*Securing Solaris, Mac OS X, Linux & FreeBSD*

3rd Edition
Extensively Revised
Over 250,000 copies in print

# Practical Unix & Internet Security


O'REILLY

*Simson Garfinkel, Gene Spafford & Alan Schwartz*

# Practical Unix & Internet Security, 3rd Edition


## Simson Garfinkel

## Gene Spafford

## Alan Schwartz



O'REILLY®

Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo

# Special Upgrade Offer

If you purchased this ebook directly from oreilly.com, you have the following benefits:

- DRM-free ebooks — use your ebooks across devices without restrictions or limitations
- Multiple formats — use on your laptop, tablet, or phone
- Lifetime access, with free updates
- Dropbox syncing — your files, anywhere

If you purchased this ebook from another retailer, you can upgrade your ebook to take advantage of a
these benefits for just $4.99. Click here to access your ebook upgrade.

*Please note that upgrade offers are not available from sample content.*

# A Note Regarding Supplemental Files

Supplemental files and examples for this book can be found at http://examples.oreilly.com/9780596003234/. Please use a standard desktop web browser to access these files, as they may not be accessible from all ereader devices.

All code files or examples referenced in the book will be available online. For physical books that sh with an accompanying disc, whenever possible, we've posted all CD/DVD content. Note that while w provide as much of the media content as we are able via free download, we are sometimes limited by licensing restrictions. Please direct any questions or concerns to booktech@oreilly.com.

# Preface

It's been 11 years since the publication of *Practical Unix Security* — and 6 years since *Practical Uni and Internet Security* was published — and oh, what a difference that time has made!

In 1991, the only thing that most Americans knew about Unix and the Internet was that they were some sort of massive computer network that had been besieged by a "computer virus" in 1988. By 1996, when our second edition was published, the Internet revolution was just beginning to take hold with more than 10 million Americans using the Internet on a regular basis to send electronic mail, cruise the World Wide Web, and sometimes even shop.

Today it is increasingly difficult for people in much of the world to remember the pre-Internet era. Perhaps 500 million people around the world now use the Internet, with several billion more touched by it in some manner. In the United States more than half the population uses the Internet on a daily basis. We have watched an Internet revolution become a dot-com craze, which then became a bust. And nobody remembers that 1988 Internet worm anymore — these days, most Internet users are bombarded by network worms on a daily basis.

Despite our greater reliance on network computing, the Internet isn't a safer place today than it was i 1991 or in 1996. If anything, the Internet is considerably less secure. Security mishaps on the Interne continue to be front-page stories in newspapers throughout the world. Sadly, these flaws continue to be accommodated rather than corrected.[1] The results are increasingly disastrous. The second edition of this book, for example, noted a security incident in which 20,000 people had their credit card numbers stolen from an Internet service provider; a few months before this third edition went to prin attackers broke into a system operated for the State of California and downloaded personal information on 262,000 state employees. Included in the haul were names, addresses, Social Security numbers — everything needed for identity theft.[2]

Computer crime and the threat of cyberterrorism continue to be growing problems. Every year the Computer Security Institute (CSI) and the San Francisco Federal Bureau of Investigation (FBI) Computer Intrusion Squad survey organizations to find their current level of computer crime and intrusions. The 2002 survey had 503 responses from security practitioners in U.S. corporations, government agencies, financial institutions, medical institutions, and universities. Some of the result of the survey include:

- Ninety percent of respondents (primarily large corporations and government agencies) detected computer security breaches within the last 12 months.[3]
- Eighty percent acknowledged financial losses as a result of system security breaches.
- The combined loss of the 223 respondents who gave dollar values for their annual loss was more than $456 million, of which $171 million was the theft of proprietary information, and $116 million was financial fraud.
- Contrary to conventional wisdom that insiders are a bigger threat than outsiders, 74% of respondents cited their Internet connection as a frequent point of attack, versus 33% who cited their internal systems as a frequent point of attack. (Of course, insiders could be attacking through the Internet to make themselves look like outsiders.)
- Slightly more than one-third (34%) reported the intrusions to law enforcement — up from 16% reporting in 1996.

Incidents reported included:

- Computer viruses (85%)
- Employees abusing their Internet connection, such as downloading pornography or pirated software, or sending inappropriate email (78%)
- Penetration from outside the organization (40%)
- Denial of service (DOS) attacks (40%)
- Unauthorized access or misuse of the company's web sites (38%)

One quarter of the respondents who suffered attacks said that they had experienced between 2 and 5 incidents; 39% said that they had experienced 10 or more incidents. The average reported financial loss per company per year was in excess of $2 million.

What do all of these numbers mean for Unix? To be sure, most of the systems in use today are based on Microsoft's Windows operating system. Unix and Unix variants are certainly more secure than Windows, for reasons that we'll discuss in this book. Nevertheless, experience tells us that a poorly-administered Unix computer can be just as vulnerable as a typical Windows system: if you have a vulnerability that is known, an attacker can find it, exploit it, and take over your computer. It is our goal in this book to show you how to prevent yourself from ever experiencing this fate — and if you do, it is our goal to tell you what to do about it.

# Unix “Security”?

When the first version of this book appeared in 1991, many people thought that the words “Unix security” were an oxymoron — two words that appeared to contradict each other, much like the word “jumbo shrimp” or “Congressional action.” After all, the ease with which a Unix guru could break in a system, seize control, and wreak havoc was legendary in the computer community. Some people couldn’t even imagine that a computer running Unix could ever be made secure.

Since then, the whole world of computers has changed. These days, many people regard Unix as a relatively secure operating system. While Unix was not originally designed with military-level security in mind, it was built both to withstand limited external attacks and to protect users from the accidental or malicious actions of other users on the system. Years of constant use and study have made the operating system even more secure, because most of the Unix security faults have been publicized and fixed. Today, Unix is used by millions of people and many thousands of organizations around the world, all without obvious major mishaps.

But the truth is, Unix really hasn’t become significantly more secure with its increased popularity. That’s because fundamental flaws still remain in the operating system’s design. The Unix *superuser* remains a single point of attack: any intruder or insider who can become the Unix superuser can take over the system, booby-trap its programs, and hold the computer’s users hostage — sometimes even without their knowledge.

One thing that has improved is our understanding of how to keep a computer relatively secure. In recent years, a wide variety of tools and techniques have been developed with the goal of helping system administrators secure their Unix computers. Another thing that has changed is the level of understanding of Unix by system administrators: now it is relatively easy for companies and other organizations to hire a professional system administrator who will have the expertise to run their computers securely.

The difference between a properly secured Unix system and a poorly secured Unix system is vast, an the difference between a system administrator with the knowledge and motivation to secure a system and one without that knowledge or motivation can be equally vast. This book can help.

## What This Book Is

This book is a *practical* guide to security for Unix and Unix-like (e.g., Linux) systems. For users, we explain what computer security is, describe some of the dangers that you may face, and tell you how to keep your data safe and sound. For administrators, we explain in greater detail how Unix security mechanisms work and how to configure and administer your computer for maximum protection. For those who are new to Unix, we also discuss Unix’s internals, its history, and how to keep yourself from getting burned.

Is this book for you? If you administer a Unix system, you will find many tips for running your computer more securely. If you are new to the Unix system, this book will teach you the underlying concepts on which Unix security is based. If you are a developer, this book will give you valuable details that are rarely found together in one place — it might even give you an idea for a new security product.

We’ve collected helpful information concerning how to secure your Unix system against threats, both internal and external. In most cases, we’ve presented material and commands without explaining in any detail how they work, and in several cases we’ve simply pointed out the nature of the commands and files that need to be examined; we’ve assumed that a typical system administrator is familiar wit

the commands and files of his system, or at least has the manuals available to study.

**A NOTE ABOUT YOUR MANUALS**

Some people may think that it is a cop-out for a book on computer security to advise the reader to read her system manuals. But it's not. The fact is, computer vendors change their software much faster (and with less notice) than publishers bring out new editions of books. If you are concerned about running *your* computer securely, then you should take the extra time to read your manuals to verify what we say. You should also experiment with your running system to make sure that the programs behave the way they are documented.

Thus, we recommend that you go back and read through the manuals every few months to stay familiar with your system. Sometimes rereading the manuals after gaining new experience gives you added insight. Other times it reminds you of useful features that you haven't used yet. Many successful system administrators have told us that they make it a point to reread all their manuals every 6 to 12 months!

Certain key parts of this book were written with the novice user in mind. We have done this for two reasons: to be sure that important Unix security concepts are presented to the fullest and to make important sections (such as those on file permissions and passwords) readable on their own. That way this book can be passed around with a note saying, "Read Chapter 4 to learn about how to set passwords."[4]

## What This Book Is Not

This book is not intended to be a Unix tutorial, nor is it a system administration tutorial — there are better books for that (see Appendix C), and good system administrators need to know about much more than security. Use this book as an adjunct to tutorials and administration guides.

This book is also not a general text on computer security — we've tried to keep the formalisms to a minimum. Thus, this is not a book that is likely to help you design new security mechanisms for Uni although we have included a chapter on how to write more secure programs.

We've also tried to minimize the amount of information in this book that would be useful to people trying to break into computer systems. If that is your goal, then this book probably *isn't* for you.

We have also tried to resist the temptation to suggest:

- Replacements for your standard commands
- Modifications to your kernel
- Other significant programming exercises to protect your system

The reason has to do with our definition of *practical*. For security measures to be effective, they nee to be generally applicable. Most users of Solaris and other commercial versions of Unix do not have access to the source code: they depend upon their vendors to fix bugs. Even most users of so-called "open source" systems such as Linux and FreeBSD rely on others to fix bugs — there are simply too many flaws and not enough time. Even if we were to suggest changes, they might not be applicable t every platform of interest. Experience has shown that making changes often introduces new flaws unless the changes are extremely simple and well-understood.

There is also a problem associated with managing wide-scale changes. Not only can changes make th system more difficult to maintain, but changes can be impossible to manage across many architectures, locations, and configurations. They also will make vendor maintenance more difficult — how can vendors respond to bug reports for software that they didn't provide?

Last of all, we have seen programs and suggested fixes posted on the Internet that are incorrect or even dangerous. Many administrators of commercial and academic systems do not have the necessar expertise to evaluate the overall security impact of changes to their system's kernel, architecture, or

commands. If you routinely download and install third-party patches and programs to improve your system's security, your overall security may well be worse in the long term.

For all of these reasons, our emphasis is on using tools provided with your operating systems. Where there are exceptions to this rule, we will explain our reasoning.

## Third-Party Security Tools

There are many programs, systems, and other kinds of software tools that you can use to improve the security of your computer system. Many of these tools come not from your own organization or from the vendor, but instead from a third party. In recent years, third-party tools have been provided by corporations, universities, individuals, and even the computer underground.

When we published the first version of this book, there were precious few third-party security tools. Because the tools that did exist were important and provided functionality that was not otherwise available, we took an inclusive view and described every one that we thought significant. We tried th same approach when the second edition of this book was published and we suffered the consequences There were simply too many tools, and our printed descriptions soon were out of date.

With this third edition of *Practical Unix and Internet Security*, we have taken a fundamentally different approach. Today, tools are both being developed and being abandoned at such a furious rate that it is no longer practical to mention them all in a printed volume. Furthermore, many of the bette tools have been incorporated into the operating system. Therefore, in this edition of the book we will for the most part, discuss only tools that have been integrated into operating system distributions and releases. We will not devote time (and precious pages) to explaining how to download and install third-party tools or modifications.[5]

# Scope of This Book

This book is divided into six parts; it includes 26 chapters and 5 appendixes.

Part I, provides a basic introduction to computer security, the Unix operating system, and security policy. The chapters in this book are designed to be accessible to both users and administrators.

- Chapter 1, takes a very basic look at several basic questions: What is computer security? What i an operating system? What is a deployment environment? It also introduces basic terms we use throughout the book.
- Chapter 2, explores the history of the Unix operating system, and discusses the way that Unix history has affected Unix security.
- Chapter 3, examines the role of setting good policies to guide the protection of your systems. It also describes the trade-offs you will need to make to account for cost, risk, and corresponding benefits.

Part II, provides a basic introduction to Unix host security. The chapters in this part of the book are also designed to be accessible to both users and administrators.

- Chapter 4, is about Unix user accounts. It discusses the purpose of passwords, explains what makes good and bad passwords, and describes how the *crypt( )* password encryption system works.
- Chapter 5, describes how Unix groups can be used to control access to files and devices. It discusses the Unix superuser and the role that special users play. This chapter also introduces th Pluggable Authentication Module (PAM) system.
- Chapter 6, discusses the security provisions of the Unix filesystem and tells how to restrict acce to files and directories to the file's owner, to a group of people, or to everybody using the computer system.
- Chapter 7, discusses the role of encryption and message digests in protecting your security.
- Chapter 8. What if somebody gets frustrated by your super-secure system and decides to smash your computer with a sledgehammer? This chapter describes physical perils that face your computer and its data and discusses ways of protecting against them.
- Chapter 9, explores who you employ and how they fit into your overall security scheme.

Part III, describes the ways in which individual Unix computers communicate with one another and the outside world, and the ways in which these systems can be subverted by attackers who are trying break into your computer system. Because many attacks come from the outside, this part of the book is vital reading for anyone whose computer has outside connections.

- Chapter 10, describes how modems work and provides step-by-step instructions for testing your computer's modems to see if they harbor potential security problems.
- Chapter 11, provides background on how TCP/IP networking programs work and describes the security problems they pose.
- Chapter 12, the longest chapter in this book, explores the most common TCP and UDP services and how you can secure them.
- Chapter 13, one of the shortest chapters in the book, looks at the Remote Procedure Call system developed in the 1980s by Sun Microsystems. This RPC system is the basis of NFS and a numbe of other network-based services.
- Chapter 14, discusses services for authenticating individuals over a network: NIS, NIS+, Kerberos, and LDAP. It continues the discussion of the PAM system.
- Chapter 15, describes both Sun Microsystems' Network Filesystem (NFS) and the Windows-

compatible Server Message Block (SMB) — in particular, the Samba system.

- Chapter 16, describes common pitfalls you might encounter when writing your own software. It gives tips on how to write robust software that will resist attack from malicious users. This information is particularly important when developing network servers.

Part IV, is directed primarily towards Unix system administrators. It describes how to configure Unix on your computer to minimize the chances of a break-in, as well as to limit the opportunities for a nonprivileged user to gain superuser access.

- Chapter 17, discusses strategies for downloading security patches and keeping your operating system up to date.
- Chapter 18, discusses why and how to make archival backups of your storage. It includes discussions of backup strategies for different types of organizations.
- Chapter 19, describes ways that an attacker might try to initially break into your computer system. By finding these “doors” and closing them, you increase the security of your system.
- Chapter 20, discusses how to monitor your filesystem for unauthorized changes. This chapter includes coverage of the use of message digests and read-only disks, and the configuration and use of the Tripwire utility.
- Chapter 21, discusses the logging mechanisms that Unix provides to help you audit the usage an behavior of your system.

Part V, contains instructions for what to do if your computer’s security is compromised. This part of the book will also help system administrators protect their systems from authorized users who are misusing their privileges.

- Chapter 22, contains step-by-step directions to follow if you discover that an unauthorized perso is using your computer.
- Chapter 23, discusses approaches for handling computer worms, viruses, Trojan Horses, and other programmed threats.
- Chapter 24, describes ways that both authorized users and attackers can make your system inoperable. We also explore ways that you can find out who is doing what, and what to do about it.
- Chapter 25. Occasionally, the only thing you can do is sue or try to have your attackers thrown i jail. This chapter describes legal recourse you may have after a security breach and discusses wh legal approaches are often not helpful. It also covers some emerging concerns about running server sites connected to a wide area network such as the Internet.
- Chapter 26, makes the point that somewhere along the line, you need to trust a few things, and people. We hope you are trusting the right ones.

Part VI, contains a number of useful lists and references.

- Appendix A, contains a point-by-point list of many of the suggestions made in the text of the book.
- Appendix B, is a technical discussion of how the Unix system manages processes. It also describes some of the special attributes of processes, including the UID, GID, and SUID.
- Appendix C, lists books, articles, and magazines about computer security.
- Appendix D, is a brief listing of some significant security tools to use with Unix, including descriptions of where to find them on the Internet.
- Appendix E, contains the names, telephone numbers, and addresses of organizations that are devoted to ensuring that computers become more secure.

# Which Unix System?

An unfortunate side effect of Unix's popularity is that there are many different versions of Unix; today, nearly every computer manufacturer has its own. When we wrote the first edition of this book, there were two main families of Unix: AT&T System V and Berkeley's BSD. There was a sharp division between these systems. System V was largely favored by industry and government because o its status as a well-supported, "official" version of Unix. BSD, meanwhile, was largely favored by academic sites and developers because of its flexibility, scope, and additional features.

When we wrote the first edition of this book, only Unix operating systems sold by AT&T could be called "Unix" because of licensing restrictions. Other manufacturers adopted names such as SunOS (Sun Microsystems), Solaris (also Sun Microsystems), Xenix (Microsoft), HP-UX (Hewlett-Packard) A/UX (Apple), Dynix (Sequent), OSF/1 (Open Software Foundation), Linux (Linus Torvalds), Ultrix (Digital Equipment Corporation), and AIX (IBM) — to name a few. Practically every supplier of a Unix or Unix-like operating system made its own changes to the operating system. Some of these changes were small, while others were significant. Some of these changes had dramatic security implications and, unfortunately, many of these implications are usually not evident. Not every vendo considers the security implications of its changes before making them.

In recent years, Unix has undergone a rapid evolution. Most of the commercial versions of the operating system have died off, while there has simultaneously been an explosion of "free" Unix systems. Security has grown more important in recent years, and now all companies, organizations, and individuals distributing Unix claim to take the subject of security quite seriously. However, it is clear that some take the subject far more seriously than others.

## Versions Covered in This Book

The third edition of this book covers Unix security as it relates to the four most common versions of Unix today: Solaris, Linux, FreeBSD, and MacOS X. Solaris and Linux are generally thought of as System V-based operating systems, while FreeBSD and MacOS X are generally seen as BSD-based systems. However, there has been so much mingling of concepts and code in recent years that these distinctions may no longer be relevant. In many cases, the underlying theory and commands on these systems are similar enough that we can simply use the word "Unix" to stand for all of these systems. In cases where we cannot, we note individual operating system differences.

Particular details in this book concerning specific Unix commands, options, and side effects are base upon the authors' experience with AT&T System V Release 3.2 and 4.0, Berkeley Unix Release 4.3 and 4.4, Digital Unix, FreeBSD 3.0 through 4.5, Linux (various versions), MacOS X, NeXTSTEP 0.9 through 4.0, Solaris 2.3 through 8, SunOS 4.0 and 4.1, and Ultrix 4.0. We've also had the benefit of our technical reviewers' long experience with other systems, such as AIX and HP-UX. As these systems are representative of the majority of Unix machines in use, it is likely that these descriptions will suffice for most machines to which readers will have access.

**NOTE**

Throughout this book, we generally refer to System V Release 4 as SVR4. When we refer to SunOS without a version number, assume that we are referring to SunOS 4.1.x. When we refer to Solaris without a version number, assume that we are referring to Solaris 7 and above.

We also refer to operating systems that run on top of the Linux kernel as Linux, even though many Linux systems contain significant components that were developed by readily identifiable third parties. (For example, the Free Software Foundation was responsible for the creation of the GNU development tools, without which the Linux system could not have been built, while MIT and the X Windows Consortium were responsible for the creation and initial development of the X Window system.)

Many Unix vendors have modified the basic behavior of some of their system commands, and there are dozens upon dozens of Unix vendors. As a result, we don't attempt to describe every specific feature offered in every version issued by every manufacturer — that would only make the book longer, as well as more difficult to read. It would also make this book inaccurate, as some vendors change their systems frequently. Furthermore, we are reluctant to describe special-case features on systems we have not been able to test thoroughly ourselves. Whether you're a system administrator o an ordinary user, it's vital that you read the reference pages of your own particular Unix system to understand the differences between what is presented in this volume and the actual syntax of the commands that you're using. This is especially true in situations in which you depend upon the specific output or behavior of a program to verify or enhance the security of your system.

**THE MANY FACES OF "OPEN SOURCE" UNIX**

One of the difficulties in writing this book is that there are many, many versions of Unix. All of them have differences: some minor, some significant. Our problem, as you shall see, is that even apparently minor differences between two operating systems can lead to dramatic differences in overall security. Simply changing the protection settings on a single file can turn a secure operating system into an unsecure one.

The Linux operating system makes things even more complicated. That's because Linux is a moving target. There are many different distributions of Linux. Some have minor differences, such as the installation of a patch or two. Others are drastically different, with different kernels, different driver software, and radically different security models.

Furthermore, Linux is not the only free form of Unix. After the release of Berkeley 4.3, the Berkeley Computer Systems Research Group (CSRG) (and a team of volunteers across the Internet) worked to develop a system that was devoid of all AT&T code; this release was known as Berkeley 4.4. Somewhere along the line the project split into several factions, eventually producing four operating systems: BSD 4.4 Lite, NetBSD, FreeBSD, and OpenBSD. Today there are several versions of each of these operating systems. There are also systems based on the Mach kernel and systems that employ Unix-like utilities from a number of sources. (Chapter 2 covers this history.)

The world of free Unix is less of a maelstrom today than it was when the second edition of this book was published. However, it remains true that if you want to run Linux, NetBSD, FreeBSD, or any other such system securely, it is vitally important that you know exactly which version of which distribution of which operating system with which software you are running on your computer. *Merely reading your manual may not be enough!* You may have to read the source code. You may also have to verify that the source code you are reading actually compiles to produce the binaries you are running!

Also, please note that *we* cannot possibly describe (or even know) all the possible variations and implications, so don't assume tha we have covered all the nuances of your particular system. When in doubt, check it out.

By writing this book, we hope to provide information that will help users and system administrators improve the security of their systems. We have tried to ensure the accuracy and completeness of everything within this book. However, as we noted previously, we can't be sure that we have covered *everything*, and we can't know about all the quirks and modifications made to every version and installation of Unix-derived systems. Thus, we can't promise that your system security will never be compromised if you follow all our advice, but we can promise that successful attacks will be less likely. We encourage readers to tell us of significant differences between their own experiences and the examples presented in this book; those differences may be noted in future editions.

## “Secure” Versions of Unix

Over time, several vendors have developed “secure” versions of Unix, sometimes known as “trusted Unix.” These systems embody mechanisms, enhancements, and restraints described in various government standards documents. These enhanced versions of Unix are designed to work in Multilevel Security (MLS) and Compartmented-Mode Workstation (CMW) environments — where there are severe constraints designed to prevent the mixing of data and code with different security classifications, such as Secret and Top Secret. In 2001, Chris I. Dalton and Tse Huong Choo at HP Labs released a system called Trusted Linux. The National Security Agency has also released a Linux variant called Security Enhanced Linux (SE Linux).[6]

Secure Unix systems generally have extra features added to them, including access control lists, data labeling, enhanced auditing, and mutual authentication between separate components. They also remove some traditional features of Unix, such as the superuser’s special access privileges and acces to some device files. Despite these changes, the systems still bear a resemblance to standard Unix. Trusted Solaris still functions basically like Solaris.

These systems are not in widespread use outside of selected government agencies, their contractors, and the financial industry. It seems doubtful to us that they will ever enjoy widely popular acceptanc because many of the features make sense only within the context of a military security policy. On the other hand, some of these enhancements are useful in the commercial environment as well, and C2 security features are already common in many modern versions of Unix.

Today, trusted Unix systems are often more difficult to use in a wide variety of environments, more difficult to port programs to, and more expensive to obtain and maintain. Thus, we haven’t bothered t describe the quirks and special features of these systems in this book. If you have such a system, we recommend that you read the vendor documentation carefully and repeatedly.

# Conventions Used in This Book

The following conventions are used in this book:

Italic

Used for Unix file, directory, command, user, and group names. It is also used for URLs and to emphasize new terms and concepts when they are introduced.

`Constant Width`

Used for code examples, system output, and passwords.

*`Constant Width Italic`*

Used in examples for variable input or output (e.g., a filename).

**`Constant Width Bold`**

Used in examples for user input.

Strike-through

Used in examples to show input typed by the user that is not echoed by the computer. This is mainl used for passwords and passphrases that are typed.

call( )

Used to indicate a system call, in contrast to a command. In the original edition of the book, we referred to commands in the form *command(1)* and to calls in the form *call(2)* or *call(3)*, in which the number indicates the section of the Unix programmer's manual in which the command or call i described. Because different vendors now have diverged in their documentation section numbering we try to avoid this convention in this edition of the book. (Consult your own documentation index for the right section.) The *call( )* convention is helpful in differentiating, for example, between the *crypt* command and the *crypt( )* library function.

`%`

The Unix C shell prompt.

`$`

T he Unix Bourne shell or Korn shell prompt.

`#`

The Unix superuser prompt (Korn, Bourne, or C shell). We usually use this symbol for examples that should be executed by *root*.
Normally, we will use the Bourne or Korn shell in our examples unless we are showing something that is unique to the C shell.

`[ ]`

Surrounds optional values in a description of program syntax. (The brackets themselves should never be typed.)

Ctrl-X or ^X indicate the use of control characters. They mean "Hold down the Control key while typing the character `X'."

All command examples are followed by Return unless otherwise indicated.

**TIP**

This icon designates a note, which is an important aside to the nearby text.

**WARNING**

This icon designates a warning relating to the nearby text.

# Comments and Questions

We have tested and verified the information in this book to the best of our ability, but you may find that features have changed (or even that we have made mistakes!). Please let us know about any erro you find, as well as your suggestions for future editions, by writing to:


O’Reilly & Associates, Inc.

1005 Gravenstein Highway North

Sebastopol, CA 95472

(800) 998-9938 (U.S. and Canada)

(707) 827-7000 (international/local)

(707) 829-0104 (fax)


You can also contact O’Reilly by email. To be put on the mailing list or request a catalog, send a message to:

info@oreilly.com

We have a web page for this book, which lists errata, examples, and any additional information. You can access this page at:

http://www.oreilly.com/catalog/puis3/

To comment or ask technical questions about this book, send email to:

bookquestions@oreilly.com

For more information about O’Reilly books, conferences, Resource Centers, and the O’Reilly Network, see the O’Reilly web site at:

http://www.oreilly.com/

# Acknowledgments

We have many people to thank for their help on the various editions of this book. In the following sections, we've included the acknowledgments for previous editions as well as the current one.

## Third Edition

We would like to express our deepest thanks to the many people who worked with us in getting out th third version of this book. In particular, Paco Hope answered questions about the Unix "jail" now present on some versions, Casey Schaufler answered questions about POSIX 1003.1e; Ed Finkler helped with testing and expansion of the random number script in Chapter 16; students associated wi the MIT Student Information Processing Board answered questions on Kerberos, and Wietse Venema answered questions about TCP Wrappers.

Many individuals reviewed some or all of the chapters in this book and provided us with helpful feedback that made the book better than it otherwise would have been. In particular, we would like to express our thanks to Brian Carrier, Dorothy Curtis, Linda McCarthy, Clifford Neuman, Gregory D. Rosenberg (N9NNO), Danny Smith, Kevin Unrue, Wietse Venema, and Keith Watson. Special thanks to Gregg Rosenberg of Ricis, Inc., for the speed and thoroughness of his review of all chapters. Any errors that remain are ours alone.

Untold thanks go to Debby Russell, our editor at O'Reilly & Associates, without whom this book would not have happened.

## Second Edition

We are grateful to everyone who helped us develop the second edition of this book. The book, and th amount of work required to complete it, ended up being much larger than we originally envisioned. We started the rewrite of this book in January 1995; we finished it in March 1996, many months later than we had intended.

Our thanks to the people at Purdue University in the Computer Sciences Department and the COAST Laboratory who read and reviewed early drafts of this book: Mark Crosbie, Bryn Dole, Adam Hammer, Ivan Krsul, Steve Lodin, Dan Trinkle, and Keith A. Watson; Sam Wagstaff also commente on individual chapters.

Thanks to our technical reviewers: Fred Blonder (NASA), Brent Chapman (Great Circle Associates), Michele Crabb (NASA), James Ellis (CERT/CC), Dan Farmer (Sun), Eric Halil (AUSCERT), Doug Hosking (Systems Solutions Group), Tom Longstaff (CERT/CC), Danny Smith (AUSCERT), Jan Wortelboer (University of Amsterdam), David Waitzman (BBN), and Kevin Ziese (USAF). We woul also like to thank our product-specific reviewers, who carefully read the text to identify problems an add content applicable to particular Unix versions and products. They are C.S. Lin (HP), Carolyn Godfrey (HP), Casper Dik (Sun), Andreas Siegert (IBM/AIX), and Grant Taylor (Linux),

Several people reviewed particular chapters. Peter Salus reviewed the introductory chapter, Ed Ravin (NASA Goddard Institute for Space Studies) reviewed the UUCP chapter, Adam Stein and Matthew Howard (Cisco) reviewed the networking chapters, Lincoln Stein (MIT Whitehead Institute) reviewe the World Wide Web chapter, and Wietse Venema reviewed the chapter on wrappers.

Æleen Frisch, author of *Essential System Administration* (O'Reilly & Associates, 1995) kindly allowed us to excerpt the section on access control lists from her book.

Thanks to the many people from O'Reilly & Associates who turned our manuscript into a finished product. Debby Russell did another command performance in editing this book and coordinating the

review process. Mike Sierra and Norman Walsh provided invaluable assistance in moving *Practical Unix Security*'s original troff files into FrameMaker format and in managing an increasingly large and complex set of Frame and SGML tools. Nicole Gipson Arigo did a wonderful job as production manager for this book. Clairemarie Fisher O'Leary assisted with the production process and managed the work of contractors. Kismet McDonough-Chan performed a quality assurance review, and Cory Willing proofread the manuscript. Nancy Priest created our interior design, Chris Reilley developed the new figures, Edie Freedman redesigned the cover, and Seth Maislin gave us a wonderfully usable index.

Thanks to Gene's wife Kathy and daughter Elizabeth for tolerating continuing mentions of "The Book" and for many nights and weekends spent editing. Kathy also helped with the proofreading.

Between the first and second editions of this book, Simson was married to Elisabeth C. Rosenberg. Special thanks are due to her for understanding the amount of time that this project has taken.

## First Edition

The first edition of this book originally began as a suggestion by Victor Oppenheimer, Deborah Russell, and Tim O'Reilly at O'Reilly & Associates.

Our heartfelt thanks to those people who reviewed the manuscript of the first edition in depth: Matt Bishop (UC Davis); Bill Cheswick, Andrew Odlyzko, and Jim Reeds (AT&T Bell Labs) (thanks also to Andrew and to Brian LaMacchia for criticizing the section on network security in an earlier draft as well); Paul Clark (Trusted Information Systems); Tom Christiansen (Convex Computer Corporation); Brian Kantor (UC San Diego); Laurie Sefton (Apple); Daniel Trinkle (Purdue's Department of Computer Sciences); Beverly Ulbrich (Sun Microsystems); and Tim O'Reilly and Jerry Peek (O'Reilly & Associates). Thanks also to Chuck McManis and Hal Stern (Sun Microsystems), who reviewed the chapters on NFS and NIS. We are grateful for the comments by Assistant U.S. Attorney William Cook and by Mike Godwin (Electronic Frontier Foundation) who both reviewed the chapter on the law. Fnz Jntfgnss (Purdue) provided very helpful feedback on the chapter on encryption — gunaxf! Steve Bellovin (AT&T), Cliff Stoll (Smithsonian), Bill Cook, and Dan Farmer (CERT) all provided moral support and helpful comments. Thanks to Jan Wortelboer, Mike Sullivan, John Kinyon, Nelson Fernandez, Mark Eichin, Belden Menkus, and Mark Hanson for finding so many typos! Thanks as well to Barry Z. Shein (Software Tool and Die) for being such an icon and Unix historian. Steven Wadlow provided the pointer to Lazlo Hollyfeld. The quotations from Dennis Ritchie are from an interview with Simson Garfinkel that occurred during the summer of 1990.

Many people at O'Reilly & Associates helped with the production of the first edition of the book. Debby Russell edited the book. Rosanne Wagger and Kismet McDonough did the copyediting and production. Chris Reilley developed the figures. Edie Freedman designed the cover and the interior design. Ellie Cutler produced the index.

Special thanks to Kathy Heaphy, Gene Spafford's long-suffering and supportive wife, and to Georgia Conarroe, his secretary at Purdue University's Department of Computer Science, for their support while we wrote the first edition.

# A Note to Would-Be Attackers

We've tried to write this book in such a way that it can't be used easily as a "how-to" manual for potential system attackers. Don't buy this book if you are looking for hints on how to break into systems. If you think that breaking into systems is a fun pastime, consider applying your energy and creativity to solving some of the more pressing problems facing us all, rather than creating new problems for overworked computer users and administrators.

The days have long passed when breaking into computers required special skills or intelligence. Now all it requires is downloading scripts from a few web sites and clicking a button. What really takes cleverness is fixing systems so that they are resistant to external attacks. Breaking into someone else machine to demonstrate a security problem is nasty and destructive, even if all you do is "look around."

The names of systems and accounts in this book are for example purposes only. They are not meant t represent any particular machine or user. We explicitly state that there is no invitation for people to try to break into the authors' or publisher's computers or the systems mentioned in this text. Any suc attempts will be prosecuted to the fullest extent of the law, whenever possible. We realize that most o our readers would never even think of behaving this way, so our apologies to you for having to make this point.

[1] We do note, however, that the vast majority of viruses, worms, security flaws, and incidents tend to occur in non-Unix systems.

[2] http://www.gocsi.com/press/20020407.html

[3] This may mean the others had incidents too, but were unable to detect them or declined to report them.

[4] Remember to pass around the book itself or get another copy to share. If you were to make a photocopy of the pages to circulat it could be a significant violation of the copyright. This sets a bad example about respect for laws and rules, and conveys a message contrary to good security policy.

[5] Articles about current security tools, with detailed configuration information, appear regularly on the O'Reilly web site and the O'Reilly Network, as well as on a variety of security-related sites. In addition, see Appendix D for some suggestions.

[6] Security Enhanced Linux is a misleading name, however, as the release does not address all of the underlying architectural and implementation flaws. Instead, SE Linux adds a form of mandatory access control to a vanilla Linux. Assuming that there are no major bugs and that you configure it correctly, you can achieve better security — but it doesn't come automatically, nor does it provide a comprehensive security solution.