TURING 图灵程序设计丛书 网络安全系列

WILEY

The Shellcoder's Handbook

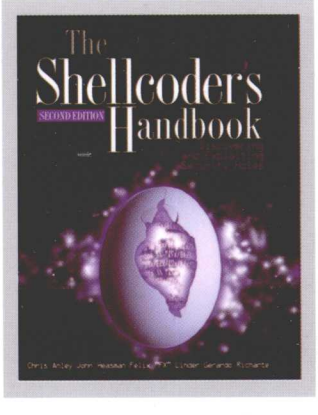
Discovering and Exploiting Security Holes Second Edition

黑客攻防技术宝典

系统实战篇（第2版）

[英] Chris Anley
[英] John Heasman
[德] Felix "FX" Linder
[美] Gerardo Richarte
等著

罗爱国 郑艳杰 译



- 跟世界顶级安全技术大师学习黑客攻防技术
- 全面分析系统安全漏洞
- 大量实例和代码片段

人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书 网络安全系列

The Shellcoder's Handbook

Discovering and Exploiting Security Holes Second Edition

黑客攻防技术宝典

系统实战篇（第2版）

[英] Chris Anley
[英] John Heasman
[德] Felix "FX" Linder
[美] Gerardo Richarte
等著

罗爱国 郑艳杰 译

人民邮电出版社
北 京

图书在版编目（CIP）数据

黑客攻防技术宝典：第2版. 系统实战篇 /（英）安雷（Anley, C.）等著；罗爱国，郑艳杰译. —北京：人民邮电出版社，2010.1
（图灵程序设计丛书）
书名原文：The Shellcoder's Handbook: Discovering and Exploiting Security Holes, Second Edition
ISBN 978-7-115-21796-7

Ⅰ. ①黑… Ⅱ. ①安…②罗…③郑… Ⅲ. ①计算机网络－安全技术 Ⅳ. ①TP393.08

中国版本图书馆CIP数据核字（2009）第216728号

内 容 提 要

本书由世界顶级安全专家亲自执笔，详细阐述了系统安全、应用程序安全、软件破解、加密解密等安全领域的核心问题，并用大量的实例说明如何检查 Windows、Linux、Solaris 等流行操作系统中的安全漏洞和 Oracle 等数据库中的安全隐患。

本书适用于所有计算机安全领域的技术人员和管理人员以及对计算机安全感兴趣的爱好者。

图灵程序设计丛书

黑客攻防技术宝典：系统实战篇（第2版）

◆ 著 [英] Chris Anley [英] John Heasman 等
[德] Felix "FX" Linder [美] Gerardo Richarte
译 罗爱国 郑艳杰
◆ 责任编辑 朱 巍
人民邮电出版社出版发行 北京市崇文区夕照寺街14号
邮编 100061 电子函件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
◆ 三河市潮河印业有限公司印刷
开本：800×1000 1/16
印张：35
字数：826千字 2010年1月第1版
印数：1－3 000册 2010年1月河北第1次印刷
著作权合同登记号 图字：01-2008-3322号
ISBN 978-7-115-21796-7

定价：79.00元

读者服务热线：(010)51095186 印装质量热线：(010)67129223
反盗版热线：(010)67171154

版 权 声 明

译 者 序

自着手翻译本书的第1版起，已近4年。4年中，有太多的变化，但这本书一直都在我的心里，是它促使我做一件以前从未想过的事情——翻译，是它让我结识了更多的朋友，是它让我明白了自己的无知。在知识更新如此频繁的时代，本书能否成为经典还有待时间的检验，但如果你对黑客技术感兴趣，它无疑会成为你的得力助手。

本书是众多作者精诚合作的成果，其中不乏创新之处。然纵观国内安全类书籍，创新者寥寥。一是国内与国外的研发环境的确有相当大的差距，这是实情；二是国内某些人即使有所突破，有所创新，也不愿意与他人分享。黑客精神强调的是自由，是创新。从这一方面说，国内很少有人称得上是真正的黑客。更有甚者借黑客之名，却不行黑客之实，将黑客技术用于谋取一己私利上。我在这里警告这些人，请正当使用黑客技术，切勿继续滥用，否则终究会引火上身。

本书比较系统地介绍了shellcoder应该掌握的知识，从最基本的栈、堆、内存布局等知识点，扩展到操作系统的层次，并花了大量的篇幅介绍怎样发现并利用漏洞，这是本书的核心，也是重点所在。更难能可贵的是，整本书的内容来源于实际，而高于实际，从更高的理论层次指导我们怎样学习shellcoding。

翻译本书的过程有快乐也有痛苦。快乐自不必说，痛苦有三：一是本书由众多高手合著而成，有些行文通俗易懂，但也有一些比较晦涩，不免让我绞尽脑汁；二是本人技术水平有限，书中的一些技术难点就如同硬骨头，让我难以下手；三是平常空闲时间有限，4年中我把大部分业余时间都花在这上面了，可谓旷日持久。但令人欣慰的是，在翻译本书的过程中，我得到了家人与朋友的理解与支持，最终使这本书得以大功告成。

在翻译本书第1版时，我感叹无人慧眼识珠。但第2版一面世，图灵公司的刘江总编就找到我联系翻译事宜，非常感谢他给我这个机会。

另外还要感谢看雪论坛上的kaien、icytear、kmyc、qos等朋友给出的修改意见。特别感谢SCZ、kanxue给予的大力支持。

致　谢

“冰冻三尺，非一日之功”，黑客攻防与学习也一样，不是照本宣科，而是一种生活方式。谨以此书献给理解此道的每一个人。

首先感谢本书的所有合著者：Gerardo Richarte、Felix “FX” Linder、John Heasman、Jack Koziol、David Litchfield、Dave Aitel、Sinan Eren、Neel Mehta和Riley Hassell。特别感谢Wiley出版公司的工作人员：杰出的责任编辑Carol Long和优秀的策划编辑Kevin Kent。从个人角度，我还要感谢NGS小组，感谢他们提供的大量研究结果、技术方面的讨论和思想。最后，我还要感谢我的夫人Victoria，感谢她对我的爱，感谢她一直以来对我的支持与包容。

——Chris Anley

感谢我的朋友和家人，感谢他们一如既往的支持。

——John Heasman

感谢Phenoelit的朋友们。尽管工作、生活中有很多波折，尽管我有许多千奇百怪的想法，他们仍旧和我在一起。要特别感谢Mumpi，他是我的好朋友，在各种各样的活动中都给予我可贵的支持。另外，还要感谢SABRE Labs团队，感谢Halvar Flake，他在团队的工作中起着关键的作用。感谢我的爱人Bine，感谢她日复一日对我的关爱。

——Felix “FX” Linder

感谢社区中的每一个人，大家共享快乐、观点和发现。尤其感谢在Core安全技术公司工作过的那些杰出的人士，还有攻击技术写写团队的老朋友们，和他们在一起永远会有简洁但极具启发性的灵感闪现。还要感谢Chris和John（合著者）以及Wiley出版公司的Kevin Kent，他们花了很多时间审读我那蹩脚的英文，使行文更流畅。同时，还要感谢我的夫人Chinchin，感谢她一直陪伴在我身边，倾听我的喜怒哀乐，并永远给予我支持。

——Gerardo Richarte

前　言

“如果术语的含义总是变来变去，不仅会导致原本不相干的推理发生混淆，就连既定的前提和结论也会频繁地被颠倒。”

——摘自艾达·奥古斯塔[①]在*Sketch of The Analytical Engine*一书（1842年）中所做的注释

本书第1版主要介绍了怎样发现和利用安全漏洞，第2版仍然紧紧围绕这一主题展开。如果你是一位技术娴熟的网络审计员、软件开发人员或系统管理员，如果你想弄明白如何发现和利用最底层的bug，这本书将是你最好的选择。

那么本书究竟会讲些什么呢？前面的内容或多或少概括了一些。本书主要关注任意的代码执行漏洞，攻击者借这些漏洞在目标机器上运行他们的代码。这种情况发生时，通常程序会把数据解释成自身的一部分，例如，攻击者利用此类漏洞可以把HTTP的Host首部变成返回地址，把Email地址的一部分变成函数指针，等等。程序在执行这些数据之后，结果通常都是灾难性的。现代处理器、操作系统以及编译器的架构是此类问题产生的根源，正如艾达所言，“操作的记号通常也是操作结果的记号”。当然，她讨论的是数学中的难点：数字5可能意味着“5次方”，也可能意味着“第5个元素”，但基本的思想是一致的。如果你混淆了代码和数据，就会陷入困境。因此，本书就来讨论代码和数据，以及混淆两者可能会带来的后果。

自本书第1版于2004年面世以来，安全领域变得更加复杂了，世界也发生了很大的变化。一方面，编译器和操作系统普遍内置了防护措施，用于防范本书重点讨论的各种漏洞——当然，这些措施远远还谈不上尽善尽美。另一方面，尚无迹象表明任意代码执行漏洞的“供应”在不久的将来会难以为继，更何况查找这些漏洞的方法仍在不断花样翻新。如果你访问美国国家漏洞数据库网站（nvd.nist.gov），单击statistics，选择buffer overflow，就会发现缓冲区溢出的数量逐年增加。

很显然，我们仍需要了解这些bug以及它们是如何被利用的。实际上，当我们考虑怎样保护自己时，有许多局部防御措施可供选择，在这种情形下，有人主张很有必要了解精确的机制。如果你正在审计网络，在你的评估过程中，做出一次成功的破解将会带给你100%的信心；如果你是一个软件开发者，生成用于概念验证的利用程序将有助于理解首先应该修复哪些bug；如果你正准备购买一款安全产品，知道怎样规避不可执行栈、利用棘手的堆溢出或者编写利用程序编码器，都将有助于你更好地判断各厂商的产品质量。通常来说，有知胜于无知。那些居心不良的人

① 英国数学家，世界上第一个计算机程序员。她是英国著名诗人拜伦的女儿。——编者注

已经知道这些东西了，所以网络审计者、软件作者、公共网络的管理者也应该了解它。

本书和其他同类的书有什么不一样呢？首先，本书的作者都长期奋战在第一线，发现并利用bug是我们工作的一部分。我们不仅是这么写的，而且每天也是这么做的。其次，你会发现我们没有在工具上花费过多的笔墨。这本书的大部分内容全是关于安全bug的第一手材料——汇编程序、源码、栈、堆等。掌握了这些内容你就能够编写工具，而不仅仅是使用别人编写的工具。最后，还有一个观点和态度的问题。尽管书中没有专门论述，但我们相信你在阅读全书的过程中随时都能够体会到：你必须动手实践，不断探索，最终彻底理解自己使用的系统。这样你才能真正体会到学习的乐趣。

无需多言，你已经一册在握了[①]。希望你能喜欢它，也希望它能帮上你的忙，更希望你不要把这些知识用错地方。如果你有什么想说的，不管是批评还是建议，都请告诉我。

Chris Anley

① 本书配套网站http://www.wiley.com/go/shellcodershandbook提供了本书中的所有示例代码和相关信息。——编者注

目　录

Part 1 第一部分

破解入门：x86上的Linux

第一部分概述怎样发现和利用漏洞。我们专为本书精心准备了各种示例代码，并列举了一些真实的漏洞，帮助读者学习破解技术。

本部分将介绍在Intel 32位（IA32或x86）Linux上利用漏洞的细节。在Linux/IA32平台上发现和利用漏洞是非常容易的，也很好理解，这正是我们选择从Linux/IA32开始学习的原因所在。对黑客来说，Linux最容易理解，因为在准备破解时就已经熟知操作系统的内部结构了。

在透彻理解了这一部分的内容并完成实验后，本书后面几部分将会逐步介绍更复杂的发现和利用漏洞的情况。第2章介绍栈溢出，第3章介绍怎样编写shellcode，第4章介绍格式化串漏洞，第5章介绍Linux平台上的堆溢出。学完这些内容之后，你就可以去理解更复杂的漏洞挖掘、利用技术了。

本部分内容

第1章 写在前面

为了使你更好地理解本书其他部分的内容，本章将介绍一些基本的概念，这些内容和大学课程中所要求的阅读材料类似，希望你们早就了然于心。本章不会面面俱到地介绍所有你需要知道的内容，你应该以本章作为学习后续章节的起点。

通读本章，温故而知新。在阅读过程中，如果不太理解某个概念，建议你记下来，以进行更深入的研讨。在学习后面的内容前，要花些时间来理解这些概念。

本书配套网站（http://www.wiley.com/go/shellcodershandbook）是对本书的有益补充，在这个站点上可以找到本书大部分例子的源码。在运行这些示例时，可以将这些示例代码复制和粘贴到你喜欢的文本编辑器中，以节省时间。

1.1 基本概念

要想真正掌握本书的内容，你至少应该熟悉计算机编程语言、操作系统和硬件体系结构等内容。如果你不能理解它们的工作机理，也就难于检测出它是否发生了故障。这个法则适用于计算机，同样适用于发现和利用计算机漏洞。

除此之外，你还有必要熟悉安全研究者常用的行话，即安全研究者常用的一些定义、术语等，以便更好地理解本书后面的内容。

漏洞（vulnerability，名词）：系统中存在的安全缺陷，攻击者常利用它们，以不同于程序设计者的意图来操纵系统，包括影响系统的可用性、提升访问特权、在未经授权的情况下完全控制系统，以及一些其他危害。漏洞通常也称为安全漏洞或安全错误。

利用[①]（Exploit，动词）：利用漏洞，试图以不同于程序设计者的意图操纵目标系统的行为。

破解（exploit，名词）：利用漏洞的工具、指令集或代码，也称为Proof Of Concept（POC）。

0day[②]（名词）：指还没有向公众揭露的漏洞的破解代码，有时也指漏洞本身。

模糊测试工具（fuzzer，名词）：它是一种工具或应用程序，主要功能是尝试着把所有可能的（或大量的）畸形数据提交给目标系统，以此来检测目标系统中是否存在错误。它能使攻击者在不完全了解目标系统的内部功能时也可能会发现错误，并在适当的时候利用它们。

① 本文中有时也译成破解。——译者注

② 0day另外一层含义是指骇客在最短时间内（不一定是当天）发布软件的破解版本。——译者注

1.1.1 内存管理

内存管理，特别是Intel 32位（IA32）体系结构的内存管理知识是学习本书所必须掌握的内容之一。本书的第一部分主要介绍IA32 Linux的内存管理知识。因为本书描述的大多数安全漏洞都源自“改写”或“溢出”内存，你还需要理解操作系统是怎样管理内存的。

指令与数据

现代计算机不会真正区分指令与数据，所以当我们把数据作为指令提交给处理器时，它也会很高兴地执行这些“指令”，正因如此，破解目标系统才成为可能。在后续章节里，我们将介绍当系统设计者要求输入数据时怎样插入指令，也将介绍怎样利用溢出用自己的指令改写程序的指令。当然，做这些的目的只有一个：控制目标程序的执行流程。

当执行程序时，程序体有序地排列在内存里。首先，操作系统在内存中为程序运行创建地址空间，地址空间包含实际的程序指令和需要的数据。

操作系统在创建地址空间后，把程序的可执行文件加载到新创建的地址空间里。程序（可执行文件）一般包含三种类型的段：.text、.bss和.data。.text段在内存中被映射为只读，.data和.bss被映射为可写。全局变量一般保存在.bss和.data段里。.data段包含静态初始化的数据，.bss段包含未初始化的数据，.text段包含程序指令。

加载完成后，系统紧接着就开始为程序初始化“栈”和“堆”。栈是一种“后进先出”（Last In First Out，LIFO）的数据结构，即最后入栈的数据，将第一个从栈上移走。栈比较适合保存暂时性的信息，即不需要长期保存的信息。栈用于保存局部变量、函数调用信息以及其他调用函数（过程）后系统通常会清除的信息。

栈的另外一个重要特征是它的地址空间“向下减少”，也就是说，栈上保存的数据越多，栈地址的值就越小。

堆是另外一种保存程序信息的数据结构，更准确的说法是，它保存程序的动态变量。堆是“先进先出”（First In First Out，FIFO）的数据结构，允许在“堆”的一端插入数据，从另一端移走数据。堆的地址空间是“向上增加”的，即堆上保存的数据越多，堆地址的值就越大，这一点和栈正好相反。如下面的内存空间图所示。

```
↑更低地址 (0x08000000)
Shared libraries
.text
.bss
Heap (grows ↓)
Stack (grows ↑)
env pointer
Argc
↓更高地址 (0xbfffffff)
```

内存管理是本书的基础，读者必须透彻、详尽地理解这一概念。建议你先抽时间阅读本书第13章前半部分介绍的内存管理知识，也可以访问http://linux-mm.org/了解Linux内存管理知识。牢固掌握内存管理的知识，将有助于你更好地掌握使用内存的编程语言——汇编语言。