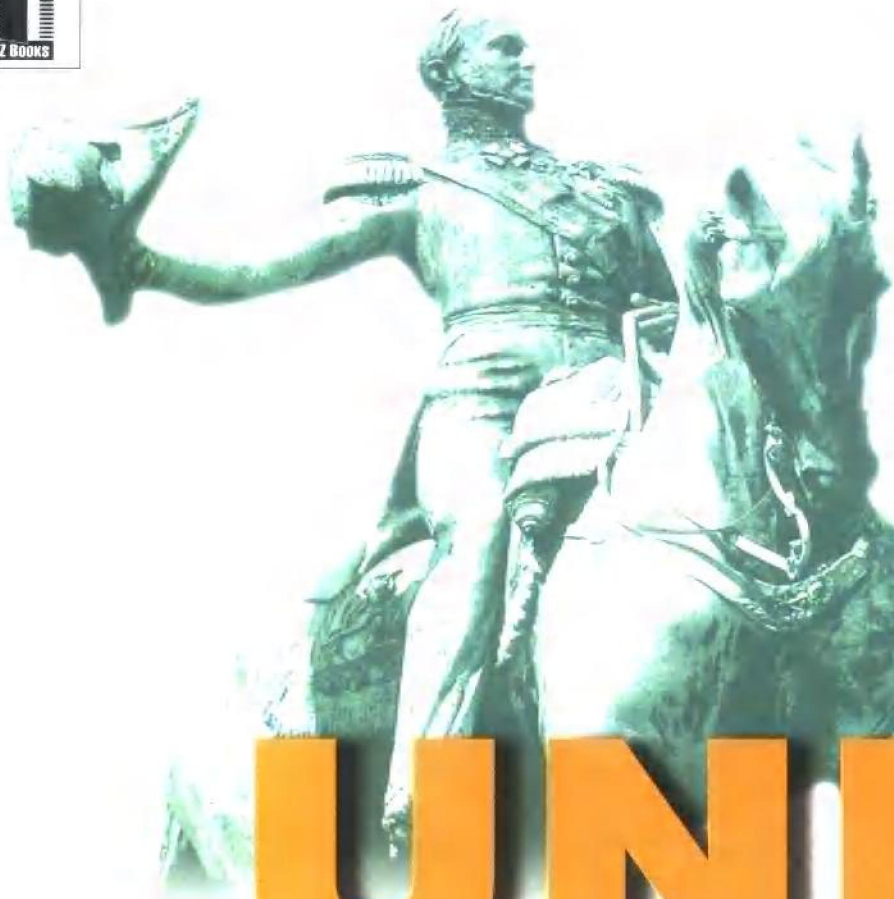
HZ BOOKS

UNIX实用工具译丛



UNIX 系统安全工具

UNIX System Security Tools

(美) Seth T. Ross 著／前导工作室 译

机械工业出版社
China Machine Press

McGraw-Hill

UNIX 实用工具译丛

UNIX 系统安全工具

（美）Seth T. Ross 著
前导工作室 译

机械工业出版社
China Machine Press

本书详细介绍了UNIX系统安全的问题、解决方法和策略。其内容包括：帐号安全及相关工具Crack；日志系统的机制和安全性，日志安全工具Swatch；如何测试系统的弱点，系统弱点的测试工具COPS和Tiger；网络安全的概念；提高网络安全性的防火墙等。本书语言简洁，层次清晰，是UNIX系统管理员的必备参考书。

本书版权登记号：图字：01-1999-3673

图书在版编目（CIP）数据

UNIX系统安全工具/（美）罗斯（Ross, S.T.)著;前导工作室译. -北京：机械工业出版社，2000.4

(UNIX实用工具译丛)

书名原文：UNIX System Security Tools

ISBN 7-111-07819-5

I.U… II. ①罗…② 前… III. UNIX 操作系统－安全技术 IV. TP316.81

中国版本图书馆CIP数据核字（2000)第13286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)

责任编辑：赵红燕

北京市密云县印刷厂印刷 · 新华书店北京发行所发行

2000年4月第1版第1次印刷

787mm × 1092mm 1/16 · 14.75印张

印数：0 001-6 000册

定价：25.00元

译者的话

近些年来，随着Internet的广泛流行，UNIX操作系统的发展非常迅速。全球的站点大多数建立在UNIX平台上，这给UNIX带来机遇，也带来了挑战。当网络上的黑客越来越多时，怎样建立一个基于UNIX的站点才能保证它的安全性，而且又不影响系统的实际应用呢？本书试图就这些问题给出可能的解决方案。

本书详细描述了UNIX系统安全的各方面内容。首先介绍了UNIX系统安全的基本概念和相关知识，使读者对UNIX系统安全有一个全面的了解。然后对具体的帐号安全、文件系统安全、日志安全和网络安全等分别进行了详细的描述，提出了可能出现的安全问题，介绍了各种安全策略，并给出了实际应用中的一些经典工具。本书详细介绍了这些工具的使用和配置，使读者能够尽快上手。这些工具基本上都是自由软件，读者可以在Internet上很容易地找到它们。为了方便读者，作者对每种工具都给出了可以找到的地址。

本书后面有三个附录供读者参考：附录A“Internet资源”提供了许多出色的有关UNIX系统安全的站点；附录B“公开端口号”提供了一个完整的端口编号和描述；附录C“GNU通用许可证”为接受和发布各种自由软件提供了依据。书后的“词汇表”解释UNIX安全中常见的术语。

本书由董威组织翻译，前导工作室的全体人员共同参加了本书的翻译、录入、校对和排版工作。由于时间仓促，水平有限，书中难免有不足和错误之处，敬请广大读者提出宝贵意见，我们将十分感谢！

1999年12月

前　言

UNIX操作系统是我们当今文明的一个重要部分。上千万人每天使用由UNIX机器提供的服务，但通常没意识到这点。当UNIX已在商业和学术界使用了几十年后，它的重要性又因为Internet的兴起而得到更大的加强。UNIX和Internet虽然是逻辑上不同的实体，但在现实世界中是密不可分的。大多数重要的Internet服务首先是在UNIX上发展起来的，尽管出现了像Windows NT这样的交互式操作系统，但UNIX仍然是由Internet引发的通信革命的主体。

有意思的是：UNIX从没刻意去成为一个安全的计算机平台。从30年前的AT&T贝尔实验室开始，UNIX就被设计成一个多用户、多任务的操作系统，并具有易于编程和性能安全的特征。UNIX成为世界范围内信息环境的重要发展部分并不奇怪，它已经遭受过好奇者和恶意者的残酷攻击。没有人知道有多少UNIX和类似UNIX的系统正在使用中，更不用说一年内会遭到多少次攻击，可以检测到的有上万次，每天就有数百次。

本书的目的是给UNIX系统管理员提供工具和技术，给一个UNIX系统提供安全基准。UNIX安全性协会已经为过去10年中出现的安全性工具进行了分类。使用像COAST这样的安全标准，本书评价了30多种能找到的工具并选择了几种来覆盖到各层次，一些代表某类特定工具（例如Crack是一种典型的口令检查工具），一些展现出整体的优越性（例如Tripwire较好地达到了“安全工具评述”中建议的标准），其他作为综述出现。

本书并不是一个有关UNIX系统安全的普及版，本书中许多材料是针对管理某个部门的服务器或家庭单机、起安全管理员作用，并拥有一定技术的个人。大多数的UNIX安装（不论大或小）具有共同的策略目标：连续操作、用户帐号和数据不应冲突、重要的系统事件应该被监测并记录、系统可用且安全。

没有一个工具或一组工具能使一个操作系统完全安全，也没有任何书能提供各种方式来使一个系统完全避免广泛的恶意攻击。如果读者使用了这里的工具和技术，就能得到基本的安全，在多数环境中，这些工具被认为是可行且足够了。当然，“多数环境”不包括秘密军事设施、核电站、飞行控制系统、银行系统等。本书没有危害这类重要系统安全的内容，这些系统的安全需要提供更深入的方法。

在使用本书提到的工具时，读者需要查看额外的附文。尽管本文指“UNIX”，但并不只是这样。UNIX由AT&T开发并向公众开放了很大一部分。现在有十几种UNIX，由不同的开发者、大学、机构和个人开发（见表1-1）。作者试图确保命令、工具和技术能在通用的UNIX产品上运行，但不能保证本书中所有东西都能在某种特定UNIX系统上像所描述的那样工作。本书中多数例子的参照平台是Red Hat Linux 5.0。作者还使用一个系统以运行HP-UX 10.20和不同的BSD系统。这些系统在一个测试网络中配置并在本书完成时拆除。

UNIX世界是动态的。甚至在写本书时，三种参考平台都已被更新的版本代替（Red Hat Linux 6.0、HP-UX 11等）。计算机安全工具的世界也是动态的，很可能这里描述的一些工具已被新版本代替。本书描述的多数工具和方法提供了URL作为Internet资源列表。强烈建议读者在使用这些工具的时候查看在线信息。如果读者有一些关于这些工具或UNIX安全性的问题，

建议把新闻组comp.security.unix作为一个帮助来源。

读者还能找到一些额外的有关本书主题的信息，如错别字、未来版本信息，可以链接到下面的站点：

http://www.albion.com/usst/

词汇提示：如计算机爱好者和工作者多年来指出的，“hacker（黑客）”（喜欢探索系统工作细节的问题解决者）不是“cracker（破坏者）”（为了兴趣和利益侵入系统的带恶意的人）。许多主流报告者、大众文化和罪犯自己都已忽略了这个区别，罪犯喜欢叫自己“hacker”而不是非法侵入者、贼、间谍、强盗、破坏者等等。建议保留术语“hacker”来描述那些创造了从World Wide Web到Linux等一切东西的真正聪明的工作者。本书把那些未经认证闯入系统的人叫作“cracker”、“attacker（袭击者）”或“intruder（侵入者）”。

作　者

第1章 概 述

本章从UNIX的一个经验定义开始，简要地介绍了UNIX的历史和不同的UNIX版本，提供了计算机安全和相关概念的分类，介绍了UNIX安全工具和评价安全工具的标准，还讨论了开放源程序软件的安全内涵和安全的全公开方式。在本章的最后，概要地总结了规划和实现系统安全的几个重要准则。

1.1 UNIX：一个经验定义

再没有像UNIX这样的东西了。我意识到作为一本有关UNIX系统安全的书的开头，这是一句奇怪的话。UNIX已存在几十年了，在这几十年中，它的改变即使没有上亿次，也有数百万次了，有成千上万的个人和公司实现了上千种不同的版本，有上百万系统管理员在从微型嵌入式系统到超级计算机上都安装过它。无可争论，没有两个实际的UNIX操作系统是完全相同的。下面是几个UNIX的定义：

- 合法性——“UNIX”一词是属于Open Group的一个商标，该组织是一个要求符号得到正确归属的国际协会[1]。重复一下：“UNIX是Open Group的一个注册商标。”读者有时看到AT&T、Bell Labs、Novell或X/Open Company Ltd.作为商标拥有者列出来——已经逐渐消失了。无疑，该标识已经被冲淡到没有具体含义。虽然如此，Open Group仍发布了“The Single UNIX Specification”，这可以在http://www.UNIX-systems.org/online.html上看到。
- 技术——按照UNIX FAQ，UNIX是“一个用C语言编写的操作系统，它有层次文件系统并集成了文件和设备I/O，其系统调用接口包括fork ( )和pipe ( )等服务，用户界面包括cc、troff、grep、awk等工具和一个被选择的shell”[2]。可以再加一些，UNIX为多任务提供一致的方式，并内置有创建、同步和终止进程的操作，它可在不同种类计算机间进行移植。
- 语言——“Unix”是双关语，表示名字Multics，它最初被写作“Unics”，表示UNiplexed Information and Computing System。“Unix”和“UNIX”在如今都被广泛使用。曾经有一段时间，Dennis Ritchie试图宣布用小写版本，因为“UNIX”不是开头字母组成。与商标不同，本书用“UNIX”。
- 社会性——许多运行Linux等类似UNIX系统的人认为他们运行的是UNIX。正式UNIX系统和非正式UNIX系统通常被认为属于一类——不论是书中、媒介、网上还是社会公认。

1.1.1 UNIX简史

要定义UNIX，需要看看它的历史。1969年，Ken Thompson、Dennis Ritchie和其他一些人在AT&T贝尔实验室开始进行一个“little-used PDP-7 in a corner”的工作，它后来成为UNIX。10年里，UNIX在AT&T的发展经历了数个版本。V4（1974）用C语言重写，这成为系统间操作系统可移植性的一个里程碑。V6（1975）第一次在贝尔实验室以外使用，成为加州

大学伯克利分校开发的第一个UNIX版本的基础。

贝尔实验室继续在UNIX上工作到80年代，有1983年的System V（“五”，不是字母）版本和1989年的System V，Release 4（缩写为SVR4）版本。同时，加利福尼亚大学的程序员改动了AT&T发布的源代码，引发了许多主要论题。Berkeley Standard Distribution（BSD）成为第2个主要“UNIX”版本。1984年的BSD 4.2版在大学和公司计算部门中得到广泛应用，它的一些特征被吸收到SVR4中。

从90年代开始，AT&T的源代码许可证创造了市场的繁荣，不同开发者开发了数百种UNIX版本。AT&T在1993年把UNIX产业卖给了Novell，Novell两年后又把它卖给了Santa Cruz Operation。同时，UNIX商标被转让给X/Open协会，X/Open协会后来成为了Open Group[3]。

当UNIX的经营从一个实体到另一个实体传递时，几个长期的开发开始收获果实。传统上，要得到一个运行的BSD系统，用户需要从AT&T得到源代码许可证。但到90年代早期，Berkeley的开发者在BSD上做了许多工作，使原始的AT&T源代码大部分被改动了。后续的程序员，从William和Lynne Jolitz开始在网络分布环境中开发BSD，后来在1992年成为386BSD 0.1版。这个最初的“免费源代码”BSD具有三个分支，即：Net BSD、Free BSD和Open BSD，都以BSD 4.4[4]为基础。

1984年，程序员Richard Stallman开始开发来源于UNIX的免费GNU（GNU's Not UNIX）。到90年代早期，GNU项目出现了几个编程里程碑，包括GNU C库和Bourne Again SHell (bash)的发行。整个系统除了一个关键因素即工作内核外基本完成。

接下来是芬兰赫尔辛基大学的学生Linus Torvalds。Linus看到了一个叫作Minix的小型UNIX系统，觉得自己能做得更好。1991年秋天，他发行了一个叫“Linux”的免费软件内核的源代码——是他的姓和Minux的组合[5]。到1994年，Linus和一个内核开发小组发行了Linux 1.0版。Linus和朋友们有一个免费内核，Stallman和朋友们拥有一个免费的UNIX克隆系统的其余部分。人们把Linux内核和GNU合在一起组成一个完整的免费系统，该系统被称为“Linux”，尽管Stallman更愿意取名为“GNU/Linux System”[6]。有几种不同类别的GNU/Linux：一些可以被公司用来支持商业使用，如Red Hat、Caldera Systems和S.U.S.E；其他如Debian GNU/Linux，更接近于最初的免费软件概念。

Linux现已到内核2.2版。Linux能在几种不同体系结构的芯片上运行，并已经被各界接纳或支持。其支持者有Hewlett-Packard、Silicon Graphics和Sun Microsystems等有较长历史的UNIX供应商，还有Compaq和Dell等PC供应商以及Oracle和IBM等主要软件供应商。或许最具讽刺的是，微软承认无所不在的免费软件的竞争性威胁，但它不愿或不能公开自己的软件源代码[7]。

后来微软开始推出Windows NT (Windows 2000)。到90年代末，许多供应商开始放弃UNIX服务器平台而转向Windows NT。例如Silicon Graphics 公司已决定把Intel硬件和NT作为未来的图形平台。

1.1.2 一个经验定义

历史悠久的UNIX供应商转头大步奔向Linux的现象多少把我们带回本节开头的问题：什么是UNIX？

当人们遵循体现在商标中的法律定义时，笔者相信这会伤害UNIX工业。作为Internet软件

的基础，UNIX技术是20世纪文明的一个显著成绩。限制UNIX在狭窄的合法性或技术性定义中——由一些供应商形成，现又被抛弃，会否定它向前发展的本质和重要性。与UNIX相像的产品如CNU/Linux和BSD的广泛流行与强大便充分证明了这一点。

为了本书的目标，我把UNIX定义为：

一组由AT&T最早形成的技术，已经被合法分为几种不同但又密切相关的操作系统，每种都能被认为是一个“UNIX系统”。如果它看起来像UNIX，操作起来像UNIX，运行通用的UNIX工具和程序，并以UNIX作为模型来开发，则它就是UNIX。

1.1.3 UNIX的多种不同版本

表1-1总结了一些常用的UNIX版本。表中列了大约40种不同版本，但UNIX世界不像过去那样多种多样了。有一些已不使用，列出它只是由于历史原因。其他一些正在消亡的过程中。某些情况下，供应商正投奔到微软阵营中，其余通过合并和发展导致不同UNIX产品的统一。

表1-1 UNIX的不同版本

UNIX 版本	公司/机构	更多信息
A/UX	Apple Computer,Inc	已不存在
AIX	IBM	http://www.rs6000.ibm.com/software/
AT&T System V	AT&T	已不存在
BS2000/OSD-BC	Siemens AG	http://www.siemens.com/servers/bs2osd/
BSD/OS	Berkeley Software Design, Inc	http://www.bsdi.com
CLIX	Intergraph Corp.	http://www.intergraph.com
Debian GNU/Hurd	Public Interest, Inc.软件部	http://www.gnu.org/software/hurd/debiangnu-hurd.html
Debian GNU/Linux	Public Interest, Inc.软件部	http://www.debian.org
DG/UX	Data General Corp.	http://www.dg.com/products/html/dg_ux.html
Digital Unix	Compaq Computer Corporation	http://www.unix.digital.com/
DYNIX/ptx	Sequent Computer Systems, Inc.	http://www.sequent.com/products/software/operatingsys/dynix.html
Esix UNIX	Esix Systems	http://www.esix.com/
FreeBSD	Free BSD group	http://www.freebsd.org
GNU Herd	GNU organization	http://www.gnu.org
HAL SPARC64/OS	HAL ComputerSystems, Inc.	http://www.hal.com
HP-UX	Hewlett-Packard Company	http://www.hp.com/unixwork/hpux/
Irix	Silicon Graphics,Inc.	http://www.sgi.com/software/irix6.5/
Linux	多个	http://www.linux.org
LynxOS	Lynx Real-Time Systems, Inc.	http://www.lynx.com/products/lynxos.html
MachTen	Tenon Intersystems	http://www.tenon.com/products/machten/
Mac OS X Server	Apple Computer, Inc.	http://www.apple.com/macosx/
Minix	none	http://www.cs.vu.nl/~ast/minix.html
MkLinux	Apple Computer, Inc.	http://www.mklinux.apple.com
NCR UNIX	NCR Corporation	http://www3.ncr.com/
SVR4 MP-RAS		product/integrated/software/p2.unix.html
NetBSD	NetBSD group	http://www.netbsd.org
NeXTSTEP	NeXT Computer Inc.	defunct,see http://www.apple.com/enterprise/
NonStop-UX	Compaq Computer Corporation	http://www.tandem.com
OpenBSD	OpenBSD group	http://www.openbsd.org

（续）

UNIX 版本	公司/机构	更多信息
OpenLinux	Caldera Systems, Inc.	http://www.calderasystems.com
Openstep	Apple Computer, Inc.	ttp://www.apple.com/enterprise/
QNX Realtime OS	QNX Software	http://www.qnx.com/
	Systems Ltd.	products/os/qnxrtos.html
Red Hat Linux	Red Hat Software, Inc.	http://www.redhat.com/
Reliant UNIX	Siemens AG	http://www.siemens.com/servers/rm/
Solaris	Sun Microsystems	http://www.sun.com/software/solaris/
SunOS	Sun Microsystems	已不存在
SuSE	S.u.S.E.,Inc.	http://www.suse.com
UNICOS	Silicon Graphics,Inc.	http://www.sgi.com/software/unicos/
Unix Ware	SCO-The Santa Cruz Operation Inc.	http://www.sco.com/unix/
UTS	Amdahl Corporation	http://www.amdahl.com/uts/

1.2 计算机安全：一个经验定义

定义“计算机安全”并不是很容易的，困难在于要形成一个足够全面而有效的定义，不论描述什么系统，所声明的都足够用来描述什么是真正的安全。通常的感觉下，安全就是“避免冒险和危险”。在计算机科学中，安全就是防止：

- 未授权的使用者访问信息。
- 未授权而试图破坏或更改信息[8]。

这可以重述为“安全就是一个系统保护信息和系统资源相应的机密性和完整性的能力”。注意第二个定义的范围包括系统资源，即CPU、硬盘、程序以及其他信息。

1.2.1 计算机安全的分类

计算机安全通常与三个方面相关，它们可以简写为“CIA”：

- 保密性（Confidentiality）——确保信息不被非授权用户访问。
- 完整性（Integrity）——确保信息不被未授权用户更改，但对授权用户开放。
- 确定性（Authentication）——确保用户就是它所声明的使用者。

一个强壮的安全协议强调所有这三个方面。例如Netscape的SSL（Secure Sockets Layer）协议能够发现哪些是真正可信的（更确切地说是不可信的）。SSL克服了事务处理中缺少可信度的问题，如通过译码确保保密性，通过校验和来确保完整性，通过服务器认证保证确定性。参见15.5节。

计算机安全并不限于这三个概念。在为计算机安全分类时还有一些需要考虑：

- 访问控制——确保用户仅能访问那些被授权访问的资源和服务，使有资格的用户能够使用他们想得到的合法服务。
- 无可拒认——确保消息的发出者不能拒认发送消息的事实[9]。
- 可用性——保证一个系统在给定时刻是可以正常操作的，通常通过冗余来提供；失去可用性通常指“拒绝服务”。
- 隐私权——保证个人拥有控制自己信息的权利，如怎样使用，谁能够使用，谁来维护和为了什么目的使用等。

这些额外内容不能简洁集成到一个定义中。从某些方面看，隐私权、保密性和安全的概念有相当大的区别且有不同的特征。隐私权是个人的属性；保密性是数据的属性；安全是赋予计算机硬件和软件系统的属性。从实际来看，这些概念是交织在一起的。一个不能维护数据保密性或个人隐私权的系统可能理论上相当“安全”，但它不可能在现实世界中得到应用。

1.2.2 功能概述

计算机安全可以通过功能来分析。它可以被分成五个不同的功能领域：[10]

- 避免冒险——安全的基础可能从这样的问题开始：我的组织或业务预约是否太冒险？我是否真的需要一个不受限制的Internet连接？我们是否真得需要使安全的业务过程电脑化？我们是否应该使没有访问控制的桌面操作系统标准化？
- 制止——减少对信息资产的威胁。能够组织通用策略来提高潜在攻击者被抓到的可能性。参见1.4节。
- 预防——计算机安全的传统核心。由本书中讲到的工具那样的保护产品组成。绝对预防只是理论上的，因为当传统预防措施不再有效时会出现缺陷。
- 检测——最好与预防措施一起使用。当预防失败时，就应进行检测，更喜欢在还有时间防止破坏时检测。包括日志保存和审计活动。
- 恢复——当其他方式都失败时，准备拿出备份介质并从其上恢复，或切换到备份服务器和网络连接上，或后退到灾难恢复设备上。该功能应在其他之前进行。

1.2.3 安全领域

计算机安全常通过几个互相依赖的领域来定义，能大概映射到特定的部位或工作主题：

- 物理安全——控制人和物资的来往，保护并防止自然灾难。
- 操作/步骤安全——包括从管理决策到报告体系的所有事情。
- 人事安全——雇佣员工，背景调查，培训，安全简报，监测和控制人员离去。
- 系统安全——用户访问和确认控制，授权，维护文件和文件系统完整，备份，监测进程，日志保存和审计。
- 网络安全——保护网络和远程通信设备，保证网络服务器和传输，反窃听，控制从不可信网络上的访问，防火墙和监测侵入。

本书只关心后两种。系统和网络安全是困难的，如果可能，在一个UNIX系统中分开考虑。过去15年中几乎每个UNIX分布式系统都包括TCP/IP协议实现和数字网络服务，例如FTP、Telnet、DNS和最近的HTTP。

1.2.4 一个经验定义

由于实用性的原因，作者喜欢Simson Garfinkel和Gene Spafford在《UNIX & Internet Security》中提出的定义：“如果某台计算机及其软件按用户期望运行，则称这台计算机是安全的”[11]。本质上，一台计算机在用户可以相信它时是安全的。今天输入的数据月份以未改变的形式放在那里。如果使服务x、y和z昨天可用，则今天它们仍然可用。

作者还喜欢Tomas Olovsson提供的一个经验定义，尽管它有一点片面：“一个安全的系统是一个在使用它的信息时能给予足够信任的系统”[12]。

本质上，一个安全的系统应该很难被未授权人员侵入——也就是，因一次未授权侵入而进行工作的价值要比保护数据的价值高。增大攻击者的工作量和被检测到的风险是计算机安全的重要因素。

就本书而言，作者定义“系统安全”为：

持续运行并且有冗余的方法，用来保护信息和系统资源的保密性和完整性，使得未授权用户必须花费不可接受数量的时间或金钱或冒太大的危险来攻击它，最终的目标是使具有保密信息的系统值得信任。

1.3 UNIX安全工具的世界

UNIX安全的历史充满了教训。自从它在60年代末作为Dennis Ritchie和Ken Thompson领导的Bell实验室项目后，UNIX操作系统已经连续被数代系统程序员进行改造和攻击。

1988年11月，Robert Tappan Morris发布了罪恶的“Internet蠕虫”程序，它在一夜之间破坏了网络连接的数千台机器[13]。Morris，一个美国国家高级安全官员的儿子，把UNIX安全性问题推向世人注目的位置。蠕虫暴露了UNIX中潘朵拉盒子的弱点，包括已有很长历史的sendmail和finger程序中的缺陷。它还引出了UNIX中“可信主机”的概念——一种作为Berkeley网络软件的一部分而开发的机制，使用户能在远程机器上执行命令。除了立即攻击和感染系统，蠕虫引出了对于UNIX安全性的“open lab”问题，它在普通的安全控制下最大化消耗共享资源。它代表了一个转折点——从蠕虫开始，许多UNIX机器要重新考虑它们的安全标准。

被Morris的蠕虫攻击惊醒后，UNIX组织开始考虑用更多精力来加强安全性。几个主要的组织成立了CERT（Center for Emergency Response Teams），现在它已成为UNIX系统安全信息的首要来源[14]。供应商推出补丁程序并努力使用户采纳他们的升级方法。更重要的是UNIX程序员开发了一系列新的安全工具并使它们在Internet上免费使用。

结果现在每个UNIX系统管理员都有自己的安全工具箱。这里许多工具和功能是内置于现代UNIX产品中的。事实上有一个主流趋势即把安全工具与商业和免费的UNIX产品捆绑在一起。在许多情况下，UNIX系统需要加固——因为所需功能不包括或者不够用。过去的某些系统也许缺乏安全保护。在其他状况下，甚至现代的UNIX也需要额外的安全功能。不论是使用过去用来实现简单安全保护的NeXT工具箱，还是拥有最新一流安全程序的Solaris 7工具箱，系统安全标准需要根据面临的安全性弱点来定制，而其他的安全性工具只能起到辅助作用。

注意许多UNIX安全工具是“免费”的，不用花钱且在使用时也没有许可证或其他什么限制。要预先警告：它们中没有一个在寻找、理解、应用和运行的时间上是“免费”的。要理解Richard Stallman所说的：要更多考虑“免费演讲”而不是“免费啤酒”。

“免费工具”趋势在过去几年中随着“公开源代码”运动的兴起而加速发展。Internet促进了广泛的经济发展，因为价值——从免费内容到免费程序——被自由地交换。Internet为程序员虚拟小组的形成提供了便利，他们开发各种软件，从操作系统（Linux）到服务器软件（Apache Web服务器），再到复杂的终端用户应用程序（Mozilla Web浏览器）。

1.3.1 安全性工具评价

对于系统管理员是否应安装并运行一个给定安全工具的问题，没有固定的回答。某认证

机构建议“选择自己适用的武器”[15]。当评价本书中出现的各种工具时，作者根据以下特点：

• 该工具的源代码是否可用？

在没有源代码的情况下无法审计或整体评价一个安全工具。有许多好的商业工具不提供源代码，作者把它们从本书中去掉了。封闭专有产品需要对供应者有高的信任度。不接触源程序就无法确信产品没引入漏洞或存在后门。要对该问题进行更多讨论，请参见1.3节。通常，安全保护程序——从门锁安全到密码方式——不应依赖设计和实现保密。

• 该工具是否易于安装？

这里的“易于”有一些微妙，因为在所有UNIX安全性工具中没有一个不用脑子就可以安装。他们都需要一些思考和努力。在许多情况下，也许非常有用的工具根本就不能在指定的机器上安装。另一些情况下，工具设计和实现的非常优美——安装没有一点问题，使得可以把时间和精力花费在配置和使用上，而不是编译出错等。

• 该工具是否可在特定环境中配置？

没有两个UNIX系统是完全一致的。一个好的工具应该灵活，设计者应考虑工具可能使用的各种环境并让用户自定义。

• 工具是否可靠？

一个不可靠的安全工具可能比不用安全工具更糟糕，因为它给系统管理员造成了一个安全的假象。

• 该工具是否合理易用？

一个工具的常用操作应该是一件相对简单的事情。人的天性是对复杂或过于难用的工具有抵触——一个太笨重的工具将不被使用并且不能提供所需的保护。同样，工具应产生可读的输出。简洁的报告是可行的，因为任何人从数据中提炼所需内容的能力是有限的。Crack工具在提供贴切的输出方面是一个较好的例子：如帐号×××××有一个薄弱的密码等。这个问题解释了一个基本原则：为了达到有效性，安全保护程序必须可接受。

• 工具的开销是否高效？

时间和金钱都要考虑。每个工具都提供某类好处。这些益处需要针对应用和运行工具的开销来权衡。在某些情况下，一个工具也许把性能降低到不可接受的状态。这是一个普遍问题，例如有些防火墙产品。

• 工具是否可维护？

有许多可用的工具因为这个或那个原因不再由最初开发者维护。通常，安全保护程序应在某段时间内持续不变，一个被“遗弃”的旧工具也许随着变迁不再适合于整个安全环境。另一方面，有时旧工具可能非常有用，特别是对于旧系统，Dan Farmer的COPS工具最早在1990年写成，但作者仍能用它来发现自己的1990 NeXT工具箱中某一精选工具的几个漏洞。有时，旧工具被新的维护者拣起并发展。一个例子是SATAN网络搜索工具，它由Dan Farmer和Wietse Venema编写，然后由World Wide Digital Security公司中的一群程序员升级（被重新命名为“SAINT”）。

• 工具是否依赖于其他（可能不安全的）程序？

UNIX的方式是组合不同程序来获得期望结果。当一个安全工具依赖一个不是为了安全目标设计的复杂UNIX程序时可能会导致相反结果。Wietse Venema讲述了一个在埃因霍温理工大学发生的为侵入者设置陷阱却被侵入并删除了系统的故事[16]。陷阱依赖于找出侵入者的主

机并把结果发送给root。后来，Wietse认识到依靠finger和mail程序（都不是为安全设计的）可能打开了安全漏洞。侵入者可以把Shell转义符包含在他的.plan文件中，它被陷阱挑出并送给root来执行。该故事的寓意在于：好的安全工具或者独立或者最小程度地依赖其他程序。

• 该工具是否可在不同UNIX产品间移植？

许多站点运行多种UNIX系统。一个编码完整的工具应能在系统间移植。如果为平台A所写，则它应能被用在平台B上。当然，也有很好的特定平台工具，所以这只是一个普通规则。

• 该工具是否有害处？

这应该毫无疑问。“没有害处”应该成为任何安全职业的首要前提。近来在市场上出现的许多安全工具号称能“击败”以前的工具。以国防部系统为例，它们能检测到试图的入侵并自动进行反击。作者认为广泛应用这种有害的工具是一个坏主意。这些系统可以很容易地攻击一个无辜的聚会。事实上，它为入侵者打开了一个新的攻击线。在系统C上的破坏者可能假装成系统B并攻击系统A，于是A将自动反击B。这被称为“用一个信息包杀死两个系统”。

1.3.2 公开源代码的益处

UNIX操作系统和程序的源代码广泛传播对于安全管理员是一种福音。为程序提供源代码在计算机界是一个古老传统。近来，随着Linux操作系统和Apache程序这些公开源代码的流行，该传统已经变的正式了。一个由Eric S.Raymond领导的免费软件传播者群体形成了一个非赢利组织，其注册为“Open Source”（参见http://www.opensource.org/[17]）。

本书中几乎所有的安全工具都是公开源代码的，就像作者使用的Red Hat Linux平台一样。安全软件公开源代码的好处非常多。就像科学界进行的本质评论一样，公开的源代码可以被成百上千的有关参与者进行分析、查核并诊疗。缺陷和错误可以很快找到并修复，为程序员在程序中放置后门、特洛伊木马和其他恶意代码制造了障碍。这种方式下，公开代码的软件比私有软件更值得信任，解释Raymond所说的“Linus's Law”（纪念Linux创造者Linus Torvalds）：“给足够多的眼睛，则所有缺陷都可找出”[18]。给出源代码，安全问题能很快找出并进行修复。

1999年1月，攻击者把工具TCP/Wrappers的一个特洛伊木马版本放在著名的FTP站点上——因为源代码公开，所以后门被迅速注意到并删除。与Windows 2000这种单一操作系统相比，后者有上千万行秘密的、存在缺陷的代码。没接触到源代码，顾客100%信赖微软公司的产品和能力，它有一个很好文档记录可疑的软件行为[19]。

自然，一个程序源代码可用并不意味着它是安全的——作者所称的“通过暴露达到安全”并不简单（尽管它通常比“通过隐藏到达安全”更强壮）。一个程序员创建干净的不含特洛伊木马的源代码在理论上也是可能的（参见1.4节）。通常，源代码需要进行广泛的区分、复查和修改，以其公开性成为一个安全的软件。也有一些情况是，公开源代码且受欢迎的UNIX工具因明显的弱点而在多年后失败[20]——也许人们认为已通用的程序是肯定安全的，或者人们只是太懒或太忙而不去进行修改。相反，有些最安全的程序可能在国家安全机构应用，但其源代码却是高度保密的。

有时候，公开源代码使“坏人”更容易首先找到安全漏洞，于是出现了一个*持续的*“猫和老鼠”游戏，开发者试图在漏洞被发现之前找到并修复它们。这就出现了“阳光效应”。Karsten M.Self这样描述它：[21]

“依然存在我称之为阳光效应的东西——好人可以公开行动，而坏人必须偷偷行动。每个人(好人或坏人）对自己系统是否安全很感兴趣。攻击将会发生，但警告和修复都很迅速。当破坏者用假名、匿名邮件发送者、自己的系统和IRC进行操作时，合法操作者可以公开用可信的信息信道进行通信（例如CERT)”。

团体在公开代码软件中扮演重要角色。像Self指出的，每个UNIX操作者都对安全感兴趣。这种公众的兴趣在公开源代码的空间中遨游，自由交换思想和代码使每个人都获益。如果对一个公开代码的工具存在疑问，则能很快得到一个解答。

公开安全工具源代码还有其他好处。它们大多数或多或少“免费”，于是解答对所有有时间且需要安装它们的人有用。这使“小家伙”和大公司站在同一个山脚下。一个公开代码的工具使系统管理员可以控制在应用工具时的风险程度。也许用户不信任公开代码的开发者而更愿相信私有开发者，那么用户可以自己审核代码或雇人来进行。最终，公开代码的方案自然形成。也许一个工具并不是用户确切需要的，那么用户可以自己进行修改。公开源代码比封闭产品提供更大灵活性。如果用户确实想发展一个公开工具，则可以把它返回给原始者并进行大量通信。拿来拿去经济使每个人获益。

1.4 10个通用安全准则

在进入到规划和实现UNIX系统安全的细节之前，作者在这里先列出10个通用准则或定理，本书将一次又一次地重复它们。

准则1：通过隐藏来实现安全是行不通的。

就像电影里所说的，你可以跑但不能藏。用户也许会想自己运行的是一个不出名的UNIX Web服务器，没有人会想着侵入它。但在这个时代，成千上万带恶意的年轻人会通过接触强大的网络扫描工具发现用户的系统及其弱点，所以躲避是不能保证安全的。也许用户认为已把关键数据藏在好几层目录之下了，但当看到UNIX中强大的搜索工具时就会觉得自己错了。软件或硬件供应者也许认识到自己的产品存在漏洞但仍四处销售，想着没有人会发现它。这些漏洞总会被发现的。

通过隐藏最多只能得到临时的安全保护，但不要被它所蒙骗——花很少的努力和时间就会发现秘密。像Deep Throat在X-档案中指出的：“总有人在监视着你”。

准则2：完全暴露缺陷和漏洞对安全有好处。

像上面所说的，有些供应商在销售有安全漏洞的软件时心安理得，期望着软件足够复杂和保密而无人会发现它们——树在倒的时候没有人听见就不能算倒。有些安全职业人员在公布发现的安全漏洞和问题时总感到心里不安。他们担心宣布了安全漏洞会给“坏人”提供怎样攻击系统的思想。另一方面，Internet上的安全组织把有关漏洞和可能问题的知识进行共享：许多像bugtraq这样的邮件列表和comp.security.unix这样的新闻组都维持公开的讨论来试图指出并解决漏洞。这有些似是而非，但公开暴露安全问题的方式对整个Internet及其上面系统的安全都有好处。通过暴露来达到安全是可行的。注意：这并不是说应该在发现一个漏洞后马上广泛地公开宣传。协议要求首先与系统供应商或程序的编写者联系，给他们一个机会来修改。发现并宣布安全漏洞是好事，但最好是在修复后再宣布。

准则3：按直接使用的比例降低系统安全级别。

这是Farmer定律（计算机安全研究人员Dan Farmer提出）：“计算机系统的安全要按直接

使用系统的人数按比例降低级别”[22]。

暂时忽略可用性，一台关闭的计算机比开启的更安全。一台关闭的计算机，锁在箱子里，放在地下防空洞，由看守守卫就更安全了。一旦有一个人使用它，则危险增加，一旦两个或更多人使用系统，则危险增加更多。把系统放入Internet并提供某些服务…可以肯定用户能接受这个想法。像Dan说的，“无知或有恶意的用户对系统安全的破坏比其他因素都大”[23]。

在安全和使用/功能之间权衡是经典的计算机安全难题。许多Linux发布系统为最多功能而开发，在销售时有大量的程序集和开放的安全设置。另一个极端是防御主机作为防火墙的一部分而建立了。它们有许多只做一件事（例如在网络A和网络B之间过滤包）。分析一下自己的需要来在安全和功能之间进行权衡并做出正确规划。

准则4：在某些人犯错之前把它做好。

计算机安全从不可能在真空中应用。仅建立安全机制并不能保证它们会按计划工作。安全措施和机制必须面对用户的正常需要：也就是必须正确工作。一个机构可以宣告没有用户能拥有Internet访问权，但将会发现聪明的用户会买便宜的调制解调器来对付该措施，这样就会增加该机构的弱点。最好是设置更实际的措施并对访问网络进行监测和控制。一个防火墙管理员可能决定应用一个法西斯式的防火墙，仅允许通过80接口的HTTP/Web访问，让需要Telnet访问的用户到其他地方去。结果，这些用户也许发现可以把禁止的协议封装在HTTP包中。管理员最好提供正常的需求而不是鼓励绕过潜在和未知的冒险。最好是自己把事情做好，而不是等别人来犯错误。

准则5：担心被抓住是聪明的开始。

不要低估障碍的价值。许多潜在的攻击可能会因攻击者逐渐害怕而被制止[24]。障碍可能对业余的白领罪犯或内部人员特别有效。其目标是阻止攻击者试图到达关键的行动。有许多保护程序可以阻止一次攻击，从登录标志警告“警告！使用本系统应同意安全监测和测试。所有活动都和你的主机名及IP地址一起记录下来”[25]到提醒与计算机相关的法律、背景检测、安全简报和审查等。当然，这些保护程序不能阻止坚定的计算机罪犯，但甚至职业罪犯在勘测一个新系统后发现一个像Tripwire这样监测工具已配置好把每天的文件系统完整性报告写入只读介质时，也会再三考虑的。

准则6：总有某个人比你更精明，拥有更多知识和更精良的设备

要仔细设想自己系统所面临的相关威胁。甚至冗余安全机制和仔细的监测也不能保护用户免遭uebercracker的攻击。考虑下面从Dan Farmer和Wietse Venema的文章“Improving the Security of Your Site by Breaking Into It”中摘录的一段话：[26]

“什么是uebercracker？很明显它是从尼采的uebermensch偷来的，文字上译成英语是“over man”。尼采并没用它指连环画中的超人，而是指一个人超越普通人的无能、卑劣和虚弱。uebercracker是已超越用简单方式侵入系统的系统破坏者。uebercracker通常没有动机来执行随机的破坏行为。目标不是随意的——它有一个目标，不论是个人金钱利益、搜索各种信息或袭击一个重要的或著名的站点或网络。一个uebercracker很难被检测到，更难被制止，最难的是把他屏蔽在站点之外。”

许多安全威胁模型假设一个坏人是一个孤癖的人或是对搜索系统有兴趣的小家伙。冗余安全机制和监测也许会使系统在这些模型下得到保护，但它不能阻止一个坚定并有技巧的职业人员——uebercracker。

比uebercracker更有威胁的是攻击单元—— 一组复杂的人在一起工作来攻击系统以达到某个目的。当一个机构准备预防孤僻的攻击者时，也许会遭到一些有足够金钱和技术资源的职业人员的攻击。一个攻击单元也许包括一个从市场上雇佣的社会工程专家，一个用“UNIX工具针对UNIX工具”来解剖用户网络的系统专家，一个花费数年开发用户工具的安全程序员和一个通过中间媒介传递信息的话务员通。它也许有显著的研究和开发能力，或者甚至是政府机构的后援[27]。本书中讨论的所有工具和技术仅在书中有效。若读者的UNIX系统（或任何系统）包含商业或政治机密，则要准备对安全管理、物理安全和人员安全进行实质的调查，并要对系统和网络安全进行重点的研究。

准则7：没有完全包办的安全方案。

商业和Internet相连接后，就希望能买到完全包办的安全措施。安全提供者不会同意，作者不相信有完全包办的方案。有太多的因素需要考虑，也存在太多种类的安全措施、威胁模型、系统配置和连接。用户要避免Maginot Line症状：依赖单独一个防火墙这样的保护程序也许会导致系统的失败。安全不是买来作为一次性事情处理的，它是一个持续的过程，需要持续的计划、监测和求精。

该准则的一个标准为：没有一个检查表能包括所有的弱点。安全性检查表是一个检查错误和疏忽的著名的方式，但不要被它们所蒙蔽。安全检查表方法将在一个有智慧的攻击者面前失效，若他已看到公布的检查表并设计出不在其中的攻击方式。

准则8：好与坏混合起来就成为灰色。

所有计算机安全的定义都包含一个隐藏的假像：存在“好人”和“坏人”，或是“白帽”和“黑帽”。几乎所有流行的书和电影都沉迷这种假像，从Clifford Stoll的The Cuckoo's Egg中狡猾的伯克利黑客追捕国际间谍到The Net中Sandra Bullock扮演一个被贼抢掠的系统管理员。有时，计算机安全的对抗本质把它变成了一种游戏。不幸的是对于安全人员，这种游戏是“与未知的敌手在未知的时间和地点策划未知的破坏进行对抗”[28]。当有人对用户的系统安全感到关注时，用户将感到自己在一个盛大的戏剧中扮演角色，本书和相关工具可以帮助用户对抗那些无时无地不存在的无名攻击者。

作者建议用户不要坠入该假像。用户也许给了自己太多的荣誉而给“对手”的太少。不要忽略一个事实，就是多数安全违规是由公司内部人员造成的。罪犯看起来更像一起在餐桌旁坐着的、不引人注目的家伙，而不是电影中精通技术的狂徒。永远不要忘了在每个白帽和黑帽角色之间，还有上百个戴着灰色帽子。

计算机安全职业包含了大量的人员，包括受过高等教育的人、退伍军人、商业供应商的推销员以及被改造过的破坏者。没有人小看计算机安全职业中所需技术的发展，也没有一个可接受的道德规范。这种情况下，职业人员类似于他的敌人：广大的、多民族的、多文化的攻击团体，从开发复杂程序的“老手”到从幼年开始就被教育说计算机犯罪就像双击一个简单的“小家伙”。

与其鲜明地把维护计算机和破坏计算机的人分开，不如考虑两种人有多紧密的关系并有大量的人掉入之间的灰色区域。考虑一下“tiger team”这个团体——他们是计算机安全职业者，被雇来以攻击手段检测系统的安全。某些情况下，这些团体由被改造的、以前有着罪恶史的系统破坏者组成。甚至IBM把该服务以“道德的黑客”来进行宣传。另一方面，许多攻击者屡次去重要的防御站点——通过侵入来宣布他来过，这实际是给受害者帮忙。相反，在