to Avoid Security Problems the Right Way

[美] John Viega 著
Gary McGraw
钟向群 王 鹏 译

清 华 大 学 出 版 社
北京

内 容 提 要

本书是目前国内第一本讲述如何构建安全的软件的教材。书中以全新视角直面软件安全性问题，其前半部分讲述了软件安全的理论和风险管理的思想；焦点是如何将安全性融入软件工程的实践中；重点是在软件开发周期中采用一系列的安全风险规避的原则、方法和技术。后半部分深入到实现的细节中，以编写安全程序所需的基本技巧来武装开发人员，向开发人员阐述如何识别和避免软件开发中形形色色的安全陷阱，以及跳出漏洞的补丁的怪圈。

本书的读者对象包括软件开发过程中的所有参与者，从管理人员到系统设计人员、编码程序员；对于计算机安全专业的师生以及专业的安全人员都是必读之书。

EISB 0-201-72152-X

Building Secure Software: How to Avoid Security Problems the Right Way

北京市版权局著作权合同登记号 图字：01-2003-1107

图书在版编目（CIP）数据

构建安全的软件：避免产生软件安全问题的正确方法/（美）维加等著；钟向群，王鹏译. —北京：清华大学出版社，2003
ISBN 7-302-06515-2

Ⅰ. 构… Ⅱ. ①维…②钟…③王… Ⅲ. 软件可靠性—研究 Ⅳ. TP311.5

中国版本图书馆 CIP 数据核字（2003）第 024761 号

出版者：清华大学出版社（北京清华大学学研大厦，邮编 100084）
http://www.tup.com.cn
印刷者：北京市耀华印刷有限公司
发行者：新华书店总店北京发行所
开 本：异 16 印张：22.625 字数：492 千字
版 次：2003 年 4 月第 1 版 2003 年 4 月第 1 次印刷
印 数：0001～5000
书 号：ISBN 7-302-06515-2/TP・4886
定 价：47.00 元

对本书的赞誉

“John和Gary对计算机的安全性提供了一个全新的视角。一开始就正确地去做，日后就无须再去修修补补，在当今这个“蹩脚软件（shovelware）”的时代，这无疑是一个很超前又很重要的概念。当主流软件供应商都在为产品发布的β测试版而伤脑筋时，本书不失为醒脑之音，值得一读！”

——Marcus J.Ranum，NFR安全公司的CTO，“Web Security Sourcebook”的作者

“我们碰到的安全性问题往往来自于软件的错误。系统越复杂，找到这些问题越困难，花费也越高。当我们期望构筑安全可靠的交易并从物理的身份识别转向数字的身份识别时，遵循本书提出的一系列原则就显得越来越重要了。本书写得非常好，对于关心软件开发安全性的人们来说，是必备之书。”

——Terry stanley，Master Card公司芯片安全部副总裁

“当入侵者肆虐时，许多人想着关门闭户，而Viega先生和McGraw先生却在计算机安全的源头开始研究如何将安全性植入起始系统。简单地说，就是如何处理基本的安全性优先问题”

——Charlie Babcock，Interactive Week （交互周刊）

“Viega和McGraw终于完成了技术界盼望已久的作品。这是两个安全领域专家对构建安全的软件的全新视角。基于风险管理的的方法来管理安全性问题是贯穿全书的中心思想。对于在代码中如何避免缓冲区溢出，以及如何理解元素的完整性和交互性，本书均提出了全面详实的指南。作者阐释了理解和管理风险对任何系统项目的重要性。这本书是从事系统设计、系统构建和系统管理的专业人员必读之书。”

——Aviel D.Rubin，博士，AT&T实验室首席研究员，“White-Hat Security Arsenal”和“Web Security Sourcebook”的作者

“防于未然，胜争朝夕”

——Michael Howard，Microsoft Windows XP小组的安全主管

“John Viega和Gary McGraw向设计和实现软件以及关注安全性的专业人员奉献了一本非常有用的手册。本书不仅解释了编写安全的软件所涉及的一系列概念和原则，也包含了具体如何构筑防范入侵者破坏的软件的方法，例如缓冲区溢出的深入解释、大多数软件灾难的深入分析等等。而且，本书也列出了许多有用工具的出处（免费软件等），从而更

增加了其实用性。这是互联网时代软件人员的必读之书。”

——Jeremy Epstein，webMethods公司产品安全及性能部总监

“安全性其实很简单，只须运行优秀的、完美的软件即可。而完美往往是不可能的，人们必须找到一种切实可行的替代办法，否则就得面对那些安全脆弱性的嘲讽。Viega和McGraw提供了软件开发通向完美的宏大构思和实践要旨。”

——Crispin Cowan，博士，Oregon研究院的副研究员，WireX的合伙人和首席科学家

“大部分人都在和安全性问题的各种现象打交道，很少有人去探究大多数安全性问题背后的真正原因：设计和开发周期中的问题。在许多大学里，学生们学到的是不安全的编码风格，许多人把开发单用户系统软件的概念带到了网络的、彼此关联的环境中，这是非常危险的。这些状况正在迅速地损害国家的关键基础设施以及大多数的商业组织，并将国民置于危险之中。因此，当务之急是大部分软件设计者必须打破他们的坏习惯，重新学习。因此，我衷心地希望本书能引起对此领域的关注。总之，这个领域需要“疗伤”，用户不会总是“正确地”使用系统，恶意的攻击者更是如此。因此，编写安全的代码来对抗这个危机四伏的环境是真正的解决办法。”

——Mudge，@stake公司R&D的执行副总裁（EVP）和首席科学家

序

如果软件本身的安全性没有搞得如此糟糕的话，我们就无需在网络安全性方面花费这么多的时间、资金和努力了。想想你最近读到的那些安全漏洞吧，要么是极具杀伤力的数据包，让攻击者通过发送特殊包而捣毁某些服务；要么就是某种缓冲区溢出，使攻击者通过发送特定的畸形消息包控制计算机；要么就是某个加密漏洞，让攻击者读取加密信息或者愚弄认证系统。而如此种种，其实都是软件问题。

有时候，网络安全可以防范这些漏洞；某些防火墙也可以阻挡特定类型的数据包或消息，或者只允许来自某个可信源的连接（但愿攻击者没有在防火墙里面，也没有被某个可信资源所雇佣）；入侵检测系统也可在检测到特定漏洞时报警；可管理的安全监控服务可以侦查到正在作案的线索并立即中断这种入侵。但在所有这些情形之下，其错误根源仍在软件之中。正是糟糕的软件导致了这些脆弱性。

蹩脚软件非常普遍，超乎你的想像。通常，一个大型应用软件在销售时，仍带着数以百计甚至数以千计的和安全相关的漏洞。其中一些随着人们对该软件的使用逐步被发现了，于是厂商们就对这些已知的漏洞打上补丁，并希望用户安装上这些厂商提供的补丁软件，或者通过配置网络安全设备来防范这些漏洞。而软件中其他未被发现的弱点则继续潜伏，甚至永远不被发现，但它们的确存在，仍有被人发现和利用的可能。

坏软件是一种过失。

导致坏软件的原因在于软件开发系统。安全性并不是开发过程结束时一锤子定音的，它必须在一开始就正确设计。遗憾的是，那些负责产品安全性的人往往并不负责软件的开发；呼吁增强安全性的人总是争不过那些要求增强功能性的人；拥护和支持软件安全性设计原则的人也不主管软件的发布日程。软件公司往往追求“更多，更快”，而不是“少一些，慢一些，更安全一些”。

要创建安全性软件，开发者须要懂得如何安全地设计软件。这包含几个方面的内容。首先我们需要良好的教育和培训，程序员必须学会如何将安全性植入其软件设计之中，如何安全地编写代码；其次，需要更好的计算机语言和开发工具——这些工具在开发过程中能抓到那些较普通的安全脆弱点，使之更容易修复。更重要的是，我们需要一种意识：风险意识、问题意识以及修正意识，这不是说一夜之间就能大功告成的，而是说必须要从始至终达到此种功力。

本书是理解软件安全性的一个非常关键的工具。本书作者Viega和McGraw在安全的软件设计理论和实践方面均做了非常杰出的工作。本书非常实用，容易理解，且非常全面。它并不能在一夜之间奇迹般地将你变为软件安全的专家，但它会使你对软件的安全性更为敏感。而对问题越是敏感，就越接近问题的解决。

我们指望读者（程序员、软件设计师、软件项目管理者）能找出问题的解决之道。软件产业本身并不能解决软件的安全性问题，因为缺少市场刺激，这也是软件缺乏可靠性的原因。市场没有动力的原因则是软件厂商在其产品不安全时并不承担任何风险。软件产业已经证明，不管软件有怎样的漏洞或弱点，它们仍占有市场份额。如果汽车制造商也不管产品可靠性的话，我们就可能买到一部一氧化二氮喷嘴（nitrous oxide injector）和油管线（fuel line）连接在一起的汽车；厂商也能使加速器快到刹车不能控制的程度；厂商也可以不计后果地制造尽可能多的性能和功能。但是我们没有看到这种情况，因为汽车制造商们将不安全的产品投入市场会面临法律诉讼。

而软件产业没有类似的法律，看看你购买过的那些软件的皱皱巴巴的许可证（甚至你也帮忙写过），软件卖得堂而皇之，没有任何保证。甚至没有保证软件的功能会像其广告中所讲的一样，就更不用说安全性或者可靠性了。它也不能保证安装时不会弄垮整个网络。事实上，软件厂商显然对产品的可靠性有足够的豁免权。

多年以来，我都在呼吁，计算机安全就像其他所有安全性问题一样，需要用风险管理的方式来考虑。我们并不能避免某些技术或过程的应用所带来的威胁。我们需要管理这些风险。在这个定义上，电子空间和真实世界并无两样。人类四千多年以来一直在试图解决社会安全性问题，但仍然没有技术可以避免盗窃、绑架、谋杀的威胁。我们最好的办法就是管理风险。电子空间为什么就可以不同呢？

本书就是采用风险管理的方法来解决安全性问题。我近来主要是关注检测与响应，本书则着重于防范，更重要的是，它防于源头：软件的设计阶段。本书的开篇就恰当地设置了目标和期望，前面几章讨论了以风险管理的方式对待软件安全性的重要性，并且在安全性和软件最终目标之间引入了技术平衡思想，并使之根植于心。第5章和第6章则深入安全性设计的核心，提供了构建安全软件的10大指导原则，并阐述了软件稽核的思想。随后的内容就比较技术化了，主要是软件安全性的技术细节，范围也很广，从恶性的缓冲区溢出，到开发者碰到的对付防火墙的问题等等。

软件安全还有很长的路要走。我们不仅需要学会如何去做，更需要懂得这样做的重要性。您拥有这本书本身就表明在正确的方向上迈出了第一步。读它，学习它，并用之于实践。

希望寄托在你们身上。

Bruce Schneier
CIS公司（Counterpane Internet Security）的创立者和CTO
http://www.counterpane.com

前　言

“一本书就是一部思想的机器”

——摘自I.A.Richards的“Principles of Literary Criticism”

本书旨在帮助那些涉足软件开发的人们学会构建安全的软件所必需的原则，尽管其中包含了一些很底层、很细节的知识，也许对程序员是最适用的，但本书的读者对象包含软件开发过程中的所有人，从管理者到程序员。本书后半部分展示了一些特别的代码示例和技术细节，而前半部分则是大的原则和通用的知识，营造了一个安全软件的环境，包括安全目标、安全技术以及软件风险管理的概念。

涉及计算机安全的技术书籍很多，但迄今为止，尚无人对开发程序本身的安全性这个主题付出很大的努力。如果你想学习如何设置防火墙，对单个主机加锁或是构建一个虚拟专用网络，都会找到许多资源，而本书的内容自在其外；因为大多数有关安全的书籍是为了解决人们在网络安全性上的压力和焦虑，因此往往关注的是如何提高保密程度，如何保护网络资源，尽管这个网络世界中软件总是一再地被攻破。

遗憾的是，许多安全人员也习惯性地认为软件中存在安全性问题是正常的、普遍的，并且泰然接受。一些人甚至认为开发者构建安全的软件太难了，因此不必提出这个问题。相反，他们将努力集中于“最佳实践”（best-practice）的网络安全解决方案上，部署防火墙，实施入侵检测，并不时地修补已知的安全性问题。

对于解决软件的安全性问题，我们是乐观的。事实是，编写没有任何安全漏洞的程序是很困难的，但我们也确信，编写一个“足够安全”（secure-enough）的程序比编写完美的无任何错误的程序要容易得多。难道因为不能将那些bugs彻底清除掉，就根本放弃对软件bugs的扫荡吗？当然不能！同样，人们不应该在了解这个问题之前，就开始在软件安全上玩起掷骰子的戏法。

学习任何一点知识都会受益非浅。许多产品有那么多的安全性问题的最大原因在于，许多从事开发的技术人员没有学习过如何编制安全的代码。其中一个原因就是迄今为止没有什么地方可以提供有益的信息。本书旨在缩小这种教育和培训的鸿沟，以编写安全程序所需要的基本技巧来武装软件开发人员。

也就是说，你不能期望本书能帮助你消除软件中的所有安全性问题。声称本书是软件安全性问题的金钥匙是不妥当的，这样会忽视真实世界中软件安全性问题的极端困难性。我们不能忽视现实，应该正视和接受它，而且运用风险管理的思想来分析和对待软件安全性问题。

在真实世界中，软件几乎不可能绝对安全，首先是没有100%安全的事物。大多数软件存在可以突破的风险和漏洞，问题仅仅在于突破系统所需要花费的资金和付出的努力。即

使软件没有任何bug，服务器也被防火墙守护着，盯准你的人也会通过内部来攻击你，抑或采用“黑包”（black bag）的突破手法。安全性是复杂的，是一个系统工程问题，因此，我们不仅提供了安全软件设计的一些基本原则，同时也关注那些最常见的风险，以及减轻这些风险的方法。

本书的组织结构

本书从内容上分为两个部分。前一部分侧重于代码编制之前所应思考和了解的软件安全性知识。焦点就是如何将安全性集成于软件工程的实践中。重点是在开发生命周期的早期就引入降低和减少安全性风险的方法和原则。在初期就在系统中设计好安全性是一种相对容易的方法，在投资和花费上也远比事后修补要便宜得多。我们除了关注需求和设计外，也特别强调了系统的安全性分析，这也是非常关键的一项技巧。本书的前一部分对涉足软件开发的任何层次的人都是有所裨益的，无论是商业层的领导者，还是具体开发者，相信都会有兴趣。

后一部分深入到实现的细节中，即使是一个相当坚固的架构，在开发时也会有许多和安全性相关的薄弱的死角。我们以血淋淋的细节向开发者阐述如何识别和避免一些常见的编码级问题，比如缓冲区溢出、竞争状态等等陷阱。本书后一部分对那些在编程第一线的人将会有所帮助。

我们尽量提供通用的材料，也就是说，除非某个特殊的安全性问题，否则我们都会尽量避免那些依赖具体操作系统或程序设计语言的东西。比如，我们不讨论POSIX“可移植性”问题，因为它们没有普遍性。不过，我们还是花了一整章来讨论缓冲区溢出，因为这是一个异常重要的问题，尽管缓冲区溢出的问题主要是与C或C++相关。

由于我们关注的是那些能应用在很广泛领域和层次上的技术，因此有许多很有价值的技术本书没有介绍，包括Kerberos、PAM（可插拔认证模块）以及移动代码（mobile code）沙箱技术（sandboxing）等等。这些技术都值得专门介绍（并不是所有这些技术都有恰当的介绍）。本书的配套站点http://www.buildingsecuresoftware.com/，提供了这些安全性技术的相关资源链接。

代 码 实 例

虽然本书提供的材料大多数和具体程序语言无关，但我们的示例大多是用C语言编写的，主要是因为它应用得较广。另外，用C语言来表达事件的准确性和完整性用其他语言

更难一些。将示例移植到其他程序设计语言很容易，主要是找到相应的调用或数据结构即可。不过，我们偶尔也用到了用Python、Java以及Perl编写的代码示例，主要是在一些与C语言差别较大的场合。本书所有代码都可在http://www.buildingsecureso- ftware.com/上得到。

本书尽管坚持操作系统独立性原则，但仍有偏重UNIX的倾向性。我们也承认在某些专题上，特别是Windows系统下，仍有一些问题不尽人意。Windows NT是基本遵循POSIX的，但实际上，Windows程序员往往不愿意使用POSIX应用编程接口（API）。比如，我们听说大多数Windows程序员不使用标准C字串库，而喜欢用双字节编码（Unicode）的字串处理例程。编写本书时，我们仍不知道Windows API中哪些常见函数是易于引发缓冲区溢出的，因而不能提供一个全面的列表。如果以后有人能给出此列表，我们将非常乐意将它贴到本书的Web站点中。

本书提供的代码全部在运行Red Hat 6.2的机器上测试过。大多数程序也在OpenBSD机器上测试过。不过，我们提供的代码仍是基于“想当然”的基础，尽管我们努力使Wed站点上贴出的代码尽可能具有可移植性，但必须提醒的是，由于可用的资源有限，而且也没有足够的时间，因此大家在各自特定系统上编译这些代码遇到的问题，恕不能一一帮助解决，但我们真诚地欢迎读者提供的任何意见及软件补丁。

联 系 方 式

欢迎大家给我们发电子邮件，任何评论、错误修订或建议都会让我们欣喜和感激。我们的网址是：http://www.buildingsecuresoftware.com

目　录