

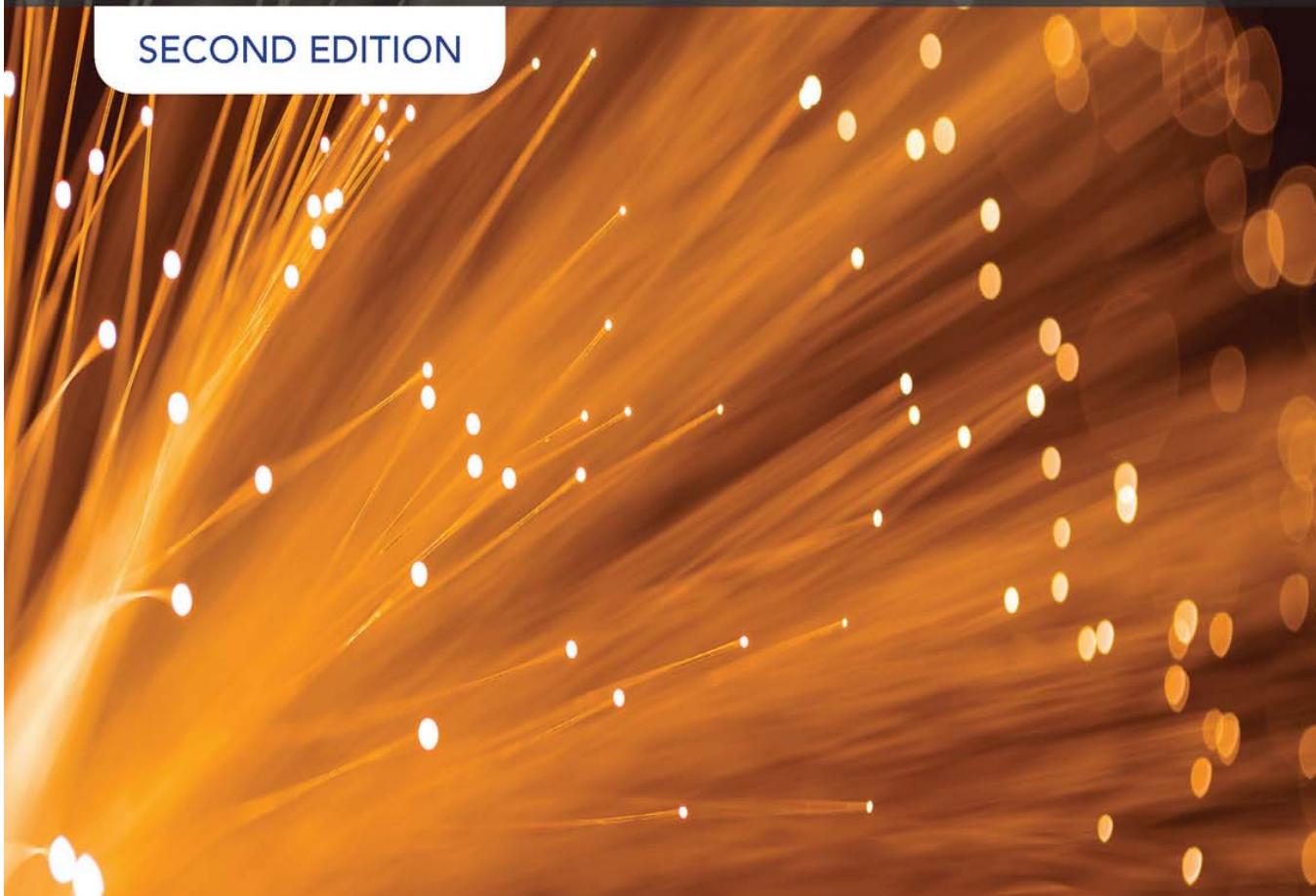
JONES & BARTLETT LEARNING

INFORMATION SYSTEMS SECURITY & ASSURANCE SERIES

Security Strategies in Linux Platforms and Applications

MICHAEL JANG AND RIC MESSIER

SECOND EDITION



JONES & BARTLETT LEARNING INFORMATION SYSTEMS SECURITY & ASSURANCE SERIES

Security Strategies in Linux Platforms and Applications

MICHAEL JANG AND RIC MESSIER

SECOND EDITION





World Headquarters

Jones & Bartlett Learning

5 Wall Street
Burlington, MA 01803
978-443-5000
info@jblearning.com
www.jblearning.com

Jones & Bartlett Learning books and products are available through most bookstores and online booksellers. To contact Jones & Bartlett Learning directly, call 800-832-0034, fax 978-443-8000, or visit our website, www.jblearning.com.

Substantial discounts on bulk quantities of Jones & Bartlett Learning publications are available to corporations, professional associations, and other qualified organizations. For details and specific discount information, contact the special sales department at Jones & Bartlett Learning via the above contact information or send an email to specialsales@jblearning.com.

Copyright © 2017 by Jones & Bartlett Learning, LLC, an Ascend Learning Company

All rights reserved. No part of the material protected by this copyright may be reproduced or utilized in any form, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without written permission from the copyright owner.

The content, statements, views, and opinions herein are the sole expression of the respective authors and not that of Jones & Bartlett Learning, LLC. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not constitute or imply its endorsement or recommendation by Jones & Bartlett Learning, LLC and such reference shall not be used for advertising or product endorsement purposes. All trademarks displayed are the trademarks of the parties noted herein. *Security Strategies in Linux Platforms and Applications, Second Edition* is an independent publication and has not been authorized, sponsored, or otherwise approved by the owners of the trademarks or service marks referenced in this product.

Microsoft, Internet Explorer, Windows, Microsoft Office, Microsoft Security Development Lifecycle, and Microsoft Baseline Security Analyzer are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. (ISC)², CISSP, ISSAP, ISSMP, ISSEP, CSSLP, CCFP, CAP, SSCP, and CBK are registered and service marks of (ISC)², Inc.

There may be images in this book that feature models; these models do not necessarily endorse, represent, or participate in the activities represented in the images. Any screenshots in this product are for educational and instructive purposes only. Any individuals and scenarios featured in the case studies throughout this product may be real or fictitious, but are used for instructional purposes only.

This publication is designed to provide accurate and authoritative information in regard to the Subject Matter covered. It is sold with the understanding that the publisher is not engaged in rendering legal, accounting, or other professional service. If legal advice or other expert assistance is required, the service of a competent professional person should be sought.

Production Credits

Chief Executive Officer: Ty Field

President: James Homer

Chief Product Officer: Eduardo Moura

SVP, Curriculum Solutions: Christopher Will

Director of Sales, Curriculum Solutions: Randi Roger

Editorial Management: High Stakes Writing, LLC, Lawrence J. Goodrich, President

Copy Editor, High Stakes Writing: Kate Shoup

Product Manager: Rainna Erikson

Product Management Assistant: Edward Hinman
Production Manager: Tina Chen
Associate Production Editor: Kristen Rogers
Senior Marketing Manager: Andrea DeFronzo
Manufacturing and Inventory Control Supervisor: Amy Bacus
Composition: Gamut+Hue, LLC
Cover Design: Scott Moden
Rights & Media Manager: Joanna Lundein
Rights & Media Research Coordinator: Mary Flatley
Cover Image: © leungchopan/Shutterstock
Printing and Binding: Edwards Brothers Malloy
Cover Printing: Edwards Brothers Malloy

ISBN: 978-1-284-09065-9

Library of Congress Cataloging-in-Publication Data

Jang, Michael H.

Security strategies in Linux platforms and applications / Michael Jang, Ric Messier. — Second edition.

pages cm

Includes bibliographical references and index.

ISBN 978-1-284-09065-9

1. Linux. 2. Operating systems (Computers) 3. Computer security. I. Messier, Ric. II. Title.

QA76.76.O63J385 2016

005.8—dc23

2015028823

6048

Printed in the United States of America

19 18 17 16 15 10 9 8 7 6 5 4 3 2 1

Contents

Preface

Acknowledgments

PART ONE

Is Linux Really Secure?

CHAPTER 1

Security Threats to Linux

The Origins of Linux

Security in an Open Source World

Linux Distributions

The C-I-A Triad

Linux as a Security Device

Linux in the Enterprise

Recent Security Issues

CHAPTER SUMMARY

KEY CONCEPTS AND TERMS

CHAPTER 1 ASSESSMENT

CHAPTER 2

Basic Components of Linux Security

Linux Security Relates to the Kernel

The Basic Linux Kernel Philosophy

Basic Linux Kernels

Distribution-Specific Linux Kernels

Custom Linux Kernels

Linux Kernel Security Options

Securing a System During the Boot Process

Physical Security

The Threat of the Live CD

Boot Process Security

More Boot Process Issues

Virtual Physical Security

Linux Security Issues Beyond the Basic Operating System

Service Process Security

Security Issues with the GUI

Linux User Authentication Databases
Protecting Files with Ownership, Permissions, and Access Controls
Firewalls and Mandatory Access Controls in a Layered Defense
 Firewall Support Options
 Mandatory Access Control Support
Protecting Networks Using Encrypted Communication
Tracking the Latest Linux Security Updates
 Linux Security Updates for Regular Users
 Linux Security Updates for Home Hobbyists
 Linux Security Updates for Power Users
 Security Updates for Linux Administrators
 Linux Security Update Administration
The Effect of Virtualization on Security
Variations Between Distributions
 A Basic Comparison: Red Hat and Ubuntu
 More Diversity in Services

CHAPTER SUMMARY

KEY CONCEPTS AND TERMS

CHAPTER 2 ASSESSMENT

PART TWO

Layered Security and Linux

CHAPTER 3

Starting Off: Getting Up and Running

Picking a Distribution
Picking a Delivery Platform
 Physical System
 Virtual Machines
 Cloud Services
Choosing a Boot Loader
 Linux Loader
 Grand Unified Boot Loader
Services
 Runlevels
 Wrappers
 inetd and xinetd
 R-services

CHAPTER SUMMARY

KEY CONCEPTS AND TERMS

CHAPTER 3 ASSESSMENT

CHAPTER 4

User Privileges and Permissions

The Shadow Password Suite

/etc/passwd

/etc/group

/etc/shadow

/etc/gshadow

Defaults for the Shadow Password Suite

Shadow Password Suite Commands

Available User Privileges

Securing Groups of Users

User Private Group Scheme

Create a Special Group

Configuring the Hierarchy of Administrative Privileges

Administrative Privileges in Services

The su and sg Commands

Options with sudo and /etc/sudoers

Regular and Special Permissions

The Set User ID Bit

The Set Group ID Bit

The Sticky Bit

Tracking Access Through Logs

Authorization Log Options

Authorization Log Files

Pluggable Authentication Modules

The Structure of a PAM Configuration File

PAM Configuration for Users

Authorizing Access with the Polkit

How the Polkit Works

Polkit Concepts

The Polkit and Local Authority

Network User Verification Tools

NIS If You Must

LDAP Shares Authentication

Best Practices: User Privileges and Permissions

CHAPTER SUMMARY

KEY CONCEPTS AND TERMS

CHAPTER 4 ASSESSMENT

CHAPTER 5

Filesystems, Volumes, and Encryption

Filesystem Organization

Filesystem Basics

The Filesystem Hierarchy Standard

Good Volume Organization Can Help Secure a System

Read-Only Mount Points

How Options for Journals, Formats, and File Sizes Affect Security

Partition Types

The Right Format Choice

Available Format Tools

Using Encryption

Encryption Tools

Encrypted Files

Encrypted Directories

Encrypted Partitions and Volumes

Local File and Folder Permissions

Basic File Ownership Concepts

Basic File-Permission Concepts

Changing File Permissions

Networked File and Folder Permissions

NFS Issues

Samba/CIFS Network Permissions

Network Permissions for the vsftpd Daemon

Configuring and Implementing Quotas on a Filesystem

The Quota Configuration Process

Quota Management

Quota Reports

How to Configure and Implement Access Control Lists on a Filesystem

Configure a Filesystem for ACLs

ACL Commands

Configure Files and Directories with ACLs

Best Practices: Filesystems, Volumes, and Encryption

CHAPTER SUMMARY

KEY CONCEPTS AND TERMS

CHAPTER 5 ASSESSMENT

CHAPTER 6

Securing Services

Starting a Hardened System

Service Management

SysV Init

Upstart

Systemd

Hardening Services

Using Mandatory Access Controls

Security Enhanced Linux

AppArmor

Servers Versus Desktops

Protecting Against Development Tools

CHAPTER SUMMARY

KEY CONCEPTS AND TERMS

CHAPTER 6 ASSESSMENT

CHAPTER 7

Networks, Firewalls, and More

Services on Every TCP/IP Port

Protocols and Numbers in /etc/services

Protection by the Protocol and Number

Obscurity and the Open Port Problem

Obscure Ports

Opening Obscure Open Ports

Obscurity by Other Means

Protect with TCP Wrapper

What Services Are TCP Wrapped?

Configure TCP Wrapper Protection

Packet-Filtering Firewalls

Basic Firewall Commands

Firewalld

A Firewall for the Demilitarized Zone

A Firewall for the Internal Network

Alternate Attack Vectors

Attacks Through Nonstandard Connections

Attacks on Scheduling Services

Wireless-Network Issues

Linux and Wireless Hardware
Encrypting Wireless Networks
Bluetooth Connections
Security Enhanced Linux
The Power of SELinux
Basic SELinux Configuration
Configuration from the Command Line
The SELinux Administration Tool
The SELinux Troubleshooter
SELinux Boolean Settings
Setting Up AppArmor Profiles
Basic AppArmor Configuration
AppArmor Configuration Files
AppArmor Profiles
AppArmor Access Modes
Sample AppArmor Profiles
AppArmor Configuration and Management Commands
An AppArmor Configuration Tool
Best Practices: Networks, Firewalls, and TCP/IP Communications

CHAPTER SUMMARY

KEY CONCEPTS AND TERMS

CHAPTER 7 ASSESSMENT

CHAPTER 8

Networked Filesystems and Remote Access

Basic Principles for Systems with Shared Networking Services
Configure an NTP Server
Install and Configure a Kerberos Server
Basic Kerberos Configuration
Additional Kerberos Configuration Options
Securing NFS as If It Were Local
Configure NFS Kerberos Tickets
Configure NFS Shares for Kerberos
Keeping vsftpd Very Secure
Configuration Options for vsftpd
Additional vsftpd Configuration Files
Linux as a More Secure Windows Server
Samba Global Options
Samba as a Primary Domain Controller
Making Sure SSH Stays Secure

The Secure Shell Server
The Secure Shell Client
Create a Secure Shell Passphrase
Basic Principles of Encryption on Networks
Host-to-Host IPSec on Red Hat
Host-to-Host IPSec on Ubuntu
Network-to-Network IPSec on Red Hat
Network-to-Network IPSec on Ubuntu
Helping Users Who Must Use Telnet
Persuade Users to Convert to SSH
Install More Secure Telnet Servers and Clients
Securing Modem Connections
The Basics of RADIUS
RADIUS Configuration Files
Moving Away from Cleartext Access
The Simple `rsync` Solution
E-mail Clients
Best Practices: Networked Filesystems and Remote Access

CHAPTER SUMMARY

KEY CONCEPTS AND TERMS

CHAPTER 8 ASSESSMENT

CHAPTER 9

Networked Application Security

Options for Secure Web Sites with Apache

The LAMP Stack
Apache Modules
Security-Related Apache Directives
Configure Protection on a Web Site
Configure a Secure Web site
Configure a Certificate Authority
`mod_security`

Working with Squid

Basic Squid Configuration
Security-Related Squid Directives
Limit Remote Access with Squid

Protecting DNS Services with BIND

The Basics of DNS on the Internet
DNS Network Configuration
Secure BIND Configuration

- A BIND Database
- DNS Targets to Protect
- Domain Name System Security Extensions
- Mail Transfer Agents
 - Open Source sendmail
 - The Postfix Alternative
 - Dovecot for POP and IMAP
 - More E-mail Services
- Using Asterisk
 - Basic Asterisk Configuration
 - Security Risks with Asterisk
- Limiting Printers
 - Printer Administrators
 - Shared Printers
 - Remote Administration
 - The CUPS Administrative Tool
- Protecting Time Services
- Obscuring Local and Network Services
- Best Practices: Networked Application Security

CHAPTER SUMMARY

KEY CONCEPTS AND TERMS

CHAPTER 9 ASSESSMENT

CHAPTER 10

Kernel Security Risk Mitigation

- Distribution-Specific Functional Kernels
 - Kernels by Architecture
 - Kernels for Different Functions
- The Stock Kernel
 - Kernel Numbering Systems
 - Production Releases and More
 - Download the Stock Kernel
 - Stock Kernel Patches and Upgrades
- Managing Security and Kernel Updates
 - Stock Kernel Security Issues
 - Distribution-Specific Kernel Security Issues
 - Installing an Updated Kernel
- Development Software for Custom Kernels
- Red Hat Kernel Development Software

Ubuntu Kernel Development Software
Kernel-Development Tools
 Before Customizing a Kernel
 Start the Kernel Customization Process
 Kernel Configuration Options
Building Your Own Secure Kernel
 Download Kernel Source Code
 Download Ubuntu Kernel Source Code
 Download Red Hat Kernel Source Code
 Install Required Development Tools
 Navigate to the Directory with the Source Code
 Compile a Kernel on Ubuntu Systems
 Compile a Kernel on Red Hat Systems
 Compile a Stock Kernel
 Install the New Kernel and More
 Check the Boot Loader
 Test the Result

Increasing Security Using Kernels and the /proc/ Filesystem

 Don't Reply to Broadcasts
 Protect from Bad ICMP Messages
 Protect from SYN Floods
 Activate Reverse Path Filtering
 Close Access to Routing Tables
 Avoid Source Routing
 Don't Pass Traffic Between Networks
 Log Spoofed, Source-Routed, and Redirected Packets

Best Practices: Kernel Security Risk Mitigation

CHAPTER SUMMARY

KEY CONCEPTS AND TERMS

CHAPTER 10 ASSESSMENT

PART THREE

Building a Layered Linux Security Strategy

CHAPTER 11

Managing Security Alerts and Updates

Keeping Up with Distribution Security

 Red Hat Alerts
 Red Hat Enterprise Linux
 CentOS Linux
 Fedora Core Linux
 Ubuntu Alerts

Keeping Up with Application Security

The OpenOffice.org Suite

Web Browsers

Adobe Applications

Service Applications

Antivirus Options for Linux Systems

The Clam AntiVirus System

AVG Antivirus

The Kaspersky Antivirus Alternative

SpamAssassin

Detecting Other Malware

Using Bug Reports

Ubuntu's Launchpad

Red Hat's Bugzilla

Application-Specific Bug Reports

Security in an Open Source World

The Institute for Security and Open Methodologies

The National Security Agency

The Free Software Foundation

User Procedures

Deciding Between Automated Updates or Analyzed Alerts

Do You Trust Your Distribution?

Do You Trust Application Developers?

Do You Trust Service Developers?

Linux Patch Management

Standard yum Updates

Updates on Fedora

Updates on Red Hat Enterprise Linux

Standard apt - * Updates

Options for Update Managers

Configuring Automated Updates

Automatic Red Hat Updates

Pushing or Pulling Updates

Local or Remote Repositories

Configuring a Local Repository

Commercial Update Managers

The Red Hat Network

Canonical Landscape

Novell's ZENworks

Open Source Update Managers

Various apt -* Commands
Various yum commands
Red Hat Spacewalk
Best Practices: Security Operations Management

CHAPTER SUMMARY

KEY CONCEPTS AND TERMS

CHAPTER 11 ASSESSMENT

CHAPTER 12

Building and Maintaining a Security Baseline

Configuring a Simple Baseline
A Minimal Red Hat Baseline
A Minimal Ubuntu Baseline
Read-Only or Live Bootable Operating Systems
Appropriate Read-Only Filesystems
Live CDs and DVDs
Keeping the Baseline Up to Date

A Gold Baseline
Baseline Backups

Monitoring Local Logs

The System and Kernel Log Services
Logs from Individual Services
Consolidating and Securing Remote Logs

Default rsyslog Configuration
The Standard rsyslog Configuration File

Identifying a Baseline System State

Collect a List of Packages
Compare Files, Permissions, and Ownership
Define the Baseline Network Configuration
Collect Runtime Information

Checking for Changes with Integrity Scanners

Tripwire
Advanced Intrusion Detection Environment

Best Practices: Building and Maintaining a Secure Baseline

CHAPTER SUMMARY

KEY CONCEPTS AND TERMS

CHAPTER 12 ASSESSMENT

CHAPTER 13

Testing and Reporting

Testing Every Component of a Layered Defense

- Testing a Firewall
- Testing Various Services
- Testing Passwords
- Testing Mandatory Access Control Systems

Checking for Open Network Ports

- The `telnet` Command
- The `netstat` Command
- The `lsof` Command
- The `nmap` Command

Running Integrity Checks of Installed Files and Executables

- Verifying a Package
- Performing a Tripwire Check
- Testing with the Advanced Intrusion Detection Environment

Ensuring that Security Does Not Prevent Legitimate Access

- Reasonable Password Policies
- Allowing Access from Legitimate Systems

Monitoring Virtualized Hardware

- Virtual Machine Hardware
- Virtual Machine Options
- Monitoring the Kernel-Based Virtual Machine (KVM)

Standard Open Source Security-Testing Tools

- Snort
- Netcat and the `nc` Command

Vulnerability Scanners for Linux

- Nessus
- OpenVAS
- Nexpose

Where to Install Security-Testing Tools

- Hint: Not Where Attackers Can Use Them Against You
- Some Tools Are Already Available on Live CDs

Best Practices: Testing and Reporting

CHAPTER SUMMARY

KEY CONCEPTS AND TERMS

CHAPTER 13 ASSESSMENT

CHAPTER 14

Detecting and Responding to Security Breaches

Performing Regular Performance Audits

The Basic Tools: `ps` and `top`
The System Status Package
For Additional Analysis

Making Sure Users Stay Within Secure Limits

- Appropriate Policies
- Education
- User Installation of Problematic Services

Logging Access into the Network

- Identifying Users Who Have Logged In
- System Authentication Logs

Monitoring Account Behavior for Security Issues

- Downloaded Packages and Source Code
- Executable Files

Creating an Incident Response Plan

- Increased Vigilance
- Should You Leave the System On?
- Acquiring the Memory Contents

Having Live Linux CDs Ready for Forensics Purposes

- Helix Live Response
- SANS Investigative Forensics Toolkit
- Digital Evidence and Forensics Toolkit
- Build Your Own Media
- Forensic Live Media

When You Put Your Plan into Action

- Confirming the Breach
- Identifying Compromised Systems
- Having Replacement Systems in Place

Secure Backup and Recovery Tools

- Disk Images for Later Investigation
- The `rsync` Command
- Mount Encrypted Filesystems

The Right Way to Save Compromised Data as Evidence

- Basic Principles for Evidence
- Remembering the Volatile Data
- Preserving the Hard Disks

Disaster Recovery from a Security Breach

- Determining What Happened
- Prevention
- Replacement

How and When to Share with the Open Source Community

If the Security Issue Is Known...

If the Security Issue Has Not Been Reported...

Best Practices: Security Breach Detection and Response

CHAPTER SUMMARY

KEY CONCEPTS AND TERMS

CHAPTER 14 ASSESSMENT

CHAPTER 15

Best Practices and Emerging Technologies

Maintaining a Gold Baseline

Monitoring Security Reports

Working Through Updates

Recalibrating System Integrity

Ensuring Availability with Redundancy

A Gold Physical Baseline

A Gold Virtual Baseline Host

Identifying Your Support Options

Red Hat Support Options

Canonical Support Options

Open Source Community Support

Checking Compliance with Security Policies

User Security

Administrator Security

Keeping the Linux Operating System Up to Date

Baseline Updates

Functional Bugs

New Releases

Keeping Distribution-Related Applications Up to Date

Server Applications

Desktop Applications

Managing Third-Party Applications

Licensing Issues

Support Issues

Sharing Problems and Solutions with the Community

Which Community?

Sharing with Developers

Sharing on Mailing Lists

Testing New Components Before Putting Them into Production

Testing Updates
Documenting Results
Beta Testing
Keeping Up with Security on Your Systems
A New Firewall Command
More Mandatory Access Controls
Penetration-Testing Tools
Single Sign-On
Incident Response

CHAPTER SUMMARY

KEY CONCEPTS AND TERMS

CHAPTER 15 ASSESSMENT

APPENDIX A

Answer Key

APPENDIX B

Standard Acronyms

Glossary of Key Terms

References

Index

*To my beautiful wife, Donna,
who has made life worth living again*
—Michael Jang

*To those who have made me who I am today:
Berkeley Breathed and Hunter S. Thompson*
—Ric Messier

Preface

Purpose of This Book

This book is part of the Information Systems Security & Assurance Series from Jones & Bartlett Learning (www.jblearning.com). Designed for courses and curriculums in IT Security, Cybersecurity, Information Assurance, and Information Systems Security, this series features a comprehensive, consistent treatment of the most current thinking and trends in this critical subject area. These titles deliver fundamental information-security principles packed with real-world applications and examples. Authored by professionals experienced in information systems security, they deliver comprehensive information on all aspects of information security. Reviewed word for word by leading technical experts in the field, these books are not just current, but forward-thinking—putting you in the position to solve the cybersecurity challenges not just of today, but of tomorrow as well.

Security Strategies in Linux Platforms and Applications, Second Edition, covers every major aspect of security on a Linux system. The first part of this book describes the risks, threats, and vulnerabilities associated with Linux as an operating system. Linux is one of the predominant operating systems used for public-facing servers on the Internet. As a result, a big focus for this book is on implementing strategies that you can use to protect your system implementations, even in cases where they are public facing. To that end, this book uses examples from two of the major distributions built for the server, Red Hat Enterprise Linux and Ubuntu (Server Edition).

With Linux, security is much more than just firewalls and permissions. [Part 2](#) of the book shows you how to take advantage of the layers of security available to Linux—user and group options, filesystems, and security options for important services, as well as the security modules associated with AppArmor and SELinux. It also covers encryption options where available.

The final part of this book explores the use of both open source and proprietary tools when building a layered security strategy for your Linux operating system environments. With these tools, you can define a system baseline, audit the system state, monitor system performance, test network vulnerabilities, detect security breaches, and more. You will also learn basic practices associated with security alerts and updates, which are just as important.

As with any operating system, a Linux implementation requires strategies to harden it against attack. Linux is based on another operating system with a very long history, and it inherits the lessons learned over that history as well as some of the challenges. With Linux, you get a lot of eyes looking at the programs, which many consider to be a benefit of using open source programs and operating systems. While there are advantages, however, there are risks associated as well. Fortunately, a large community is built around improving Linux and the various software packages that go into it. This includes the National Security Agency (NSA), which initially developed a set of security extensions that has since been implemented into the Linux kernel itself.

When you are finished with this book, you will understand the importance of custom firewalls, restrictions on key services, golden baseline systems, and custom local repositories. You will even understand how to customize and recompile the Linux kernel. You will be able to use open source and commercial tools to test the integrity of various systems on the network. The data you get from such tools will identify weaknesses and help you create more secure systems.

Learning Features

The writing style of this book is practical and conversational. Each chapter begins with a statement of learning objectives. Step-by-step examples of information security concepts and procedures are presented throughout the text. Illustrations are used both to clarify the material and to vary the presentation. The text is sprinkled with notes, tips, FYIs, warnings, and sidebars to alert the reader to additional helpful information related to the subject

under discussion. Chapter assessments appear at the end of each chapter, with solutions provided in the back of the book.

Throughout this book are references to commands and directives. They may be included in the body of a paragraph in a monospaced font, like this: `apt-get update`. Other commands or directives may be indented between paragraphs, like the directive shown here:

```
deb http://us.archive.ubuntu.com/ubuntu/ lucid main restricted
```

When a command is indented between paragraphs, it's meant to include a Linux command line prompt. You will note two different prompts in the book. The first prompt is represented with a `$`. As shown here, it represents the command-line prompt from a regular user account:

```
$ ls -ltr > list_of_files
```

The second prompt is represented by a `#`. As shown here, it represents the command-line prompt from a root administrative account:

```
# /usr/sbin/apachectl restart
```

Sometimes, the command or directive is so long, it has to be broken into multiple lines due to the formatting requirements of this book. Line wraps are indicated by a curved arrow, as is shown at the start of what looks like the second line of the `iptables` command. It is just a continuation arrow, which would be typed as a continuous command on the command line or an appropriate configuration file.

```
iptables -A RH-Firewall-1-INPUT -i eth0 -s 10.0.0.0/8  
→ -j LOG --log-prefix "Dropped private class A addresses".
```

Chapter summaries are included in the text to provide a rapid review of the material and to help students understand the relative importance of the concepts presented.

Audience

The material is suitable for undergraduate or graduate computer science majors or information science majors, students at a two-year technical college or community college who have a basic technical background, or readers who have a basic understanding of IT security and want to expand their knowledge. It assumes basic knowledge of Linux administration at the command-line interface.

Acknowledgments

I would like to thank Jones & Bartlett Learning and David Kim of Security Evolutions for the opportunity to write this book and be a part of the Information Systems Security & Assurance Series project. This book required a more substantial team effort than ordinary book projects. I would also like to thank the amazing project manager, Kim Lindros; the top-notch technical reviewer, Mike Chapple; the sharp copy editor, Kate Shoup; the marvelous compositor, Mia Saunders; the eagle-eyed proofreader, Ruth Walker; and Larry Goodrich along with Angela Silvia of High Stakes Writing for facilitating the entire process.

In addition, I acknowledge the gracious help of Billy Austin of the SAINT corporation, along with Mike Johnson of AccessData with respect to their products. The author also acknowledges the fortitude of Linux security professionals everywhere, white-hat hackers at heart who have to deal with cultural biases from the mainstream security community along with the legitimate fears of the open source community.

Most importantly, I could not do what I do without the help and support of my wife, Donna. She makes everything possible for me.

Michael Jang

Writing any book is a process. Revising an existing book for a second edition is also a process. It takes a team of people to get from conception to completion. Thanks to Mike, Kate, Mia, Larry, and everyone else who helped get this second edition to the goal line.

Mostly, I'd like to acknowledge all those people who jump into things without any idea what they are getting themselves into. This fearlessness is the best way to jump into something new and guarantee that you are going to learn a lot. Try it some time if you haven't already.

Ric Messier

About the Authors

MICHAEL JANG is a full-time writer, specializing in Linux and related certifications. His experience with computers dates back to the days of badly shuffled punch cards. He has written books such as *RHCE Red Hat Certified Engineer Study Guide*, *LPIC-1 In Depth*, *Ubuntu Server Administration*, and *Linux Annoyances for Geeks*. He is also the author of numerous video courses, and teaches preparation courses on Red Hat certification.

RIC MESSIER has been working with Unix and Unix-like operating systems since the mid-1980s. In the intervening decades, he has done system administration, network engineering, penetration testing, and programming; developed managed security services; and worked in operations security and a number of other jobs in between.

Ric is a security professional who has worked with a number of companies from large Internet service providers to small software companies. He has run a small networking and security consulting practice for the last several years. Additionally, he has taught courses at both the graduate and undergraduate level. Currently, in addition to writing books and recording training videos, he is the program director for Cyber Security and Digital Forensics at Champlain College in Burlington, Vt. He also maintains a blog on information security and digital forensics at securitykilroy.blogspot.com.

PART ONE**Is Linux Really Secure?****CHAPTER 1****Security Threats to Linux****CHAPTER 2****Basic Components of Linux Security**

CHAPTER

1 Security Threats to Linux

O

NE OF THE MOST SIGNIFICANT ATTACKS in the more than 40-year history of the Internet happened in the late 1980s. The overall numbers may not have been impressive, but when you look at it from a percentage perspective, it may have been the most devastating attack ever. In November of 1988, Robert T. Morris released a worm from a system located at the Massachusetts Institute of Technology (MIT), although he was at Cornell. The worm is estimated to have attacked 10 percent of the systems that were then connected to the Internet. The impact of the attack continued over several days while various networks that were attached to the NSFNet backbone were disconnected to get systems restored and patched. (The NSFNet was created by the National Science Foundation in the 1980s to pull together all the disparate specialized and regional networks. Initially, the links to the NSFNet were 56Kbps. The NSFNet took over where the ARPANET left off and became the launch point for what we now call the Internet.)

In addition to increasing the awareness of system vulnerabilities, the worm led to the creation of a Computer Emergency Response Team (CERT) at Carnegie Mellon. There were other actions taken to help coordinate activities in the case of another network-wide attack. Less than 15 years later, these coordination efforts were necessary when the first documented and large-scale distributed denial of service (DDoS) attack took place in February of 2000. A young man from Montreal, who called himself Mafiaboy, launched attacks against a number of prominent Web sites using the Stacheldraht tool in control of a botnet.

The significance of these two events is that both targeted system weaknesses on Unix-like operating systems. The worm attacked several Unix services that could easily be exploited by remote users. Some of these services were weak and unsecure to begin with, while others were simply a result of exploitable bugs in the software. The DDoS attacks were a result, in part, of the way the networking stacks within these operating systems were written. They were vulnerable to particular types of attacks that rendered the systems incapable of taking in more network requests. While some of this was because of the way the network protocols were designed, some was also a result of the implementation of these protocols within the operating system itself.

According to W3 Techs Web technology surveys, servers based on the Unix operating system make up two-thirds of the systems on the Internet. So the better we can understand how to provide security to these servers, the less vulnerable to attack they will be. Of course, the reality is that no operating system is secure. Security isn't a state. Security results from appropriate controls and processes, and can't be measured at a point in time. Understanding the appropriate technical means that need to be implemented is part of the process, but the state of a system constantly changes. This is due in no small part to influences from the outside. Among these is the so-called "research" constantly performed both by those who hope to improve the resilience of the system against attack and by those who wish to weaken that resilience.

Because the topic under consideration here is Linux, how does Unix factor into the equation? As it turns out, one must go back several decades to explain it.

Chapter 1 Topics

This chapter covers the following topics and concepts:

- What the origins of Linux are
- How security works in the open source world
- What distributions of Linux exist
- What the C-I-A triad is
- How Linux operates as a security device
- What Linux's place in the enterprise is
- What some examples of recent security issues are

Chapter 1 Goals

When you complete this chapter, you will be able to:

- Describe the basics of security in an open source world
- Explain various roles of Linux systems in the IT architecture
- Differentiate between Linux and the operating environment that runs on top of Linux
- Explain threats that can target Linux

The Origins of Linux

The **Linux** operating system is part of a very long and complicated family tree that began in the 1960s. In 1964, MIT joined with General Electric and Bell Labs to create a multiuser, time-sharing operating system. At the time, computers cost hundreds of thousands if not millions of dollars and were generally used only by a single user or process at a time. The goal of this project was to create an operating system that would allow multiple processes to run, seemingly simultaneously. The operating system was named Multics, short for Multiplexed Information and Computing Service, and its design reflected the concern about protecting one user from another user.

Two members of the Multics team from Bell Labs, Ken Thompson and Dennis Ritchie, became concerned that the system was becoming overly complex because of the design goals. In 1969, Bell Labs pulled out of the five-year-old project. When that happened, Thompson and Ritchie decided to develop their own operating system with goals almost entirely opposite to those of Multics. Thompson and Ritchie called their operating system Unics as a play on the name Multics. The “Uni” in Unics stood for *uniplexed*, suggesting that the goal was to create a small, workable operating system that didn’t have multiuser functionality as one of its aims. Where Thompson and Ritchie felt Multics was over-designed, they worked to create a system that was easier to use but still employed the hierarchical file system and the shell that were created for Multics. In general, though, where Multics favored large, complex user commands, Unics was developed with a lot of very small, single-purpose commands that could be chained together to create more complex functionality.

Fast forward 20 years or so and the name Unics had been changed to **Unix**, AT&T had been broken apart, and there were several versions of Unix available from AT&T, University of California at Berkeley, Microsoft, and others. By the mid-1980s, one of the advantages of Unix was that the source code was readily available and the design was simple enough that it made a good source of study for computer science students.

FYI

Linux is only the operating system, also called the **kernel**. This is what interfaces with the hardware to manage memory and file systems and make sure programs are run. Sun Microsystems used to make a distinction between its operating system, which it called SunOS, and the operating environment, which was Solaris. The operating environment is all of the programs and the user interface that communicates with the user and the kernel.

In 1987, a computer science professor and textbook author named Andrew Tanenbaum released a Unix-like operating system called MINIX in an appendix to an operating system textbook. The operating system was also

available on a set of floppy disks. Where Unix was primarily a large system operating system, MINIX was targeted at IBM PC-compatible systems. Not surprisingly, a large number of students began using the source code and talking about it on USENET. One of those students was Linus Torvalds, who began adding features and modifying what was in MINIX. Torvalds went on to release his version of the operating system, which he called Linux. Linux was first released on October 5, 1991.

FYI

Because the GNU project developed the common Unix utilities that users would employ if they were using a command-line shell, not to mention the compiler that is commonly used to build the Linux kernel, many GNU proponents prefer that the entire package be referred to as GNU/Linux. This has been the source of a number of long and heated debates over the years. In fact, it is sometimes referred to as a *religious war* because neither side is likely to convert the other side to their way of thinking. While it may be slightly inaccurate, the term *Linux* is generally used to describe the entire operating environment. Otherwise, you may have to start referring to a complete system as KDE/Apache/GNU/Linux or something equally unwieldy just to make sure all the different groups get appropriate billing.

Since its release, a number of open source projects have contributed to Linux. One of the most significant over the last 20 years has been **GNU's Not Unix (GNU)**, which was an attempt to create a Unix-like operating system.

Linux itself is a very fractured collective of different distributions. A **distribution** is a collection of a kernel, userland, graphical interface, and package-management system. The package-management system is used to install software. Package management is developed by the maintainers of the distribution, and there are many package-management systems available. RedHat (Fedora, RedHat Enterprise Linux, CentOS) uses RedHat Package Manager (RPM), though it also supports the Yellowdog Updater, Modified (Yum) that will check dependencies and download and install requested packages. Debian-based systems like Mint and Ubuntu use the Advanced Package Tool (APT) and the related utilities.

Security in an Open Source World

Linux is part of a large collection of software developed by teams that provide access to the source code and all the programming language text from which the final executable is generated for anyone who wants to look at it. This approach is called **open source** because the source code is open for anyone to see. Software developed by companies that ask you to pay for the program is commonly called *closed source* in addition to being commercial software.



NOTE

In a university setting, having access to source code enabled you to learn from what others were doing. If someone else had a better idea and improved what was there, then everyone could learn and benefit from it.

The idea behind open source goes back many decades to a time when programmers just wrote programs for the fun of it, leaving the source around for someone else to look at and improve. You didn't pay for software. The system software came with the machine you bought and the computer companies made their money on hardware.

The thing about programming is that everyone has a different style. Some people are far better at writing efficient code, while others are better at writing code that performs a lot of checks, making the resulting program more resistant to attack. Because of this, having access to source code means a couple of things:

- You can learn from the source code. You can see clearly what it does. You can have a better understanding of how the program operates.
- If you are so inclined, you can make fixes to the source in case there are bugs.

One of the leading proponents of open source software is Richard Stallman, who created the GNU project while he was at MIT. Stallman believed all source code should be available. This should not be read as a belief that everything should be without cost. There is a concept of *gratis versus libre*, commonly rendered in the open source community as “free as in free speech, not free as in free beer.” *Gratis* means without cost, as in free beer. *Libre* means without restriction, as in free speech. Stallman has long said that he believes that if he finds something that isn’t working right with a piece of software, he should be able to go into the source code and fix it. Without access to the source, he doesn’t have that freedom, which he believes is essential.

FYI

There are a number of well-known aphorisms that suggest that the more bugs you find, the more bugs there are. Proponents of open source suggest that making sure everyone has access to the code will lead to fewer bugs. More eyeballs means there are more people who can find issues and fix them. Open source detractors may counter that by saying not all open source developers are highly skilled and trained. This can lead to more bugs because the best programmers may not always contribute to well-used software projects.

What sorts of security issues are there with open source? First, just because the source code is open doesn’t mean the project has processes in place to ensure code is written securely. It also doesn’t mean there has been rigorous testing. If the only testing that has been done is by the developer, then there hasn’t been sufficient testing done on the source code. Developers have a different focus when they are testing code they have written than someone who is intent on doing complete security testing or even just regression testing. Larger projects have the advantage of ensuring that people who are competent at testing perform full testing before releases are issued. Smaller projects don’t have the luxury of dedicating a lot of people to testing, which can potentially put them at risk.



NOTE

Malicious code modifications have happened with open source projects in the past. One example was the ProFTP server that was hijacked in 2010. The source code available for download had been replaced with an altered copy that included a back door. Ironically, the person got in through an unpatched vulnerability in the FTP server software that was serving up the source code.

Open source projects put their source code out on the open Internet at public repositories. Along with the source code, there is generally a cryptographic hash generated to demonstrate that what you downloaded is what you are actually looking for. Where you are protected with this is if the download gets corrupted. It doesn’t protect you if the tarball has been altered unless the person doing the altering is really dumb. If the tarball gets modified, an attacker is going to generate a new MD5 and replace the existing one so everything looks correct. If an attacker can get access to the repository, he or she can upload modified source code that may include a back door or some other malicious modification.

Commercial software may suffer from the problem of lengthy processes that can get in the way of the speedy resolution of problems. A vulnerability must be logged, investigated, and then perhaps brought before a program or project manager for prioritization before the issue can be resolved. Once it’s been resolved, the fix likely has to wait for the next build schedule, at which point the entire build must be regression tested and unit tested to make sure all the issues have been resolved. A company like Microsoft batches all of its updates (unless they are considered critical) and releases them all at one time. An advantage to an open source project is that it may not

suffer from this process-heavy path to get a vulnerability fixed. This can be an enormous benefit, but it can sometimes be balanced by less documentation or less testing in a rush to get the fix out with an updated version. Open source projects, depending on their size, may be less concerned with release schedules and just issue a new minor version when a bug gets resolved. This isn't always the case, of course.

One of the key ideas behind open source projects like Linux (and all of the packages that go into a regular Linux distribution) is that you are less constrained by human resource issues. Anyone can pick up the source code and also pick up a bug and go fix it. This helps with speed to resolution, but it may not guarantee a high-quality fix. It also doesn't guarantee you will actually get contributors to your project.



NOTE

Nessus is a vulnerability scanner that began life as an open source project. However, the primary developers discovered they weren't getting a lot of help, so they closed the source and started a business selling Nessus.

One of biggest advantages to open source projects is the ability for anyone to start a project and have anyone else who is interested work on it. It doesn't require business plans and meetings to determine funding levels and returns on investment or marketing strategies. It just takes someone willing to get started. There are a lot of ways to post source code so someone else can take a look at it and make alterations. Most open source software is licensed in such a way that any changes are required to also remain open. This is another contribution of the GNU Project and its founder Richard Stallman in particular.



Stallman developed the GNU **General Public License (GPL)**. Stallman doesn't use the term *copyright* when talking about rights and privileges that are due software authors. Instead, he uses the term *copyleft*. Under copyleft and the GPL, any software that is based on GPLed software is required to retain the same rights as the original software. In other words, if I create a software project that I license using the GPL and you take my software, make some modifications to it, and want to release it yourself, you would also have to release it under the GPL.

Linux Distributions

There are a large number of Linux distributions, and their popularity waxes and wanes over time. Slackware, one of the early Linux distributions, retained tremendous popularity for a number of years. Now, however, it doesn't even register in the top 25 Linux distributions according to DistroWatch. At the time of this writing, it sits at number 33. Other distributions have fallen over time. **RedHat** was popular for a long time. Its level of support made it a top choice for a lot of users looking for Linux. Now, however, there is no so-called RedHat distribution. It has fragmented into RedHat Enterprise Linux, a piece of commercial software that you have to buy from RedHat, and Fedora, which is the development distribution for RedHat. **Fedor**a is more cutting-edge and is where RedHat tries out new concepts to get them stable before rolling them into RedHat Enterprise Linux.

Currently at the top of the popularity charts are **Mint** and **Ubuntu**, both derivatives of Debian. **Debian** has been around for a long time. It was created about 20 years ago by Ian Murdock, who wanted to pay homage to his girlfriend at the time, Debra. Debian is a merging of the name Ian with the name Debra. Debian has long been known in the Linux community as a very stable distribution of Linux. Often, it has been well behind what are considered current versions of packages because the maintainers were more interested in an operating system that was solid and didn't crash than they were with keeping up with the bleeding edge.



NOTE

Gentoo is named after a particular species of small, fast penguin.

Most distributions have pre-compiled packages. The distribution determines all the dependencies, meaning you might end up with a lot of extra packages that you don't really want because some package has dependencies built in from another package. One way to avoid this is to build your own version of Linux. Some distributions, such as Linux From Scratch and Gentoo Linux, are source-based distributions. The idea behind these sorts of distributions is that you decide how much or how little you want to put into it. This may have the upside of making it much faster and more responsive. However, the downside to these distributions is that all packages must be compiled from source, and compilation and installation can be a very time-consuming process. Getting one of these distributions up and running may take several hours or even the better part of a day, depending on the speed of your machine and your Internet connection. When you are finished, you will have exactly what you want, but every time you want to update, you will need to go through the compilation process again.



WARNING

Source-based distributions are not for the faint of heart, and are probably not best attempted by novice users.

The C-I-A Triad

When it comes to security, there are three fundamental concepts. You may sometimes hear these referred to as C-I-A, the C-I-A triad, the A-I-C triad (to distinguish it from the U.S. intelligence agency), or maybe just the triad. The three concepts are as follows:

- **Confidentiality**—Keeping secrets is the essence of **confidentiality**. If someone says something to you in a crowded room, you won't be assured of much in the way of confidentiality because it would be very easy for someone to overhear what the two of you are saying. If someone were to tap your phone and listen to your conversations, your confidentiality would be violated. This is certainly true when it comes to computer communications. If someone could listen in on network communications by port spanning on a switch, performing a spoofing attack, tapping the physical network cable, or some other method, your confidentiality would be violated. One common way to protect against this is to use encryption. This is not a flawless answer, of course, because not all encryption is created equal. Also, there are issues with keeping the encryption keys secret and protected. In general, however, if you are worried about confidentiality, you will want to find a way to ensure someone can't listen in on or intercept your conversations.
- **Integrity**—Ensuring that the data that is sent is the data that is received is what **integrity** is all about. It's also about protecting against corruption. The MD5 hash mentioned earlier that often accompanies software distributions is used to maintain the integrity of the data that is being transmitted. Note that integrity pertains to more than software downloads or messages that are emailed. It also relates to data at rest on disks. Any magnetic media like a hard drive or tape drive can be altered unexpectedly over a long period of time or from large electromagnetic pulses. Additionally, hardware sometimes fails, which might mean that data that is either written or read gets corrupted. Fortunately, you can use cyclical redundancy checks or cryptographic hashes as ways of ensuring that data has not been altered.
- **Availability**—If software is really buggy and crashes a lot, you will have **availability** issues. If you have integrity issues, as described in the preceding bullet, it might lead to availability issues. A denial of service (DoS) attack is a specific and malicious form of a denial of service condition. If a service fails for whatever reason, you will have a denial of service condition because normal users will be denied the use of the service. If the Ethernet card on a Web server suddenly fails, for instance, you will have a denial of service condition because the users who are trying to get to the Web server will be denied service. However, it's not malicious, so it's not an attack. It is definitely a problem of availability, however.



If an event falls into one of these buckets—confidentiality, integrity, or availability—it is considered to be a security issue.

Some security professionals don't consider the C-I-A triad to be sufficient to discuss all the problems in the world of security. As a result, you will sometimes hear the term *nonrepudiation* discussed separately because it doesn't fit nicely within the triad. Nonrepudiation is usually about preventing a party involved in a transaction from denying that the transaction occurred. One way this manifests is through cryptography. In public key cryptography, everyone has a private and a public key. Sometimes the private key is used to digitally sign a document, like an e-mail message. The private key is supposed to be protected and under the control of the owner of the key. If someone receives a message that was signed with that key, non-repudiation dictates that the private key owner can't say it didn't come from him or her. If the owner does, he or she is admitting that the key is no longer appropriately controlled.

In the late 1990s, Donn Parker, a security professional, proposed an expansion of the C-I-A triad to cover additional concepts he felt were critical to the realm of information security. This collection of ideas is called the *Parkerian hexad* and includes the following:

- Confidentiality
- Possession or control
- Integrity
- Authenticity
- Availability
- Utility

Although the Parkerian hexad is well-respected within the security community, it hasn't overtaken the C-I-A triad as the core and defining set of concepts with respect to information security.

Linux as a Security Device

If a Linux distribution makes a distinction between server and desktop/workstation, it has more to do with the number of packages that may be installed by default than it does with any differences in the look and behavior of the operating system once everything is loaded up. Typically, a server operating system comes without a user interface other than the command-line shell. This limits the number of packages that are installed, which makes the installed system less vulnerable to attack. A smaller number of packages means a smaller surface area for attack, which hopefully limits the potential for exploitation of the system. Limiting the number of software packages installed is a fundamental step to system hardening.

When you *harden* an operating system, you make it more resistant to attack. To do this, you remove any software you aren't going to be using. As a result, there are fewer ways in for an adversary. There are also fewer packages you have to monitor for updates in case vulnerabilities are found in the software. Hardening also involves removing all but the most critical users from the system. (Of course, this is always a good idea.) Making sure permissions are restricted on files and directories and removing all but the necessary system services are also common approaches to hardening the system.

After you have a completely hardened operating system, you may choose to deploy it as a bastion host. A **bastion host** is a system that is presented to the outside world that can be used to protect other systems that are more sensitive. Traditionally, a bastion host was a gateway system. For example, you might deploy a bastion host running a Secure Shell (SSH) server. An administrator might connect to the bastion host to gain administrative access to more sensitive network segments that shouldn't be exposed to the outside world,

whether that world was the Internet or just the desktop network within an enterprise. The bastion host was used in these situations as a security device. The definition of a bastion host varies, though, depending on whom you talk to. Some consider any Internet-facing device to be a bastion host.

Linux has long been used as a platform for developing security services, however. Certainly there are many Internet-facing services that have been developed on Unix-like operating systems like the Apache Web server, Sendmail, Postfix, various FTP servers, and a number of other software packages designed to provide specific services to users. Linux, though, has also been an area where specific security services have been developed and deployed. One of the first to come to mind is Snort. **Snort** is an intrusion-detection service that watches network traffic as it goes by, looking for areas of concern. Any packet or frame that passes by the Snort sensor that looks suspicious generates an alert.



NOTE

The word *bastion* simply means a fortified place. If you have a hardened operating system that has been completely locked down, you might say you have a bastion host.

Snort is not the only intrusion-detection service that's available under Linux, of course. A number of host-based intrusion-detection systems have been deployed under Linux. In fact, one of the first host-based intrusion-detection systems was developed for Unix-like operating systems. Tripwire, now a commercial product, was developed to monitor critical operating system files. If any of them changed, Tripwire generated an alert. Tripwire and other host-based intrusion-detection systems like AIDE, LIDS, and Samhain all run under Linux and provide similar functionality.

A Linux system may be deployed as a place to collect log files from across the network. The system logger (syslog) facility was developed a long time ago on Unix-like operating systems. There are a number of implementations of it available for Linux, including rsyslog and syslog-ng. When you deploy one of these packages and configure it to allow other systems to send logs to it, you have a centralized syslog server. There is a lot of value to a system like this. If all your logs are in one place, all you need is one program to monitor the logs to look for anomalies that should trigger an alert. There are a number of packages you can use, like Logwatch, that will monitor log files and generate an alert.

Of course, Linux also makes a good firewall. A **firewall** is hardware or software capable of blocking networking communications based on established criteria, or *rules*. Where Linux previously used a package called ipchains, the current firewall software available for Linux is called iptables. Iptables is a program that you use to create firewall rules that live within the kernel. The operating system—specifically, the network stack inside the operating system—keeps track of the rules and makes decisions about how to process network packages based on them. You can set up rules to allow, drop, or log specific packets from the network. If you have multiple network interfaces and a network behind your Linux firewall, you can have it perform network address translation (NAT), effectively hiding all the systems behind your Linux firewall.

You can make Linux into a network-based firewall or you can simply deploy a host-based firewall on your Apache Web server. This offers another layer of protection to the services that you offer (should you want to offer Web services to people who can connect to your system). On top of the firewall, you can also deploy encryption services using **Secure Sockets Layer (SSL)** or **Transport Layer Security (TLS)**. Both of these protocols can be deployed on a Linux-based system. In fact, you can use SSL/TLS or even IP Security (IPSec) on your Linux system as a gateway for a virtual private network (VPN). There are a few packages that will support this functionality under Linux.

You can see the wide range of security capabilities that a Linux system can support and its value in the security strategy for an enterprise. A small business may even make use of Linux servers at very little cost to help better secure their network because they can easily acquire a Linux distribution for free. This can be considerably less expensive than trying to purchase a number of appliances that will separately do all the things that a Linux system can do by itself. In fact, Linux will support a number of different systems installed on it using

virtualization software. You can end up with a number of virtual machines installed within a single Linux host. A *virtual machine* is an operating system that doesn't run directly on the user's hardware. You may have multiple virtual machines running on a system. This gives you a single system to manage from a physical perspective, but also enables you to logically separate your different functions onto different systems. An attacker may never know that the different systems being attacked are all resident on a single virtualized system.

Linux in the Enterprise

As mentioned, Unix-like operating systems in all of their forms, including Linux, comprise roughly two-thirds of all Web servers on the Internet. The Web server is one of just a couple of network services that are commonly exposed to outside systems from within an enterprise. The Web server is the way customers get access to information, products, and services offered by businesses. Additionally, many business-to-business services make use of Web services to function, so there is a lot of exposure from this one type of system. **Apache** is a common software package used to provide Web services, but it is by no means the only one. In fact, the Apache Foundation was created out of the efforts to develop a Web server to manage licensing but also as an umbrella organization to manage a number of other projects.

When it comes to Linux in the enterprise, it has a lot of additional purposes. One is as an application server. An **application server** is software that provides a framework inside which applications can be written. As an example, you may write a Java application that runs inside the application server and provides services to users over a network. The Apache Foundation is also responsible for developing Tomcat, which is a Java application server used to serve up Web services where all the business logic and functions are written in Java on the server side. JBoss is another application server designed for Java-based applications. You can think of this as analogous to the Internet Information Server (IIS) on the Windows side using .NET on the server to write application software. The interface for this software is the Web browser on the client's machine. Linux systems may also make good security systems. You may find Linux systems used to perform functions like serving as firewalls or intrusion-detection systems. In fact, Linux may be used as the operating system for a number of security-based appliances.



NOTE

If it isn't Linux, it may well be a Linux relation. Some firewalls have been known to use BSD as their operating system. BSD, of course, also has a Unix heritage and performs in a similar way to Linux, but the design of the kernel, the core of the operating system, is different. If you can operate a Linux system and understand how it works, you can easily pick up a BSD system.

While Linux isn't very common on the desktop, there are some users who use it. This often happens with system and network administrators because the tools that come built into most Linux distributions are helpful for those positions. Even if users commonly use Windows or even Mac OS X on their desktops, they may still run virtual machines that have Linux as a guest OS. This provides the functionality of the other desktop while still offering the benefits of a Linux system, without having to run a second system.

You may also use Linux as a proxy server or a gateway in the network. You may use it to perform some filtering between network segments or even use it as a jump host or bastion server. One advantage to Linux is that there are a lot of packages that support functions that are commonly found on Windows systems, so you could use a Linux server to perform as a Remote Desktop Protocol (RDP) system. You can also use it as a file server that makes use of Server Message Block (SMB) or Common Internet File System (CIFS) systems. Of course, you can also use more common Unix-based file sharing like the Network File System (NFS) to connect to either other Linux systems or other Unix-like systems.

Thanks to this versatility, Linux can perform a wide variety of functions within the enterprise. This versatility can also expose it to vulnerabilities, however. The more functions you add to a system, the more complexity you

create. The more complexity, the more potential for problems that you can't see. This is where testing is important before deployment. Even if you test extensively, however, there will be behaviors you won't see from everyday users accessing the system.

Recent Security Issues

Security is a hot topic in the news. While there have been serious attacks for years (if not decades), the mainstream news media is just now starting to pick up on what is happening and to report them. One reason for this is the prevalence of notification laws. Most states in the U.S. have laws that indicate that when a business has been breached by an attacker, the business must notify customers of the breach. This is because customers have the right to know when their personal information is at risk. Businesses commonly store a lot of information about their customers. In many cases, it can be personally identifiable information (PII), meaning that the information could be used to target a customer or use the information to create a new identity based on the customer's information. This new identity, which might be used to acquire credit and purchase goods and services or be sold to someone else, can do significant damage to the customer who is affected.

The media likes to refer to these criminals as *hackers*. The computing community, however, has long used the word *hacker* to describe someone who shows great technical skill. It may also refer to someone who shows some amount of creativity in pulling off a specific task. As an example, students at MIT refer to pranks around the campus as *hacks*. Many of these hacks are legendary, including placing a police car on the top of the Great Dome. These hacks have been going on for decades—long before computers became commonplace. It's because of this that those who were interested in computing at MIT began to refer to neat programming tricks or well-done programs as *hacks*. As a result, those who were engaged in the act of programming were often called *hackers*.

For a long time, to distinguish between the common technical prowess definition of hacker and those who were engaged in malicious or outright illegal actions, the term *cracker* was used. These days, however, neither term—hacker nor cracker—really describes what these people actually are. In most cases, they are simply criminals who happen to be using computers to perform their crimes. For this reason, the terms *attacker* or *adversary* will be used in this book. These terms are specific, and not sensationalistic. When someone is trying to get into a fortified system, he or she is an attacker. Someone who I am squaring off against is an adversary. They are clear words that describe the relationship of that person to the system or network you are attempting to protect.

In the security community, however, three expressions have become common:

- **Black-hat hacker**—The presence of a black hat often signals the bad guy in a Western movie. These types of characters would commonly wear a black hat so you could clearly identify the bad guy in the film. A **black-hat hacker**, then, is someone who performs attacks against victims for malicious purposes.
- **White-hat hacker**—In contrast, someone who wears a white hat is good. So a **white-hat hacker** may use some of the same techniques as a black-hat hacker, but he or she is a good guy.
- **Gray-hat hacker**—In most cases, there is a range of actions that a white-hat hacker will not undertake. Someone who will undertake those activities may be called a **gray-hat hacker**. This person is somewhere in between a white-hat hacker and a black-hat hacker. When you mix white and black, you get gray.

These are really shorthand terms to describe which side of a particular situation someone may be on. It's especially useful in penetration testing, when someone deliberately tries to get into a network or system. The white-hat hacker is on the inside while the gray-hat hacker is on the outside. A black-hat hacker wouldn't commonly be involved. He or she has no interest in finding holes for the purpose of fixing them, which is the intent of a penetration test.

There was a time when the most common sort of attacker was a young person who was either trying something out or trying to learn something new. It may also have been a result of so-called Internet Relay Chat (IRC) wars, where a common approach was to launch a denial of service (DoS) attack against someone who had said something annoying to you. Those days are gone. Today, the Internet is filled with determined and skilled

attackers who are looking for gain. This may be financial gain through the theft of personal information or credit cards. It may also be information. This could be intellectual property, but it could also be information that may be used by a government or quasi-government organization. You may see either corporate espionage or inter-country espionage. Of course, another sort of attack that you will commonly see is just a malware attack to get more zombie hosts for a botnet. This is also an attack for financial gain because the zombies may be running Web server software as a front to an illegal storefront for, say, pharmaceuticals. The botnet may also be used to rent to someone who just wants to attack someone else for political or financial gain.



CHAPTER SUMMARY

Linux has a long history as a standalone operating system as well as the heritage that comes from being an implementation of the concepts of the Unix operating system. Linux, though, is only the kernel, which is sometimes called the operating system. (Often, people refer to the operating environment as the operating system.) In the case of Linux, the operating environment brings in pieces from a number of other groups and packages, including the GNU Project and the Apache Foundation. Additionally, when you add in the graphical interface, you are talking about a handful of other groups. Because all of those components run on the foundation of the kernel, the entire system is commonly referred to as Linux, although some would consider that a misnomer.

Security is as important with Linux as it is with any other operating system. Linux is not immune to bugs or viruses. While it hasn't been as prone in recent years to attack, there have been both well-known attacks and viruses that have targeted the Linux system. In the end, any operating system that is poorly managed and configured can become vulnerable to attack. For this reason, security is commonly considered by security professionals to be a process and not a state. Security professionals would not consider a system to be either secure or unsecure. It may be vulnerable to exploit, but that's not the same as being unsecure.

Linux does have a lot of versatility, primarily because it has been open source since its inception. Having access to the source code and the ability to understand what is going on with it opens the door to a lot of development work on packages to extend its functionality. This extension takes us into the areas of security devices as well as the desktop and certainly as a server operating system. Linux remains the most popular choice for a Web server or at least for the device that sits in front of a Web server either as a load balancer, a reverse proxy, or a Web application firewall to protect Web applications from attack.



KEY CONCEPTS AND TERMS

Apache
Application server
Availability
Bastion host
Black-hat hacker
Confidentiality
Debian
Distribution
Fedora
Firewall
General Public License (GPL)

GNU's Not Unix (GNU)

Gray-hat hacker

Integrity

Kernel

Linux

Mint

Open source

RedHat

Secure Sockets Layer (SSL)

Snort

Transport Layer Security (TLS)

Ubuntu

Unix

White-hat hacker



CHAPTER 1 ASSESSMENT

1. Which of the following concepts is part of the C-I-A triad?
 - A. Authority
 - B. Access
 - C. Authenticity
 - D. Availability

2. Which of the following components makes up the core of the Linux operating system?
 - A. Cloned software from Unix
 - B. The kernel
 - C. Linux libraries
 - D. Linux services

3. Which of the following is an open source license?
 - A. Freeware
 - B. Public domain
 - C. GNU GPL
 - D. Any Microsoft license

4. From the following options, select a security advantage of open source software.
 - A. The efforts of the open source community
 - B. Secrecy in the source code
 - C. No information is released before a solution is available
 - D. None of the above

5. What percentage of Internet Web servers use Linux as their operating system?
 - A. 15 percent
 - B. 25 percent
 - C. 50 percent
 - D. 67 percent

6. Which of the following might be used to perform intrusion-detection services?



- A. Apache
 - B. X
 - C. Snort**
 - D. Mandrake
7. The open source license associated with the GNU project is .
8. Security is a .
9. Linux is based on which of the following?
- A. Debian
 - B. Ubuntu
 - C. Xinu
 - D. Unix**
10. What is the primary purpose of hardening an operating system?
- A. Creating a bastion host
 - B. Building a Web server
 - C. Protecting against attack**
 - D. Building a kernel
11. What is one risk associated with hand-building a kernel?
- A. Leaving your system unusable**
 - B. Creating a custom kernel
 - C. Violating the GPL
 - D. Not having the right tools
12. What does the acronym C-I-A stand for?
- A. Correctness, Intrusion, and Analysis
 - B. Confidentiality, Intrusion, and Availability
 - C. Confidentiality, Integrity, and Availability**
 - D. Clarity, Integrity, and Availability

CHAPTER

2 Basic Components of Linux Security

T

HIS CHAPTER COVERS BASIC COMPONENTS of Linux security. The starting point for securing a Linux system is the kernel. The kernel is not only the software that runs the system by interfacing between the software and the hardware, it's also the code that gets booted up when the system starts. In addition to the kernel and the boot process that helps to protect it, this chapter also looks at potential vulnerabilities associated with Linux applications. Linux security also depends on the authentication databases that apply to users and groups. User security forms the foundation for file ownership and access controls.

As security threats continue, it's important to keep up to speed with the latest Linux security updates. Updates can be a challenge if you administer a large number of systems, especially if they're virtual machines. Finally, if you administer systems based on different Linux distributions, you need to be aware of the differences between the major Linux distributions. A **Linux distribution** is a unified collection of applications, services, drivers, and libraries configured with a Linux kernel. Each Linux distribution is managed by either a company or a group of volunteers.

Chapter 2 Topics

This chapter covers the following topics and concepts:

- How Linux security relates to the kernel
- How you can better secure a system during the boot process
- Linux security issues that exist beyond the operating system
- Which user authentication databases are available for Linux
- How files can be protected with ownership, permissions, and access controls
- How to use firewalls and mandatory access controls in a layered defense
- How to use encrypted communications to protect Linux networks
- How you can track the latest Linux security updates
- The effect of virtualization on security
- How distributions vary

Chapter 2 Goals

When you complete this chapter, you will be able to:

- Describe security features, starting with the boot process
- Understand basic security options for users, groups, and files
- Work with basic security features for networks
- Review appropriate resources for the latest security updates

Linux Security Relates to the Kernel

This section reviews the basic philosophy behind the **Linux kernel** along with the types of kernels that are available. (The Linux kernel is the core of the Linux operating system. Different Linux kernels are in effect different operating systems. The Linux kernel includes a monolithic core and modular components.) To that end, you'll learn about the differences between the stock kernels released by Linux developers and how developers of a distribution modify them.

Frequently, the solution to a security issue is the installation of a new kernel. Although the differences between kernels are often minor, the work required to install a kernel is the same. If you choose to use a custom kernel, it may better serve your needs. However, it takes a lot more work to customize and compile a kernel. You also run the risk of ending up with a non-functional operating system if you aren't careful or you don't know what you are doing. The Linux kernel comes with a lot of components. If you leave a critical component out, the kernel will compile but you won't be able to boot the system. Furthermore, when a security issue requires a kernel **patch**, you'll have to repeat the process of customizing and recompiling a kernel. In the context of the Linux kernel, a patch is an incremental upgrade. It is a small piece of source code that replaces existing source code. It is used to fix bugs.

FYI

Whether to choose a **monolithic kernel** or a **modular kernel** is a matter of debate. If you combined all kernel modules into a monolithic kernel, it would load more quickly, without fragmentation. On the other hand, operating-system monolithic kernels may be huge, as they include drivers for every conceivable kind of hardware and lots of additional software. A monolithic kernel is one with all of the drivers and features built directly into a single kernel file. A modular kernel has drivers and features in separate kernel module files, apart from the kernel proper. Modular kernels are often easier to deal with from a development standpoint because the kernel can be loaded and unloaded as necessary.

The Basic Linux Kernel Philosophy

The Linux kernel is robust. One reason for this is its structure. It includes a monolithic core and modular components. The monolithic part of the kernel contains those drivers and options essential to the kernel boot process. The modular part of the kernel adds everything else needed for smooth operation, including many drivers that enable communication with installed hardware.

For example, most Linux sound cards are connected with drivers that are loaded with kernel modules if and as needed. If that kernel module fails, it does not affect other parts of the Linux operating system. You may just have to live without sound until you can load or reload a working driver. Mac OS X uses an even smaller kernel than Linux does. This is because Mac OS X is based on Mach, which was designed to be a very small and fast kernel. Windows is a mix of monolithic and modular, as Windows supports kernel mode modules, just as Linux does.

Basic Linux Kernels

Most Linux kernel developers are volunteers. Only a few are affiliated with a specific distribution such as Red Hat or Ubuntu. They follow open source principles. They are open about their development work. They share their progress online. Through the **Linux Kernel Organization** at <http://kernel.org>, they release production and developmental kernels. The Linux Kernel Organization is a nonprofit group established to distribute the Linux kernel and other open source software. Their stock production kernels are used for the core of every Linux distribution.

If you want to create a custom kernel, one place to start is the stock Linux kernel. You can customize that kernel using some of the changes made by the developers of a Linux distribution, without the need to appeal to a wide audience.

When security issues appear, Linux developers release changes to the kernel in the form of a patch. You can download that patch, apply it to the existing **source code**, compile the combined source code, and then update your system. Source code consists of human-readable computer language that can be collected and compiled into a computer program, library, or application.

While the developers behind various Linux distributions use the same patches, they test their changes on all their target systems. There's often a lag between the release of a security patch and the release of an updated kernel customized for a distribution. On the other hand, developers compile the resulting kernel for you. All you need to do is install it on your system.

Distribution-Specific Linux Kernels

For most users, it's best to stick with the **binary kernel** released for a Linux distribution. That distribution-specific kernel has been compiled and built with the intended hardware in mind, and in many cases has been labeled as such. This is especially true when it comes to 64-bit versus 32-bit hardware. The name of the kernel file will generally include the version number as well as the hardware architecture. For example, one of the latest releases of Debian Linux has a kernel named

vmlinuz-3.2.0-4-amd64. The last part, amd64, indicates the architecture. Even if the kernel is not so labeled, it may be optimized for certain purposes—for example, high availability or database support.

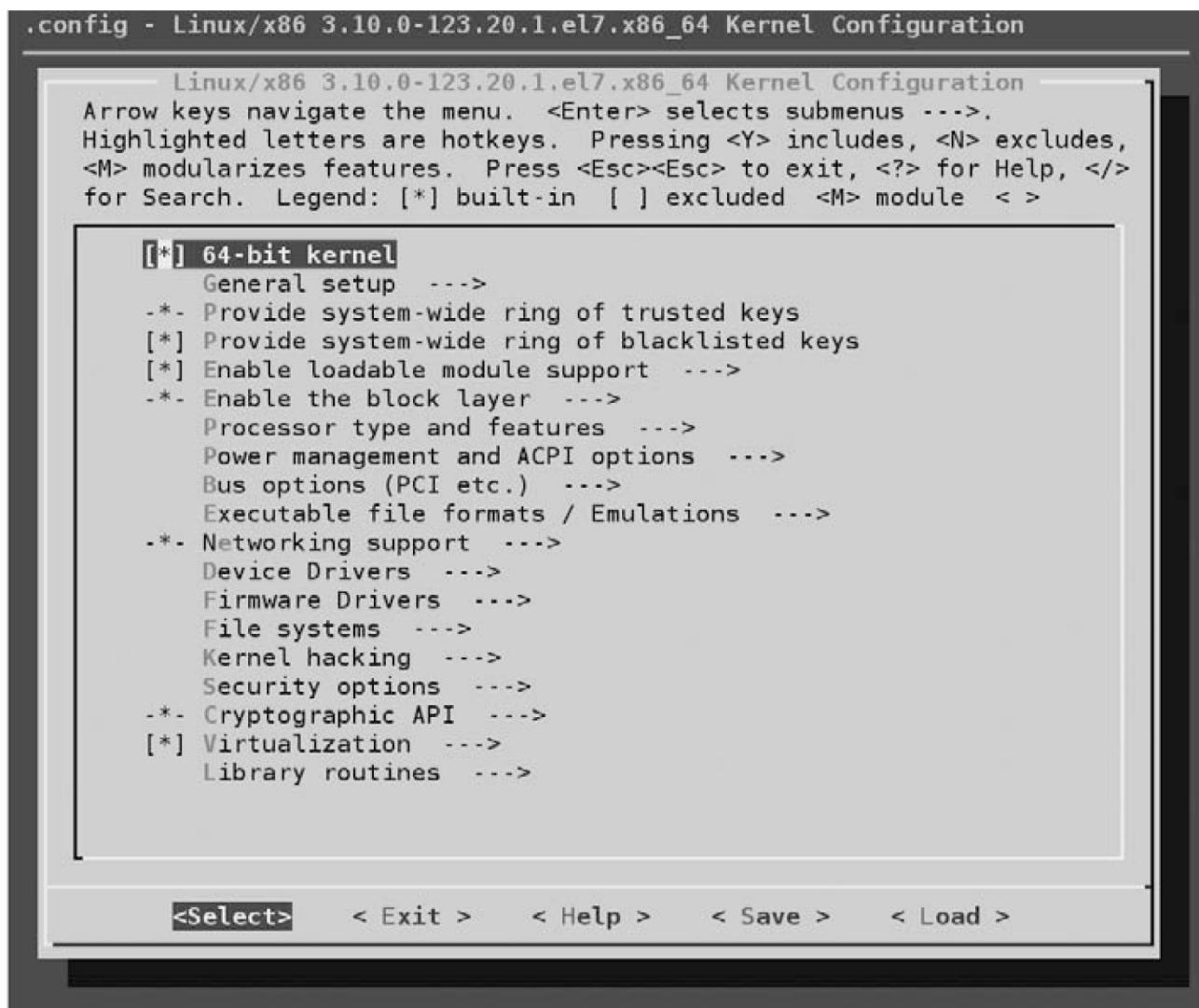
When security issues appear with a Linux kernel, most popular Linux distributions release updates fairly quickly. They've done the work to compile the kernel for you. Their compiled kernel packages will even automatically update standard Linux boot loaders. All you need to do is download and install the updated kernel.

Custom Linux Kernels

If you choose to customize a Linux kernel, you'll need to get into its nuts and bolts. The Linux kernel is highly customizable. The 3.10 kernel line, included with Red Hat Enterprise Linux 7, has more than 5,000 configuration options. If you want to preview some of these options, look for a config-* file in the /boot directory of a Linux system. It's a text file. These options are divided into a number of categories. One standard customization menu with some of these categories is shown in [Figure 2-1](#).

[Table 2-1](#) provides an overview of the categories of options associated with the Linux kernel. As kernel development continues, these categories may change.

Take some time to understand the options associated with the Linux kernel. It's worth the trouble. Even if you never recompile a kernel, that knowledge will help you understand how the kernel can help maintain Linux security. In addition, some of those options can be changed in real time, with the help of options in the /sys/ directory and using the sysctl utility as well as the /etc/sysctl.conf configuration file.

**FIGURE 2-1**

A Linux kernel configuration menu.

If you choose to recompile a kernel, you'll need to download the source code. In general, it's best to start from one of two different source-code trees. You can start with the stock kernel source code released by the developers of the Linux kernel. Alternatively, you can start with the source code released by a distribution. As the source code for all Linux kernels are freely available under open source licenses, the choice is yours.

Linux administrators who want the most secure kernel will remove or disable many features. If used by a malicious user, some of these features may lead to security risks. But take care. Test any changes that you make. With thousands of options, some changes could make that kernel unusable.

WARNING

The programming libraries required to compile the Linux kernel can present security risks. If a malicious user gets access to that system, that person can run those compilers to build more dangerous malware. If you choose to compile a custom kernel, you need to either compile that kernel on a system isolated for that purpose or uninstall those libraries after the compile is complete.

TABLE 2-1 Basic configuration categories for the Linux kernel.

CATEGORY	DESCRIPTION
----------	-------------

General setup	Includes a variety of basic kernel settings
Loadable module support	Sets conditions for how modules are loaded for a modular kernel
Block layer	Defines how block devices can be loaded
Processor type and features	Configures support for a variety of processor types and memory levels
Power management options	Defines available power-management features
Bus options	Specifies support for hardware cards
Executable file formats/ emulations	Associated with executable and linkable format binaries
Networking	Includes network drivers, protocols, and more
Device drivers	Support for a wide variety of hardware
Firmware drivers	Supports nonvolatile systems such as the Basic Input/Output System (BIOS) and the Unified Extensible Firmware Interface (UEFI)
Filesystems	Includes various filesystem formats
Instrumentation support	Adds options for performance modules
Kernel hacking	Designed for kernel debugging
Security options	Includes options such as Security Enhanced Linux (SELinux) and Application Armor (AppArmor)
Cryptographic API	Defines supported encryption algorithms for application programming interfaces (APIs)
Virtualization	Adds code for hardware-based virtual machines
Library routines	Includes modules for cyclic redundancy checks

Linux Kernel Security Options

Linux kernel security options range far and wide. Direct options may enable features such as AppArmor, SELinux, and **iptables**-based firewalls (**iptables** is the Linux packet filtering command for firewalls and masquerading). Indirect options may make it easier for malicious users to break into your systems. Three basic kernel security options, as listed in the Linux Security HOWTO, include the following:

- `CONFIG_FIREWALL` for network firewalls
- `CONFIG_IP_MASQUERADE` for IP address masquerading to help protect systems on a LAN
- `IP_FORWARDING` to set up a system as a gateway

Securing a System During the Boot Process

If a malicious user has physical access to a system, that user can boot a fully functional version of Linux on your servers from a **live CD**—with full root administrative access. Even without a live CD, some distributions allow a user to boot Linux in single-user mode with full administrative privileges—without a password. (A live CD is a CD or DVD with a bootable operating system. That same data may also be loaded on a USB drive.)

This chapter will assume that the boot process ends once the Linux kernel is loaded on the system. As you'll see later in this chapter, the startup process continues by loading preconfigured services.

Physical Security

Physical security goes beyond locks in the server room. It also involves security for CD/DVD drives and universal serial bus (USB) ports. While a live CD may in some cases be an excellent way to recover from problems on a server or workstation, access to such systems should be limited. Live CDs provide password-free access to the administrative account. Therefore, a malicious user with a live CD may be able to do anything that an administrator can do to your system.

The Threat of the Live CD

It's not enough to set up secure usernames and passwords on your systems. A malicious user with physical access may be able to boot a live CD Linux distribution such as **Knoppix**, Ubuntu, or even **CentOS** directly on your servers. (Knoppix is a Linux distribution best known for its live CDs and DVDs. Short for the Community Enterprise Operating System, CentOS is sometimes known as a rebuild because it is a distribution built by third parties, based on source code released for the Red Hat Enterprise Linux distribution.) It doesn't matter if the Linux distribution on the live CD differs from that on the hard drive. It also doesn't matter if the Linux kernel on the live CD has a different version number. Many live CDs even provide a graphical user interface (GUI), making it relatively easy for even less-experienced users, regardless of their motive, to break into your system.

All that's needed is boot access from a CD/DVD drive or a USB port. Once booted, a live CD provides password-free administrative access to any system. At that point, it's easy to access hard drives on the local system.

Boot Process Security

Anyone with physical access to your systems may be able to access boot menus, such as those available through a BIOS or a UEFI menu. Such menus should be password-protected. Network access to the UEFI menu should be disabled. If possible, these menus should be used to exclude all options but the hard drive with the operating system from the boot process.

Boot process security extends to the boot menu for the operating system. Linux includes two options for boot loaders: the **Grand Unified Bootloader (GRUB)** and the **Linux Loader (LILO)**. (Boot loaders like GRUB and LILO are responsible for locating the kernel and loading it into memory so it can run. GRUB is the default boot loader for Ubuntu, Red Hat, and many other Linux distributions, while LILO is an alternative boot loader.) Both boot loaders include their own boot menus. Both boot menus can be used to start other operating systems, such as other versions of Linux or even Microsoft Windows or one of the BSD variants. Both boot loaders also include a number of security risks.

In fact, unless password protected, both boot loaders can be used to boot a system into single-user mode, with full administrative privileges. On most Linux distributions, an administrative password isn't required to access this mode.

Other boot loaders can be installed on most systems, including Microsoft options such as NTLDL and bootmgr, as well as third-party options such as BootIt and Partition Magic. Each of these boot loaders can be used to start a Linux operating system.

More Boot Process Issues

The Transmission Control Protocol/Internet Protocol (TCP/IP) suite includes default port numbers for hundreds of services, such as port 22 for Secure Shell (SSH) and port 23 for Telnet. Some security professionals prefer to use nonstandard port numbers for key services. That obscurity can slow the efforts of attackers who want to break into a system.

Some services require network access during the system startup process. For example, if you keep servers synchronized with the **Network Time Protocol (NTP)**, those servers may need to access external NTP servers during the boot process. NTP is a protocol and service for synchronizing clocks across systems on a network. The standard NTP port is 123. If you want to block that port after the boot process is complete, you need to make sure the firewall isn't started until after the NTP client is synchronized. At that point, if you introduce such a firewall rule, you will no longer have the ability to sync with the NTP servers as long as the system is running and the firewall rule is in place.

Virtual Physical Security

Virtual machines add another dimension to the challenges of physical security. Standard Linux user and group options add security to virtual machines. However, anyone who gains access to a virtual machine can more easily change virtual physical components. For example, a malicious user who adds a live CD file to a virtual CD drive would arouse less suspicion than a malicious user who unlocks a CD/DVD drive on an actual physical system. Where physical systems run the risk of a similar situation, virtual machines are even more at risk. If an adversary gets access to the host system, he or she can make changes to any virtual machine on that host. Where physical systems require physical access to alter the

hardware, virtual machines can be modified through changes to configuration files, and those changes can be made remotely.

Linux Security Issues Beyond the Basic Operating System

While issues have been found within the Linux kernel, the majority of security vulnerabilities within Linux distributions are found within the applications and services that run on top of the kernel.

One of the main advantages of Linux is its structure as a multi-user operating system. When properly configured, Linux services run on nonprivileged accounts. If a malicious user breaks into one service, that user may gain access to the corresponding non-privileged account. However, that user won't have access to other accounts.

A GUI is also beyond the basic Linux operating system. If included, a GUI introduces a whole raft of additional security risks, discussed shortly.

Service Process Security

For the purposes of this chapter, the boot process ends when the Linux kernel is loaded. The startup process then begins automatically, loading and starting a series of services configured in the **default runlevel**. A **runlevel** defines the services to be run. The default runlevel defines the services to be run after a system is booted up. Most distributions try to keep the number of default services to a minimum, but a couple that you might expect to see are the SSH daemon, sshd—allowing remote administrative connections—and syslog, which is the system logging service.

Red Hat Enterprise Linux includes a number of services that are active by default. Most users do not need many of these services. [Table 2-2](#) lists services that were active after installation of a fairly minimal 64-bit server system. The services in [Table 2-2](#) are listed in lowercase because that's how they appear in the /etc/init.d/ directory.



NOTE

You specify a default runlevel that will be selected each time the system starts without intervention by someone when the system is booting up. The runlevel determines which services will start during the init process, which is the super process that is responsible for starting all other processes during system startup.

TABLE 2-2 Typical default active services.

SERVICE	DESCRIPTION
cups	Common Unix Printing System (CUPS) service
microcode_ctl	Microcode utility for Intel 32-bit processors
ntpd	Network Time Protocol (NTP) daemon
sshd	Secure Shell (SSH) service
syslog	System log message service

If any service is not maintained, related security flaws may not get fixed. This may be a security flaw that has been located and an exploit developed by an adversary. The malicious user is free to use that exploit as long as the system remains unpatched. Any other attacker can also use exploits that may be found in the wild to attack your system. This is why it's essential to have a patch-management and update process.

Other services that may be active on default Linux installations include the Apache Web server, the Samba file server, the Network File System service, the **sendmail** e-mail service, and the Very Secure FTP server. (The sendmail e-mail service is the open source SMTP server maintained by the Sendmail Consortium. Do not confuse this with the commercial SMTP server known as **Sendmail**.) Because security issues on some of these services are relatively common, administrators should disable these services if they're not being used. In fact, any service that isn't being used should be disabled to limit the attack surface exposed to adversaries.



NOTE

The Very Secure FTP server is a GPL version 2-licensed **File Transfer Protocol (FTP)** server for Unix and Linux. FTP is a standard protocol and service for exchanging files.

Security Issues with the GUI

Applications that require a GUI are common security risks. Although malware written for Windows GUI applications may not affect Linux GUI applications, similar risks do apply. Furthermore, the GUI itself is a security risk. On Linux, the GUI is a networkable client-server system. Users can log into a GUI remotely. Users can also run GUI clients over a network. In fact, administrators can configure a system to display GUI clients on a remote terminal. So malware on one Linux GUI application can be spread across a network to other GUI systems.

For these reasons, Linux security experts encourage administrators to avoid installing the GUI unless absolutely necessary. In the Linux world, a system without a GUI is no handicap. Linux administrators are accustomed to configuring services from the command-line interface.



One Linux server access control program, xhost, supports cleartext authentication. Like Telnet, xhost makes it easy for malicious users to read any information sent over that connection, including usernames and passwords. The SSH service can solve both problems, as it supports encrypted logins as well as networking of individual GUI clients. If you are using SSH, though, ensure that version 1 is disabled because it has vulnerabilities that can lead to the encryption being broken.

Linux User Authentication Databases

There are four major user authentication databases available for Linux. The local Linux authentication database is known as the shadow password suite. The files of the shadow password suite are /etc/passwd, /etc/group, /etc/shadow, and /etc/gshadow. These files include encrypted passwords, user and group accounts, home directories, and login shells.

The other three user authentication databases are designed as a central authentication database for multiple systems on a network. When properly configured, a user can log into any connected workstation. That workstation checks login credentials from that central database. Two of these databases are native to Linux: the Network Information Service (NIS) and the Lightweight Directory Access Protocol (LDAP). With the help of the **Winbind** service, Linux can also serve as the domain controller repository for some authentication databases commonly used on Microsoft-based networks. Winbind is a component of the Samba file server that supports the integration of Linux/Unix and Microsoft authentication information.



NOTE

Since Samba version 4, you can use your Linux system as a Windows domain controller to authenticate Windows users. Samba can also be configured to authenticate to a Windows Active Directory server.

In all these cases, the user and the group get both a user ID and a group ID number. With those ID numbers, each user (and group) can own files on the local Linux system. Each networked user can be configured as a member of other special groups with administrative privileges, just like any other Linux user.

The advantage of NIS is that it can start with the files of the shadow password suite. NIS has two disadvantages, however: It is relatively unsecure and it does not support authentication of Microsoft users or groups. In contrast, while LDAP is a more complex authentication scheme, it does support authentication of both Linux/ Unix and Microsoft users and groups.

In a Linux authentication database on a desktop system, a user is typically configured as a member of a group with the same name. For example, if a Linux system has a user named mike, that user is a member of a group named mike. That Linux user mike may be a member of other groups. For example, one group may have print-administration privileges; another group may have access to telephone modems.

To help manage this variety of authentication databases, Linux uses the `/etc/nsswitch.conf` file, also known as the name service switch file. That file lists databases associated with a number of different configuration files in the `/etc/` directory, in some specific order. That order determines where the local system looks first to verify a username and password. Red Hat includes a descriptive configuration tool for all these databases and more, shown in [Figure 2-2](#).

Linux implements **pluggable authentication modules (PAMs)** to determine how a user is to be authenticated and whether there are password policies associated with the password databases. PAMs are a series of configuration files that provide dynamic authentication for administrative and other services. The rules associated with PAM files in the `/etc/pam.d/` directory can further specify local access limits for different administrative tools and commands.

Some Linux appliances, such as home routers, include default usernames and passwords. It is important to configure such appliances with nondefault usernames and strong passwords. One recently discovered botnet, named for Chuck Norris, can infect and take over such Linux appliances if their administrators retain standard or weak usernames and passwords. The botnet was discovered by Masaryk University researchers in the Czech Republic with the help of a **honeypot**, which tempts black-hat hackers with seemingly valuable data.

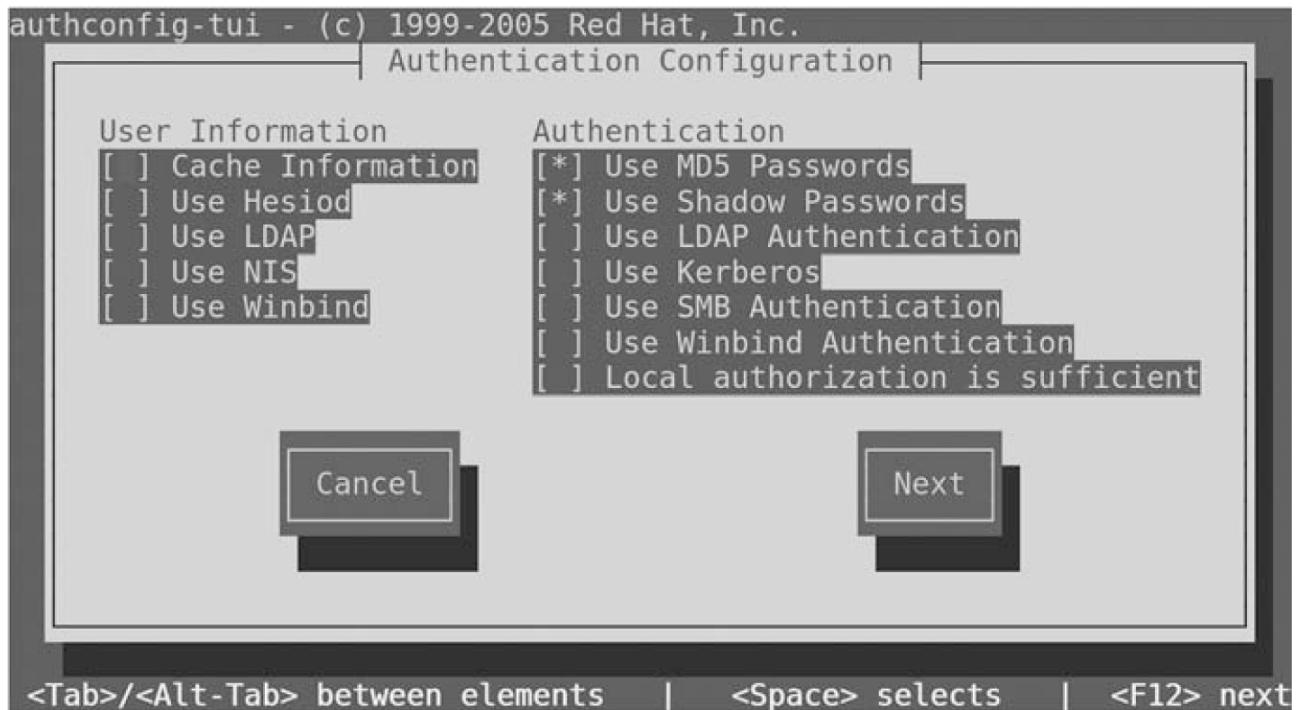


FIGURE 2-2

Red Hat's Authentication Configuration tool.

Protecting Files with Ownership, Permissions, and Access Controls

Everything on Linux is represented by a file. A directory is a special kind of file, which should be executable for users who are allowed to list files on that directory. If the execute permission isn't set, you won't be able to change into that directory using the `cd` command. Other special kinds of files include devices, commonly used to represent and communicate with certain hardware components; character devices such as sound cards; block devices such as drives and partitions; and soft links, which can connect files across different partitions or volumes.

Every file and directory on a Linux system is owned by a specific user and group. With discretionary access controls, the file or data owner can set different read, write, and executable permissions for the user owner, the group owner, and all other users.

While a lot can be done with read, write, and execute permissions for a file, such permissions may not provide enough control. For an additional layer of permissions, Linux can work with access control lists. To make that happen, the target Linux filesystem(s) must be configured and mounted with the access control list option. Once mounted, you can configure access control masks that supersede regular file permissions.

Current Linux distributions may include one or two levels of administrative access control. To give administrative access to individual utilities to specific users, you would use the **sudo** command and configure the access using the /etc/sudoers file. Later distributions support fine-grained access to hardware components through the **PolicyKit**.

So Many Access Controls

Linux implements access controls in a couple of different ways. Even if you've studied them in other Linux books, it may be difficult to keep them straight. So here's a short primer of various access controls in Linux.

Discretionary access controls are read, write, and execute permissions that can be given to users and groups. They are called discretionary because they can be implemented at the discretion of the data owner. Linux includes two types of discretionary access controls. The first is standard octal read, write, and execute permissions. The second is **access control lists (ACLs)**. ACLs provide a second layer of discretionary access control. Specifically, they support read, write, and execute permissions that may supersede other discretionary access controls.

Mandatory access controls are systems like SELinux and AppArmor. You'll have to choose one or the other, as these systems are not compatible. While SELinux is the default for Red Hat distributions, AppArmor is the default for many other Linux distributions, including Ubuntu and SUSE.

When used together, discretionary and mandatory access controls provide multiple layers of protection. These controls are usually also coupled with the firewall protection associated with the **iptables** command. Most iptables rules also control access to systems and protocols, making them another layer of access control.

Firewalls and Mandatory Access Controls in a Layered Defense

Defense in depth is a common and well-accepted approach to implementing security on any system. A firewall may be your first line of defense when it comes to remotely accessing a system, but it shouldn't be the last. In addition to the firewall, a system administrator may implement TCP Wrappers and **mandatory access controls** using either AppArmor or SELinux. Mandatory access controls are permissions that have been set administratively. Individual users cannot change them.

When it comes to protecting your remote services, however, the starting point should be a firewall. Linux has made several attempts to implement a firewall. The initial implementation, ipfwadm, based on BSD's ipf, was included with Linux 2.0. When Linux 2.2 was released, it included ipchains. That was replaced by iptables in version 2.4. That version of the firewall software for Linux stuck around through version 2.6 and into the 3.0 line. It wasn't until version 3.13 was released in early 2014 that the next iteration arrived. This time, it's called nftables.

Firewall Support Options

Most Linux distributions still use iptables to manage firewall rules. Although you can get a GUI to manage the rules for you, they eventually translate to a set of calls to the iptables program. This program manages the in-memory settings that determine how to categorize and then filter packets flowing through the system. These can be inbound, outbound, or just moving from one interface to another. Iptables can also perform network address translation, where Internet Protocol (IP) addresses are manipulated as a packet passes through the filter. A common set of packet filter rules will look like the following.

```
iptables -A INPUT -s 10.0.0.0/8 -d 192.168.1.10 -p tcp -m tcp
      --dport 80 -j ACCEPT
iptables -A INPUT -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -p tcp -m tcp -j LOG
iptables -A OUTPUT -j ACCEPT
```

In the first rule, the **-A** indicates you are appending to an existing chain named **INPUT**. This chain is used when the filter receives packets coming into the system. It also looks for source and destination addresses. The source address is an address block, while the destination is a host address because it doesn't include the bitmask. After that, you check for the protocol in use by using the **-p** flag. You can also specify a particular matching strategy using the **-m** flag. In this case, you are just looking to match with **tcp** packets. The parameter **--dport** specifies the destination port, which is the port used for Web

servers, 80. Finally, with `-j` you indicate where you are jumping. You have the option to ACCEPT, DROP, or LOG as standard jump points. In the third rule, you are performing logging on all output packets that are TCP.

The Linux netfilter modules have always been stateful. The second rule looks at the state of the connection. In this case, you are looking for new connections as well as established ones. The established connections are ones that have previously been allowed through the firewall. You can check for NEW, ESTABLISHED, or RELATED. In the case of RELATED, you may be looking at a data flow that is separate from the flows you've seen. One example of this is an FTP transfer. In an active FTP session, the data transfer happens over a different set of ports, which would be related to control session.

Red Hat Enterprise Linux (RHEL) uses a different method for managing firewalls. Where other Linux distributions use iptables, RHEL switched to using firewalld in RHEL7. Firewalld uses zones to categorize rules. You can associate different rules with different zones and use different interfaces with different zones to pick up different rules. This makes it a lot easier for enterprises to more easily support multiple security zones on a system that may have an administrative interface as well as an interface that may be public facing. Another feature of firewalld is specifying where the changes take place. Either they are made to the rules in memory or they are made on a permanent basis to disk. With iptables, all changes are made in memory. To make them persistent, you had to dump the rules from memory into a file and then read that file in at boot time.

[Figure 2-3](#) shows the user interface for firewalld. Rather than specifying rules by port numbers, as was the case with iptables, firewalld keeps track of all of the services installed on the system. You specify rules based on the services on the system. This can help to protect the system by not opening up ports for services that are not installed appropriately.

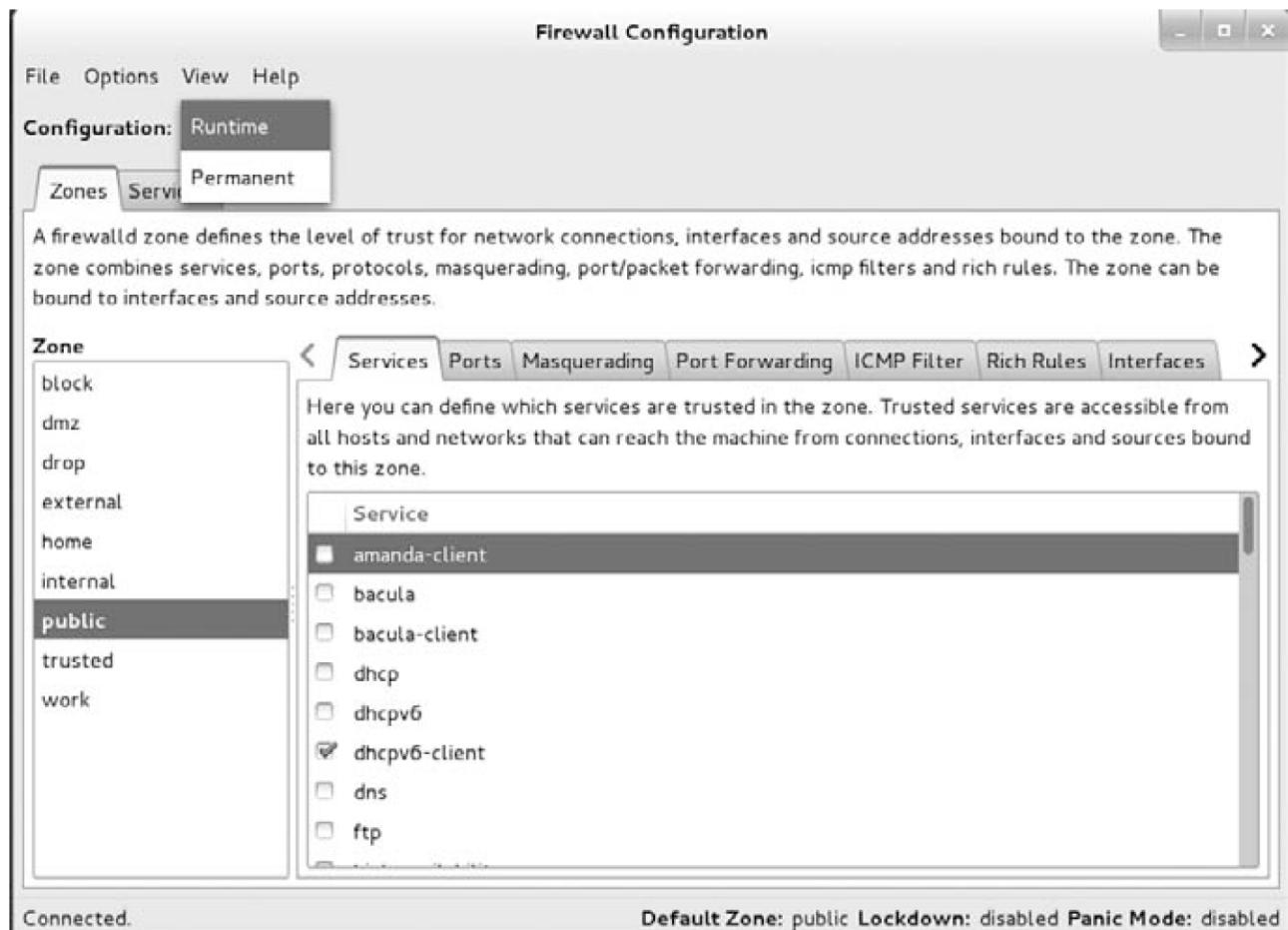


FIGURE 2-3
The firewalld interface.

Mandatory Access Control Support

To paraphrase a famous philosopher, networks cannot be protected by firewalls alone. The mandatory access controls associated with SELinux and AppArmor provide another layer of security. When properly configured, these controls can check access to services and commands to make sure they're run only by intended users, in properly labeled directories.



You can't use both SELinux and AppArmor on the same system.

Proper mandatory access controls make a system more robust. Compromised users, even service users with some level of administrative privileges, can't use other services. However, the configuration of a mandatory access control system requires discipline. For example, you can't just add a new directory to help organize FTP server files. With a mandatory access control system like SELinux, you must make sure those directories are labeled and configured for their intended purposes.

It's normally best to administer a system from the command-line interface. However, some of these tools are tricky. Some GUI tools can help less-familiar administrators understand more about the service. To that end, Red Hat's SELinux Administration tool, shown in [Figure 2-4](#), is excellent.

In a similar fashion, AppArmor includes profiles for specific commands. You can add profiles for additional commands as you learn more about mandatory access controls. In contrast, SELinux is an all-or-nothing security tool, with restrictions for just about everything.

Both AppArmor and SELinux include a trial mode, where violations are logged without preventing access. On AppArmor, this is known as complain mode; on SELinux, this is known as permissive mode. The resulting log files can help you better customize either of these mandatory access control systems.

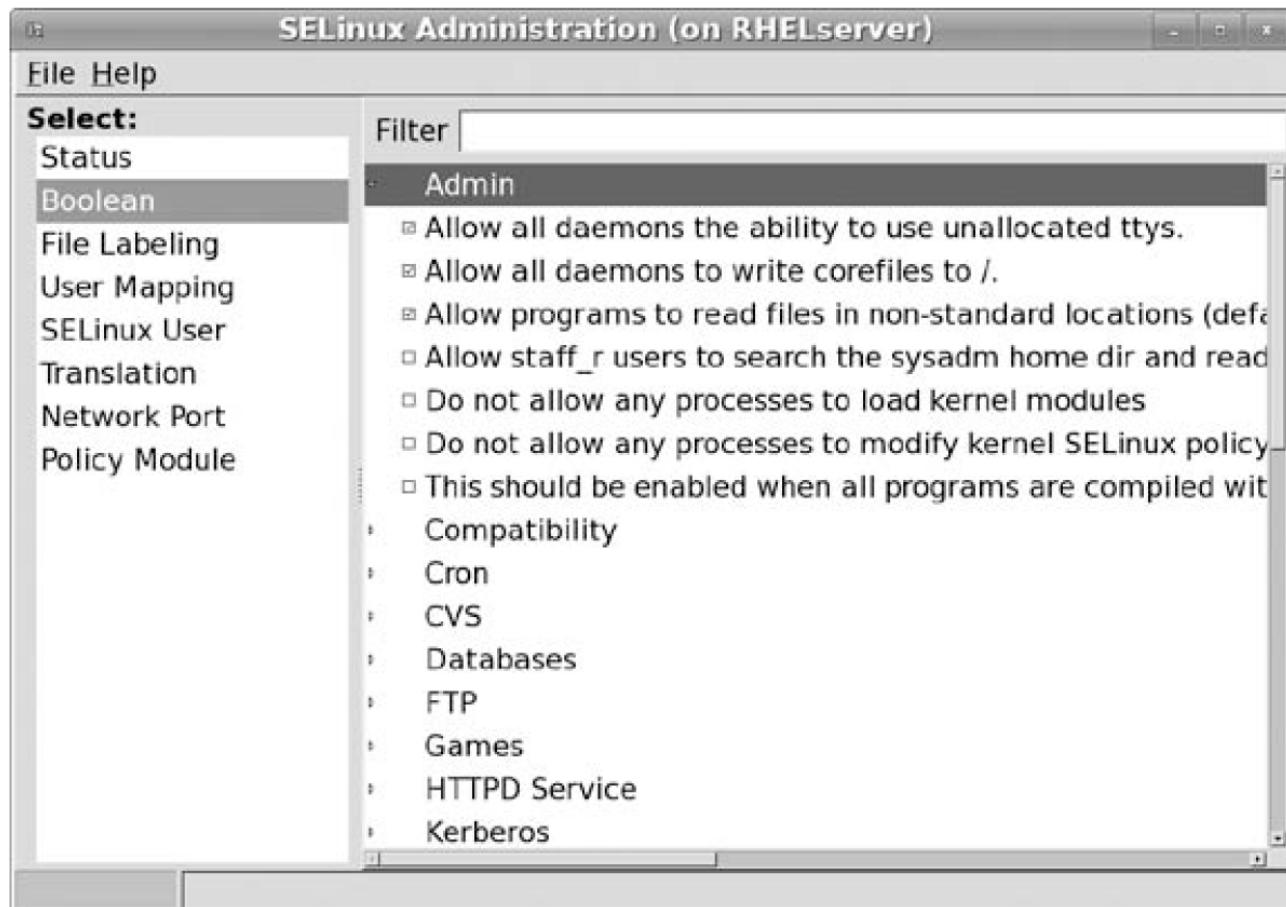


FIGURE 2-4

The SELinux Administration tool.

SELinux, originally developed by the National Security Agency, has a reputation for being difficult to manage, especially if you are trying to implement it with specialized services and applications. Understanding how to correctly manipulate the access control lists requires a fairly steep learning curve, but it is possible to have a system that is less prone to infiltration. AppArmor is another approach, using profiles to achieve application protection. AppArmor also attempts to supplement the discretionary access controls provided by Linux with mandatory access controls, just like SELinux, but the implementation is much different.

Protecting Networks Using Encrypted Communication

In the current security environment, you might be surprised by the number of users who still use cleartext communication protocols. Telnet is still a popular way to log into remote systems. Unencrypted protocols are still frequently used for other functions such as file transfers, e-mails, and backups.

Cleartext communication dates back to the early days of Unix, which was used primarily at universities where colleagues were more likely to trust each other. As enterprises move from Unix to Linux, a few older users may retain their preference for applications that use cleartext networking. These services remain in use in Linux, although due to an increased focus on encryption and security, there is a push to move away from Telnet and FTP because of their cleartext nature.

With the ready availability of encrypted connection protocols, there is no reason to use cleartext connection tools such as Telnet. With the `ssh` command, you can connect to remote systems with the SSH server. The communication—and more importantly, the password—is encrypted by default. You can even set up passphrases to protect the private key used to encrypt communication between the two endpoints. Passphrases are more secure than passwords. Because the passphrase is used solely to give access to the encryption key, it is never transmitted over the network, even in an encrypted manner.

SSH can be used for more than just shell connections. It can also be used to securely transmit backups and more. It can even be used to connect securely to a remote GUI client. While it's possible to tunnel many different connections over SSH, SSH packages commonly come with a secure copy client (`scp`) that is used to transfer files over an encrypted SSH connection.

FYI

The Telnet client is a very useful tool. It can help you verify that a service is accessible. You can use the `telnet` command both locally and over a network to verify the availability of a service over any TCP/IP port. By using the Telnet client and specifying a port number after the hostname, you can initiate a TCP connection to the port on the host, giving you a raw TCP connection to that port. This is very valuable for manually testing services or quickly verifying that a port is listening.

Tracking the Latest Linux Security Updates

For many regular users, it's good enough to download and install updates provided by a distribution when they become available. But Linux is used by a range of users, from home hobbyists to engineers and scientists who know a lot about Linux, to administrators like you, and more. The response of each of these groups to a security update may vary.

The developers behind most Linux distributions do an excellent job with security updates. They monitor security issues in real time. They monitor the work of the open source community as security issues are addressed.

Linux Security Updates for Regular Users

Generally, the package updates released for a distribution have been tested with standard configurations. Linux users who don't have a lot of time to maintain their systems may choose to set up automatic updates. That works if the user trusts the company and the open source community behind a distribution. Alternatively, you can read every security bulletin to see if and how the problem affects the systems on your networks.

In general, it's not too difficult to back out of a security update. In the worst case, configuration files can be saved. Updates can be uninstalled. Older versions of updated packages can be installed again.

Kernel updates may be kept out of automatic updates to ensure the stability of the system. If you do update the kernel, the update package will leave the existing kernel in place and configured in the boot loader. This ensures that you have a fallback option in case the new kernel doesn't work as expected, either as it is or with applications you are running.

Linux Security Updates for Home Hobbyists

Some people may think of Linux users as tinkerers—people who like to experiment with their computers. Home hobbyists like to work with the latest in hardware. Some hardware manufacturers develop drivers for Microsoft operating systems first. Even then, not all hardware manufacturers release their drivers under an open source license. Linux developers frequently have to compile their own code to work with the latest hardware. The first versions of compiled code may be tied to a specific kernel release.

Some specialty software may also be tied to a specific version of a kernel. Examples include some VMware and Virtualbox virtual machine systems. If you install a new kernel, you'll need to reinstall and/or recompile the code for that software. If you install a kernel security update, you're installing a different kernel release. Kernel drivers, like those used by virtualization software, must be rebuilt to work with the new kernel. In many cases, the kernel modules are tagged with the kernel version and won't work with any other version. Most importantly, though, the modules have to be built to ensure all the right kernel pieces are in place to support the driver.

Linux Security Updates for Power Users

Users who run Linux at home are more likely to be happy to run Linux in the enterprise. However, these users may want to install software that requires additional work to compile. This custom work has a significant impact when it comes to updates because the automatic updates won't work on the custom-built software. This is because the automatic update software isn't aware of the custom-built software. This means users must pay attention to security alerts for that software package, if they can find them.

Other types of users have other needs. While you generally want to limit the number of packages installed on a system and reduce the amount of administrative control any user has, this isn't always possible. Developers, for example, need ready access to administrative tools such as compilers. Compilers can be used by malicious users or attackers to build custom software that can be employed to launch further attacks. This concern was far more valid when there was less homogeneity across Unix implementations. If a malicious user gets into one Linux system, the same pre-built binary will work on that Linux system just as it will work on any other Linux system. This is because the application binary interface that all programs use is the same across versions and implementations of Linux, or else it's no longer Linux. The only variation here would be if you ran across a very, very old version of Linux. In that case, the application binary interface might be different enough that a newer binary may not work.

Security Updates for Linux Administrators

As a Linux security professional, you should read all security bulletins related to software on your systems. You should consider the effects of updates before deciding whether they should be installed.

If you determine that the security bulletin does not apply to the systems on your network, don't install that update. Unless some changes are made, that update will still be there. Depending on several factors, the updated software may be in a distribution-specific repository or a download from a public server maintained by the developers of the software in question.

To deviate from updates provided for a distribution, you can create a local update repository, loaded with packages customized for local systems. You then configure the systems on the local network to take their updates from that local repository. Yes, that may involve some extra additional work, but a local update repository can also save bandwidth on a network's connection to the Internet.

In other words, every Linux workstation and server normally gets its updates from remote repositories somewhere on the Internet. All packages downloaded for a specific distribution are the same. Linux workstations may require periodic updates of huge packages, such as for the [OpenOffice.org](#) suite. If you create a local update repository, these packages are downloaded from the Internet only once. The updates can then be pulled over the local network from each system.

Linux Security Update Administration

If you administer just a few Linux systems, you can manage their updates directly. As the administrator, you may be able to administer those systems remotely with a tool such as SSH. You can then run needed update commands on those systems.

If you administer a substantial number of Linux systems, consider the centralized management tools available for several Linux distributions. Tools such as the Red Hat Network, Canonical's Landscape, and Novell's Zenworks support the administration of groups of computers from a centralized interface. While these tools require subscription fees, they do save time. If you manage that many systems, you are probably already paying one of these companies for a support services anyway (and if you aren't, you probably should be).

With these tools, you can set up a single command or a script to be run on a group of computers. For example, you might set up different update commands for servers, engineering workstations, and data-entry systems.

The Effect of Virtualization on Security

Virtualization makes it easy to start with a very secure baseline system. Because virtual-machine hard disks reside on only a few very large files, it takes just a few commands to create new virtual machines. In other words, you can start with a stripped-down, well-hardened system and add a single service, blocking packets to all unused ports, and securing every other account and directory with appropriate mandatory access controls.

Virtualization is essential to most computing environments, whether it's a small company or a much larger, enterprise network. One of the issues with virtualization is that you don't have physical systems to look at. Instead, you have a lot of logical systems that may be much easier to overlook. A virtual machine that is forgotten is not updated. And a system that is not updated is an open invitation to a malicious user to break into your system. If that system remains forgotten, it can serve as a foothold—a way for that malicious user to break into more important parts of your network.

While virtualization has roots in the time sharing found on mainframe systems, it has taken a number of new turns for modern computers. There are five different categories of virtualization:

- **Applications**—While a number of companies are developing virtual machines that encapsulate single applications, Linux already has an application-level virtualization tool known as Wine Is Not an Emulator (WINE).
- **Platform-level virtual machines**—The first PC-based virtual machines were applications installed in operating systems. Examples of virtual machines installed as applications include VMware Player, VMware Workstation, Virtualbox (open source edition), and Parallels desktop. In the language of virtualization, these are called Type 2 hypervisors. A **hypervisor** is the software that manages and runs the virtual machines. In these cases, there is a host operating system that sits between the machine (or bare metal) and the hypervisor, or virtualization software.
- **Paravirtualization**—A software interface for virtual machines with limited resources. Paravirtualized virtual machines can be configured on older CPUs.
- **Hardware-assisted virtualization**—The processor here has extensions to support various supervisory functions in hardware rather than having to support something like an emulator in software. For the last several years, both AMD and Intel have supported these virtualization extensions in their CPUs.
- **Bare metal virtualization**—In the language of virtualization, these are called Type 1 hypervisors and the virtualization software runs directly on the hardware without a host operating system. VMWare's ESX Server and Citrix's Xen Server as well as Microsoft's Virtual Server are all examples of Type 1 hypervisors that operate on the bare metal.

To confirm that a system can handle hardware virtualization, examine the /proc/cpuinfo file. If it is an Intel CPU, it should include the **vmx** flag. If it is a CPU from Advanced Micro Devices (AMD), it should include the **svm** flag.

Variations Between Distributions

One of the hallmarks of open source software is diversity. For example, you can select from at least four different Simple Mail Transfer Protocol (SMTP) servers to manage e-mail on a network. While some distributions focus on system administration, others focus on consumer applications such as multimedia. There are even live CD Linux distributions with security tools.

Despite the diversity in standard services, you can generally install (or remove) the software, services, and applications of your choice on any Linux system. For example, while Ubuntu developers include **Postfix** as the default SMTP server, you can install other SMTP servers, including **sendmail**, **Exim**, and **Qmail**, directly from Ubuntu repositories.

A Basic Comparison: Red Hat and Ubuntu

This book focuses on two Linux distributions: Red Hat Enterprise Linux and Ubuntu Server Edition. The developers behind these distributions have made different choices. Although the Red Hat and Ubuntu distributions occupy a large share of the Linux marketplace, there may be another distribution that better fits your needs. Understanding the differences between these distributions can help you make a better choice.

The biggest difference may be the package-management system. Not surprisingly, Red Hat uses the Red Hat Package Manager (RPM) to build packages from source code and install them on the Red Hat distribution. Red Hat also uses a utility called the Yellowdog Update, Modified (yum) to actually install the RPM packages to the system while also checking for dependencies. Because Ubuntu built its distribution from Debian Linux, it uses the Debian Package Manager to build and install its packages.

There are also similarities between these distributions. For example, Red Hat and Canonical (Ubuntu) both release distributions with long-term support. Red Hat supports its Enterprise Linux releases for seven years. The most recent version of Red Hat Enterprise Linux (RHEL) is RHEL7, released in the first half of 2014.

Ubuntu releases distributions every six months, and releases more robust distributions every two years. These releases are noted as long-term support (LTS) releases. The server edition of every Ubuntu LTS release is supported for five years.

Along with long-term support, Red Hat and Canonical have made a number of other similar decisions. Both companies offer subscriptions to a Web-based support tool that allows authorized users to administer groups of Linux systems remotely.

Both Red Hat and Canonical have made similar choices for some default services. For example, both use Apache as the default Web server, the **very secure File Transfer Protocol daemon (vsftpd)** service as the default FTP server, and the **Common Unix Printing System (CUPS)** as the default print server. Incidentally, both Red Hat and Canonical use the GNU Network Object Model Environment (GNOME) as the default graphical desktop, although the actual implementation looks different from one system to another because of custom modifications.

There are differences that go beyond the package-management system, however. For example, Ubuntu includes a minimal installation option with its server release, well suited for bastions on virtual machines. That installation includes a kernel optimized to work as part of a virtual guest.

Red Hat and Ubuntu have also made different choices in terms of default applications. Of course, you can override these with the applications of your choice. In many cases, Red Hat and Ubuntu also support these alternatives.

Sometimes, Red Hat and Ubuntu learn from each other. Given their releases under open source licenses, they can take the source code from a competitive distribution and use it themselves. Ubuntu uses a number of GUI administrative tools originally built for Red Hat.

More Diversity in Services

Linux includes a diverse range of services. If a security issue arises with one service, it's normally best to install the update or patch to address the security issue. Sometimes, however, that update might cause more severe problems.

In that case, you may want to consider a different service. [Table 2-3](#) lists a number of alternatives for different services. It is not by any means a complete list. However, it does describe alternatives if one service option has a fatal security flaw. It also gives you an idea of how many services could be installed on a single server.

TABLE 2-3 Alternatives for different services.

FUNCTION	SERVICE OPTIONS	NOTES
Domain Name System (DNS)	Berkeley Internet Name Domain (BIND) Daniel J. Bernstein's DNS (djbdns)	DNS is a hierarchical database of domain names and IP addresses. It enables you to look up hostnames from IP addresses and vice versa. BIND and djbdns are both implementations of a DNS server. BIND is the most common DNS server on the Internet. Djbdns is a relatively lightweight DNS server alternative to BIND.
File transfer	Professional FTP Daemon (ProFTPD) Troll FTP vsftp Trivial File Transfer Protocol (TFTP)	TFTP is a protocol and service that uses a simplified form of FTP. It is a very lightweight means of transferring files with very few commands available.

Graphical desktop environment	GNOME K Desktop Environment (KDE) Xfce Lightweight X11 Desktop Environment (LXDE)	In Linux, the graphical desktop environment is separate from but requires the use of an X Windows System Server. It may also include a window manager to control the placement of windows within that GUI.
Graphical user interface	X.org XFree86 Freedesktop.org	XFree86 is an implementation of the X graphical interface
Graphical login manager	GNOME Display Manager (GDM) KDE Display Manager (KDM) X Display Manager (XDM)	A graphical login manager is a service for graphical logins to a Linux GUI. GDM is a graphical login manager built by the developers of the GNOME Desktop Environment. KDM is a graphical login manager built by the developers of KDE. XDM is a graphical login manager built by the developers of X.Org GUI server.
Mail user agents	Cyrus Dovecot	Cyrus is an e-mail server developed at Carnegie-Mellon University, primarily for IMAP version 4 e-mail delivery. Dovecot is an open-source e-mail service designed for regular and secure versions of the POP and IMAP protocols. Cyrus and Dovecot are implementations of mail servers that are used to retrieve mail for clients.
Printing	CUPS System V Line Printer next generation (LPRng)	LPRng is a server used to send print jobs to or from users or systems.
Remote connections	Secure Shell (SSH) Remote Shell (RSH) Telnet Kerberos Telnet	Kerberos Telnet is a version of the Telnet server that can use Kerberos tickets to enhance security.
Mail transport agents	Postfix sendmail Sendmail Exim Qmail	
Structured Query Language databases	MySQL PostgreSQL Proprietary options from Sybase Oracle	MySQL is open source database program, currently owned by Oracle. PostgreSQL is an open source database program sponsored by a variety of open source and other IT companies.
Web server	Apache Boa Caudium Lighttpd Roxen Sun Java	



CHAPTER SUMMARY

Linux security starts with the kernel. The modularity of the Linux kernel means that it is more robust. Many modular options affect security. To that end, you need to choose whether to accept the kernel as customized for your distribution or whether to customize and compile the kernel yourself. Linux security continues with the boot process. Good security prevents unauthorized users from booting from a live CD and selecting administrative options from the GRUB or LILO boot loaders. Beyond the basic operating system, you should keep active services to a minimum. If possible, don't install a GUI.

File ownership and permissions start with discretionary access controls, enhanced by access control lists. The use of sudo can provide restricted administrative privileges and eliminate the need for someone to be logged in as root, which can be dangerous. Host-based firewalls using iptables or firewalld will help to prevent unwanted network communications from getting into the system. You can provide additional protection on services that have to be exposed through the firewall by using TCP Wrappers through the internet super server or by using additional restrictions built into specific services. Mandatory access control systems such as SELinux and AppArmor add another layer of control, protecting your system even if there is a security breach. With the availability of SSH, there is no reason to use cleartext protocols such as Telnet.

With respect to security, you may need to pick and choose whether to download a security update. Some security updates may not apply to all configurations, although automatic update services will determine whether distribution-provided updates are relevant to your system. If they are, they will either be updated automatically or be presented to you as options for updates. Virtualization adds another dimension to security. Isolating services to appropriate security zones within the same physical host will help to improve the security of other systems. Services or systems that need different levels of protection can be put on other virtual machines, limiting points of attack. Finally, if you understand the differences between Linux distributions, you can make better decisions when it comes to choosing security options for the network.



KEY CONCEPTS AND TERMS

Access control lists (ACLs)

Berkeley Internet Name Domain (BIND)

Binary kernel

CentOS

Common Unix Printing System (CUPS)

Cyrus

Daniel J. Bernstein's DNS (djbdns)

Default runlevel

Discretionary access controls

Domain Name System (DNS)

Dovecot

Exim

File Transfer Protocol (FTP)

GNOME Display Manager (GDM)

Grand Unified Bootloader (GRUB)

Graphical desktop environment

Graphical login manager

Honeypot

Hypervisor
iptables
KDE Display Manager (KDM)
Kerberos Telnet
Knoppix
Line Printer next generation (LPRng)
Linux distribution
Linux kernel
Linux Kernel Organization
Linux Loader (LILO)
Live CD
Mandatory access controls
Modular kernel
Monolithic kernel
MySQL
Network Time Protocol (NTP)
Patch
Pluggable authentication modules (PAMs)
PolicyKit
Postfix
PostgreSQL
Qmail
Runlevel
sendmail
Sendmail
Source code
sshd
syslog
Trivial File Transfer Protocol (TFTP)
Very Secure File Transfer Protocol daemon (vsftpd)
Winbind
X Display Manager (XDM)
XFree86



CHAPTER 2 ASSESSMENT

1. Which of the following statements best describes the structure of the Linux kernel?
 - A. A single monolithic kernel
 - B. A completely modular kernel
 - C. A modular core with monolithic components
 - D. A monolithic core with modular components
2. The Web site associated with the Linux Kernel Organization is _____.
3. Which of the following statements is *not* true about a live CD distribution? Assume your system can boot from appropriate locations.
 - A. It can be booted from a DVD drive.
 - B. It can be booted from a USB port.
 - C. It automatically installs that Linux distribution on your system.
 - D. It provides administrative control of your system without a password.

- 4.** Which of the following is a security risk associated with the LILO boot loader?
- A. Changes to LILO can be password protected.
 - B. It supports password-free access to the administrative account.
 - C. It allows a user to boot Microsoft Windows.
 - D. It supports the booting of a monolithic Linux kernel.
- 5.** Which of the following services should *not* be disabled on a bastion host used as an FTP server? Assume that the host is administered remotely, over an encrypted connection.
- A. SSH
 - B. Telnet
 - C. CUPS
 - D. iptables
- 6.** Which of the following is *not* a potential security issue with respect to the Linux GUI?
- A. The Linux GUI is a client-server system.
 - B. Linux GUI applications can be networked.
 - C. Linux GUI applications can be accessed over an SSH connection.
 - D. Users can log into the Linux GUI remotely.
- 7.** Which of the following authentication tools work locally?
- A. NIS
 - B. PAM
 - C. LDAP
 - D. Winbind
- 8.** Which of the following is an example of discretionary access controls?
- A. SELinux
 - B. AppArmor
 - C. PolicyKit
 - D. User-defined read, write, and execute permissions
- 9.** Which of the following options is *not* used to block access from certain IP addresses?
- A. iptables
 - B. SELinux
 - C. TCP Wrappers
 - D. Extended internet super server
- 10.** Which of the following statements best describes the role of mandatory access controls?
- A. They protect other services after a security breach in an account.
 - B. They protect a system from access by a malicious user through firewalls.
 - C. They disable cleartext services such as Telnet.
 - D. They provide specific requirements for access to critical services.
- 11.** Packages associated with SSH include a client for which of the following protocols?
- A. Samba
 - B. FTP
 - C. Telnet
 - D. SMTP
- 12.** Under normal circumstances, what happens when a system can't be booted with a newly installed Linux kernel?
- A. You need to install the old kernel.
 - B. The system can't be booted. You need to reinstall that Linux distribution.
 - C. The system can't be booted. You need to recover the old Linux kernel with the help of a recovery or rescue mode for that distribution.
 - D. The old kernel is still available through the boot loader.
- 13.** What is the best course of action if you want to take control of those packages that are updated on your distribution?
- A. Create your own update repository.

- B. Deselect the packages that should not be updated.
 - C. Change to a different distribution.
 - D. Use the update repositories from a different distribution.
- 14.** Which of the following is *not* a standard open source option for SMTP e-mail services?
- A. sendmail
 - B. Postfix
 - C. Dovecot
 - D. Exim

PART TWO**Layered Security and Linux****CHAPTER 3****Starting Off: Getting Up and Running****CHAPTER 4****User Privileges and Permissions****CHAPTER 5****Filesystems, Volumes, and Encryption****CHAPTER 6****Securing Services****CHAPTER 7****Networks, Firewalls, and More****CHAPTER 8****Networked Filesystems and Remote Access****CHAPTER 9****Networked Application Security****CHAPTER 10****Kernel Security Risk Mitigation**

CHAPTER

3 Starting Off: Getting Up and Running

LINUX COMES IN HUNDREDS OF VARIETIES, called *distributions*. Some distributions may have a specific focus, so it's useful to know what each distribution is good for and what its philosophy is regarding updates and software versions. After you select a Linux distribution, you must make a number of other choices. These include how you are going to get it, how you are going to install it, and how you are going to load it once it's installed. You will also need to decide how much software you are going to install and assess how that will affect the security posture of your resulting installation.

Chapter 3 Topics

This chapter covers the following topics and concepts:

- How to pick a distribution
- How to pick a delivery platform
- How to choose a boot loader
- What services are
- What inetd and xinetd do
- How to use r-services

Chapter 3 Goals

When you complete this chapter, you will be able to:

- Choose the Linux distribution that will work best for you
- Select a delivery platform
- Determine the services installed on a Linux system
- Enable or disable network services
- Protect network services with TCP Wrapper
- Configure xinetd services
- Describe the risks associated with r-services

Picking a Distribution

Before you can do anything at all, you need to pick a Linux distribution. But first, what is a distribution? Linux is merely an operating system or *kernel*. For it to be useful, it requires a number of other software packages that a user will actually interface with. These include network services, graphical user interfaces (GUIs), language compilers, and many other kinds of software. Linux distributions bundle their own collection of software packages with specific versions of those packages. Distributions also bundle in specific ways all the files required

for a piece of software to be installed. These software bundles are generally called **packages**. Each distribution has software that will unbundle and install the software inside each of these packages.

There are hundreds of distributions of Linux available, and each one is unique. It may simply be that a different set of software packages gets installed by default, or it may be that a different package manager is used. In any case, each one is different, or the distribution's maintainers would have simply used an existing distribution rather than bothering to create their own.

There is, though, something to be said for the technical credibility you may get from developing your very own distribution of Linux for you and your friends. In spite of the fact that it's all Linux, and the packages you will have to choose from are mostly the same across the board, there's something personal about picking a Linux distribution. Each one may offer a slightly different look, depending on whether the interface has been customized, or it may have a different approach to the packages that are used. Some distributions, like Debian, go for a more stable experience, while others, like Fedora, may be a little more bleeding edge. Some distributions are even source-based, like Arch and Gentoo, meaning you build from source everything you install.

One current trend is creating new distributions on top of existing ones by maintaining the packages differently or by creating a slightly different look. Some of the most popular distributions, like Ubuntu and Mint, are actually based on a much older one. Lots of distributions come and go because the maintainers create something but don't have a community to support their efforts on an ongoing basis.

How do you go about choosing a distribution, then? A handful of factors may be relevant to you. The first is simply comfort. Because this is Linux, the system will behave the same from one distribution to another in terms of how the kernel and hardware operate. One comfort feature is the package manager. It's likely that you will be installing, updating, and removing packages on a fairly regular basis—certainly when it comes to updates. You want a package manager that you will be comfortable using—one that you understand and with which you can get work done quickly. The interface is important from a comfort perspective, but because you can probably find a way to install at least the major interfaces like GNOME, KDE, or Xfce on any distribution, this is less of a factor.

In any discussion about securing your implementation of Linux, one important factor is how you keep your system up to date. How frequently is the distribution updated? Does it keep up with the latest fixes from the upstream development teams? How thoroughly do they perform integration testing to ensure the stability of the platform? Do they have a way for you to keep track of security updates, whether through a mailing list, a Twitter feed, or an RSS/Atom feed? All of these are important considerations, especially if you're looking to have a server that exposes services to any network, whether it's an intranet or the Internet at large. Reading through the distribution's Web site will help to answer these questions. If there are mailing lists, you may be able to get access to the archives to get a better idea of how often the distribution is updated and how the developers provide announcements about package updates.

You may also consider whether the distribution you are looking at has a server distribution or if there is just a one-size-fits-all model. A server distribution may start off with a very limited set of packages and require you to perform some additional steps to install a user interface on it. The goal of any installation should be to minimize the number of software packages. This will be a recurring theme because the fewer moving parts you have, the less trouble you can get into. In practical terms, having fewer software packages reduces the number of potential vulnerabilities to which you may be exposed. You may run across the terms **threat vectors**—to describe the number of ways into your system—and **attack surface** or **attack canvas**—to indicate the total number of vulnerabilities you may have, which factors in the number of threats. Having a distribution begin with a very small number of packages and then only installing what is absolutely necessary will keep your system less vulnerable, just because there are fewer ways to get in.

The other way a server distribution may differ from a one-size-fits-all or *desktop* distribution is that the kernel may be tuned differently. Because the kernel is responsible for ensuring programs run in a timely fashion, it has to make some decisions about which programs get processor time. It may also need to make some decisions about kicking programs out of the processor based on process priority. This doesn't mean a program will stop, it just means it will temporarily not get time in the processor. Of course, it's a question of milliseconds in most cases, so it's unlikely a system user would notice. But if you have a heavily used system that is getting a lot of network or other user requests, those milliseconds may be critical and will add up.

The following are some of the different distributions that you may commonly run across, their package-management systems, and the support infrastructure behind them.

- **Red Hat Enterprise Linux (RHEL)**—Red Hat is the company behind this distribution. You pay for it, which means you get support. This is a stable distribution designed for use in enterprise environments. Red Hat has long used the Red Hat Package Management (RPM) format to create its packages, which allows for tracking dependencies. You can use the rpm utility to install, query, and remove packages. But the Yellowdog Updater, Modified (yum) is more commonly used for installation because it will download, check dependencies, and install the dependencies before installing the package you asked to install. RHEL has also moved to using different subsystems than most of the other Linux distributions, meaning you manage it differently. Red Hat supplies regular announcements of its updates. These are very detailed with respect to versions of packages and RHEL releases.
- **Fedora**—This is also a Red Hat distribution, but it's community supported. Because it's from Red Hat, it uses rpm for packages and yum for easier package management and installation. It was formerly known as Fedora Core and is supported by the community, although officially Red Hat maintains it. Fedora also has very short support windows, because the distribution issues regular updates. It is generally considered the development platform for RHEL. New features are added to Fedora and eventually, RHEL branches off from a Fedora release.
- **CentOS**—This is identical to RHEL except for the branding. All the logos and names have been changed. CentOS is built from the source of RHEL, making it binary identical to RHEL. But because it's not a Red Hat distribution itself, you don't get support from Red Hat. Unlike Red Hat, there is no company with a support staff in place for CentOS. Instead, it's supported by the community. It's available free of charge, however. Because it is binary identical to RHEL, package management is the same as for RHEL.
- **Debian**—One of the older distributions, Debian is named for the distribution's founder and his then-girlfriend (Ian and Deb, in reverse order). Debian offers mailing lists and [wikis](#), which are community-maintained documentation repositories to support the distribution. It recommends that users look for answers themselves before turning to other support mechanisms. This includes searching through the documentation and mailing list archives. Debian uses the Advanced Package Tool (apt) for package management. There are a number of utilities for searching for, installing, updating, and upgrading packages. Debian's package system keeps track of dependencies and the apt utilities resolve dependencies and install the packages required for other packages to function correctly. There are also other tools like synaptic that can be used as more graphical front ends to the package-management system.
- **Ubuntu**—Ubuntu is based on Debian. Package management is all inherited from Debian. Support, however, is offered by Canonical, the company that maintains Ubuntu. There are support Web sites and mailing lists. Ubuntu also marks periodic releases for long-term support (LTS), meaning the release is supported for three years. Because Ubuntu issues a release every six months, each release is commonly supported only until it is superseded by the next release. The LTS releases are issued every two years in the April release. Ubuntu has spawned several distributions for specific purposes, such as Edubuntu. Ubuntu also offers a server version of its distribution.
- **Linux Mint**—Mint is based on Debian and Ubuntu. Like Ubuntu, it relies on community support, but it doesn't have a company behind it as Ubuntu does. Instead, a collection of volunteers manage the distribution. Because it's based on Debian and Ubuntu, it uses the same package-management system. In some cases, the packages from one can be used on another. The most significant difference between Mint and Ubuntu is the desktop. Where Ubuntu has used GNOME and the Unity desktop environment, Mint's team has developed two different desktop environments (built on top of GNOME). Called Mate and Cinnamon, these are likely to appeal more to those who are familiar with desktops like Microsoft Windows.
- **Arch Linux**—Arch is targeted more toward power users who like to have complete customization control over how their packages are built and added to the system. Arch uses a package-management tool called pacman, which can be used to install both pre-built, binary packages and custom-built packages. The user

has complete control over what gets installed, what gets upgraded (or not), and what dependencies are used when creating custom-built packages. Arch can use packages from its repository or can generate a package by building it on the user's system. This is an intermediate step between a built-from-scratch distribution and a completely binary distribution, like the ones noted previously. The risk with a distribution like this is that when you custom build your own packages—especially if you control what gets upgraded and what doesn't—the potential for breaking your system is fairly high, even if you are an experienced user. Support for Arch comes from the user community via mailing lists and Web-based discussion forums.

- **Gentoo Linux**—Gentoo began as a completely custom-built Linux distribution. It is offered here as what may be considered the precursor to Arch. Gentoo users also have a lot of control over what gets built and which dependencies should be considered. Where Gentoo was a fairly popular Linux distribution for technical users several years ago, its audience has largely moved to Arch, according to popularity figures provided by DistroWatch. Gentoo uses a package-management system called Portage, which is based on the ports system from FreeBSD, another Unix-like operating system. Ports are built on the user's system, meaning it takes time to compile all the packages before they can be installed.

RPM Versus Yum

It's not uncommon to have confusion with acronyms, protocols, and formats. As one example, RPM might refer to the format of a Red Hat package, but it may also refer to the utility used to install and manipulate software packages. When you see rpm with lowercase letters, it's the utility or program. RPM in uppercase letters refers to the file structure used for Red Hat packages. The utility rpm has a lot of functions and can install packages as well as query the contents of RPM files. It will also check for dependencies and let you know, specifically, what the dependencies are. While rpm and the package format have been around since the beginning of Red Hat, Yellowdog was another Linux distribution that was based on Red Hat but added an additional package-management utility called the Yellowdog Updater, Modified (yum). The yum utility offered something that rpm didn't—the ability to download packages as well as all of the dependencies for the packages and install them all, leaving you with the package you asked for without any additional work on your part.

Most users, especially businesses, will end up choosing a binary-based distribution, meaning that all of the packages have been compiled and put together for the user and need only be unpacked and placed on the user's system. Distributions that are considered from-scratch, like Arch or Gentoo, are good for serious power users trying either to learn about Linux at a lower level or to wring as much performance out of their system as they can. Those distributions, though, rely solely on community support rather than a company you can call to ensure that problems get fixed. Red Hat Enterprise Linux and Ubuntu both offer server versions. Fedora and Mint are commonly used for desktops and Arch and Gentoo could be created as servers based on the user preferences and build configurations.

Picking a Delivery Platform

After you have selected a distribution, you have choices regarding where to deploy it. Your first thought may be, "Well, on my computer system, of course." But there are other possibilities, depending on what you are using it for. For example, rather than using Linux full time, you may want it as a secondary system for specific a purpose. If you are deploying a server, you may choose not to use one of your systems at all. The options are as follows:

- Deploying on a physical system under your control
- Deploying on a virtual system under your control
- Deploying through the use of a cloud-based provider

In the latter case, the provider may offer a virtual system and a choice of operating systems installed on the system. Or, the provider might simply provide a system where you don't have a choice of operating systems or distributions. In other words, with one provider, you may get a selection of several distributions to choose from, while with another, you may simply get to choose Windows or Linux without any other options.

Physical System

A physical system is one that is under your control. You have access to it and thus complete control over it. There are considerations, however. If you are using a physical system as a desktop system where you are going to be doing most of your day-to-day work, this makes a lot of sense. If it is a laptop, it may be under your control most of the time. However, if it ever gets out of your control, you will want to provide protections to prevent it from booting for someone other than you. These include strong passwords, password-protected screen savers, and basic input/output system (BIOS) passwords or boot-loader passwords.

Having so many passwords to enter just to get into your system can be tedious, but if you store anything personal or potentially sensitive, it's worth it. You will also want to consider encrypting your hard drive in case it gets out of your control. A malicious user can easily bypass physical systems security by simply opening the device and extracting the hard drive. If it isn't encrypted, that malicious user can then read and extract sensitive information. In addition, hackers can **crack** passwords, meaning they can use a specialized piece of software called a **password cracker** to figure out the password. A password cracker either guesses the password from a number of known passwords or simply tries every password possible. This latter approach is called a **brute-force attack**.

A physical system without a BIOS password can also be attacked by using a boot disk that's separate from the disk in use inside the system. This could be a CD, a DVD, a USB flash drive, or even an external USB drive. Once the system is booted, the disk can be accessed (if it isn't encrypted), passwords changed, or information extracted. If you need a physical system to do your work, all of these should be considerations when you are trying to protect it.

Of course, physical systems also run the risk of being stolen outright. With server systems, you can remediate this to a degree by putting them in a data center or server room. A data center or server room could be protected by a lock with a key or, better yet, proximity badges and biometrics. Inside the data center, the server could be locked into a rack or cage to further protect the system from unauthorized access. While laptops are commonly thought to be easy targets because they are so portable, it's not difficult to pick up a desktop or even a rack-mounted server. It's not impossible to unscrew a system from a rack, unplug it, and remove it. These are not very heavy systems. They are going to be a lot more conspicuous, of course, but if an adversary manages to make it into the building, it's not unreasonable to expect that he or she will find a way back out with a system containing a lot of interesting information on it. Of course, an adversary's merely getting hold of the hardware doesn't necessarily put any information contained within the system at risk if the device has a password to protect it from booting when out of the owner's or manager's control. Additionally, encrypting the hard drive will protect its data.



NOTE

One risk associated with adding passwords to the boot process, whether in the boot loader or in the BIOS, is requiring a user or administrator to be in place when the system boots up. This can potentially be dealt with using lights-out management (that is, by remote control) of the server. As always with security, there are tradeoffs between protecting your assets and being able to easily use them.

Virtual Machines

Virtual machines, the idea that you can potentially run multiple operating systems on a single physical device, were first posited in the 1950s and developed by IBM in the 1960s. The first implementation of virtual machines on personal computers was in the 1980s, although there was no hardware support for their implementation at

the time. When an operating system starts up, it expects to have complete control of all of the hardware on the system, as that's one of its major functions. If one operating system is running inside another operating system, the guest operating system won't be able to gain control of the hardware because the host operating system already has it. So there needs to be a way to get the guest to believe it has control of the hardware.

Even if you aren't running one operating system inside another, you are still using virtualization. If a system is going to run more than one program at a time, it needs to support **virtual memory**. Virtual memory is where a program requests some amount of memory, and the operating system suggests to the program that it will get all of the memory. Instead of providing a set of real physical addresses for the program to use, however, the operating system hands in a set of logical addresses that must be translated to actual physical addresses when the program tries to access it.

This is where hardware support comes in handy. Instead of performing software lookups to get the real address from the logical address, which can be time-consuming, the task falls to a specific set of hardware called a **translation lookaside buffer (TLB)**. This hardware looks up one value given another value. This is sometimes called a *hash* or, in some types of hardware, like a network switch, *content addressable memory*.

Virtual machines are popular in part because they offer possible cost savings. Most systems today have far more computing horsepower than they are going to use, particularly on an ongoing basis. Disk space and memory are also very inexpensive. If you pay for enough disk and memory, you can have multiple operating system instances running on a single physical system. This limits your power, cooling, hardware, and floor-space costs because you don't have as many systems installed in your facility. It also makes migrating from one system to another much easier when you want to increase your hardware specifications.

When it comes to operation, there is software to catch the interrupts the guest operating system attempts to send to the hardware. The software must then either handle these interrupts or send them down to the hardware under the control of the host operating system. The software that does this is commonly referred to as a *hypervisor*, and has been since IBM began doing virtualization decades ago.

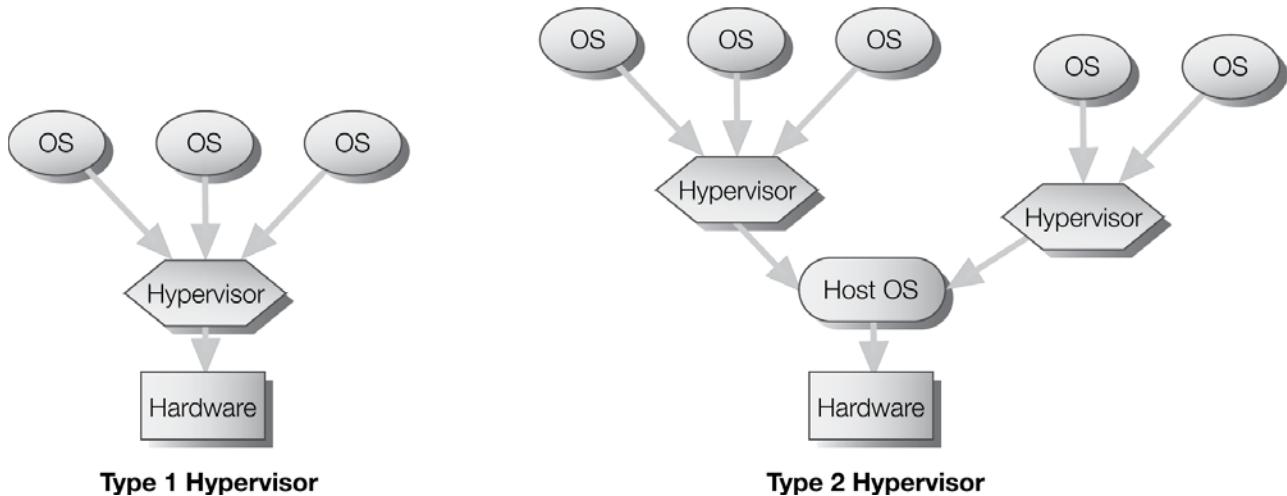
There are two types of hypervisors. The first type runs on what is called *bare metal*. This means the hypervisor is what runs when the system firmware hands off control to the software. You can think of this type of hypervisor as being the operating system that boots up when the system starts. This is a Type I hypervisor. VMware ESX server and Microsoft's Hyper-V server are examples of Type I hypervisors. Perhaps not surprisingly, considering its leadership in this area over decades, IBM's CP/CMS and its current z/VM operating systems are also considered Type I hypervisors.

A Type II hypervisor, on the other hand, uses a host operating system, and the hypervisor runs on top of it. In this case, rather than handling the interrupts directly, as a Type I hypervisor does, the Type II hypervisor intercepts the interrupt and then passes it down to the operating system itself. Software like Parallels Desktop, VMware Workstation, and VirtualBox are examples of Type II hypervisors. You can see a visual representation of the two different types of hypervisors in [Figure 3-1](#).



NOTE

Software that controls running programs has sometimes been called a *supervisor*. Because virtualization software will run multiple supervisors, another word that would sound like a higher rank than supervisor was needed. The result is the word *hypervisor*.

**FIGURE 3-1**

Type I and Type II hypervisors.

Linux has long had virtualization abilities. Currently, Linux has the Kernel-based Virtual Machine (KVM), a kernel module that supports virtual machines. Because it's built into the kernel, KVM could be considered a Type I hypervisor. However, because it's built into the Linux kernel, which is a general-purpose operating system and can run other software besides virtual machines, it really sits somewhere between Type I and Type II. Linux can operate as a hypervisor in and of itself, so it can also run as a guest operating system inside any other hypervisor.

The risk with virtual machines is *escaping*. This is when an attacker compromises a virtual machine but then manages to escape the virtual machine to get control of the hypervisor. When the attacker has control of the hypervisor, he or she has control of all of the guest virtual machines. The goal of any attacker in this scenario is to compromise the hypervisor itself. This is why Type I hypervisors offer more protection, because the attack surface is so small. There is no general-purpose operating system to gain a foothold in from the outside other than potentially remote management of the hypervisor. On a Type II hypervisor, though, the host operating system and all of its services can be compromised.

Because a virtual machine can support a number of guest operating systems, however, deploying Linux inside a virtual machine has a number of advantages. One advantage is cost, as mentioned previously. Another is the ability to capture snapshots of a system image. These snapshots can be used very effectively on systems employed for testing. If something goes wrong, you can roll back to the snapshot to recover the system. Snapshots can also be used to provide the same environment over and over again for testing purposes. In addition, they are very helpful for disaster recovery.

Another advantage to virtual machines is ease of deployment. With a virtual machine, you can build the OS and create the system image and then store it. When you need a new system, you simply copy the image into place with a new name and start it up. With this approach, all your systems have the same configuration, making it much easier to apply updates and determine where the risks in your network and your organization are.

Cloud Services

Unless you've been living in a basement with no access to any media at all, you will have heard the term **cloud computing**. A simple definition of cloud computing is a computing service offered by a provider via the cloud, or Internet. This concept and term have been around for years, if not decades. Cloud computing's ease of use, cost savings, increased speed to deployment, and reliability are driving more people to its use. Although a number of types of cloud services are available from numerous providers, this section will focus on Infrastructure as a Service (IaaS) and Platform as a Service (PaaS).

FYI

Other services, like Software as a Service (SaaS) and Storage as a Service (STaaS) don't offer any exposure to the underlying operating system. Therefore, they are not relevant to the subject of Linux or securing Linux. When you buy access to an application in the cloud like Salesforce, you have no control over the underlying operating system. In fact, you can't even guarantee what operating system the application is running on. You are purchasing not an application stack, but rather a set of functionality. It's the job of the provider to give you that functionality in whatever manner makes the most cost-effective sense to them.

Infrastructure as a Service

With **Infrastructure as a Service (IaaS)**, you are buying computing time with an operating system that is hosted within a virtual environment by whichever provider you choose. Two popular ones are Amazon Web Services (AWS) and Google Compute Engine. The AWS Elastic Compute Cloud (EC2) gives you a way to create a new server instance in minutes with a variety of operating systems installed, depending on your needs. Amazon offers the choice between SUSE Linux, Red Hat Enterprise Linux, Ubuntu Linux, and its own version of Linux. Any of these choices will give you a Linux installation that runs on infrastructure managed by Amazon. You can select which geographic region the server will be installed in, which helps you place it as close to you or your customers as needed. Obviously, with the Internet, being geographically close isn't the same as being network close, but Amazon has enough handoffs with large network providers that geographically close should be similar to network close.

Google also offers the option of using its infrastructure and, like Amazon, has handoffs with all the large network providers. It offers many of the same distribution options as Amazon does. Google has its own geographical distribution offerings. Operating systems supported by Google include the following:

- CentOS 6.6
- CentOS 7
- CoreOS alpha
- CoreOS beta
- CoreOS stable
- openSUSE 13.1
- openSUSE 13.2
- Ubuntu 12.04
- Ubuntu 14.04
- Ubuntu 14.10

Both Amazon and Google provide remote access to their systems. With Linux, the preferred remote access method is generally Secure Shell (SSH). Both providers offer the means to generate keys that can be used to SSH into their servers once the instance is up and running. In the case of Amazon, a Web interface will generate a key that can be used with any SSH client.

Both Google and Amazon offer an **application programming interface (API)** to make it easier for developers to get access to all the functions available within their platforms. The APIs expose functions that programmers can call to interface programmatically with their cloud instance. To get the API, however, you must install a **software development kit (SDK)**. An SDK typically provides the libraries and headers necessary to build your own software that includes the functionality that uses the API. Both SDKs also include a command-line tool that can be used to communicate with the Google servers. As an example, you can use the gcloud program to connect remotely to your Google cloud instance over SSH. The first time you connect, it will generate the SSH keys you will need to make the connection. You can see the process in [Figure 3-2](#), including the final connection to the server.

```

portnoy:~ kilroy$ gcloud config set project crucial-module-88520
portnoy:~ kilroy$ gcloud compute ssh instance-1
For the following instances:
- [instance-1]
choose a zone:
[1] asia-east1-b
[2] asia-east1-c
[3] asia-east1-a
[4] europe-west1-c
[5] europe-west1-a (DEPRECATED)
[6] europe-west1-d
[7] europe-west1-b
[8] us-central1-a
[9] us-central1-c
[10] us-central1-b
[11] us-central1-f
Please enter your numeric choice: 8

WARNING: You do not have an SSH key for Google Compute Engine.
WARNING: [/usr/bin/ssh-keygen] will be executed to generate a key.
This tool needs to create the directory [/Users/kilroy/.ssh] before
being able to generate SSH keys.

Do you want to continue (Y/n)? y

Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /Users/kilroy/.ssh/google_compute_engine.
Your public key has been saved in /Users/kilroy/.ssh/google_compute_engine.pub.
The key fingerprint is:
c4:13:d2:f7:23:14:bb:ff:bf:09:2e:c8:9a:92:6d:6c kilroy@portnoy.local
The key's randomart image is:
+--[ RSA 2048]--+
|    ...   |
|    o..o   |
|    +o..   |
|    . ....o|
|    S ... .|
|        .   |
|     + . . ..|
|    o E.o .... .|
|    +o.    ...+o|
+-----+
Updated [https://www.googleapis.com/compute/v1/projects/crucial-module-88520].
Warning: Permanently added '108.59.87.55' (RSA) to the list of known hosts.
Identity added: /Users/kilroy/.ssh/google_compute_engine (/Users/kilroy/.ssh/google_compute_engine)
Warning: Permanently added '108.59.87.55' (RSA) to the list of known hosts.
Linux instance-1 3.16.0-0.bpo.4-amd64 #1 SMP Debian 3.16.7-ckt4-3~bpo70+1 (2015-02-12) x86_64

```

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent

FIGURE 3-2

Logging into a Google compute instance.

This is not to suggest, of course, that Amazon and Google are your only choices if you want an IaaS provider. A number of them are available. A quick search using your favorite search engine will turn up a lot of service providers that will provide you with a system all your own. It's been that way for years, of course, but they weren't always called IaaS providers, and the service wasn't always based on virtual servers to keep costs down.

In some cases, particularly with Linux systems, you received access to a shared system, where your access was jailed into a particular part of the overall system.

Platform as a Service

Another cloud option is **Platform as a Service (PaaS)**, which is a step up from IaaS. This may provide a complete Web application solution for easier application development. Depending on the service provider you chose, you may get access to the operating system, or it may just give you access to the application development features. Some examples of PaaS are [Salesforce.com's Force.com](#) and the Elastic Beanstalk for Amazon Web Services (AWS). Google also offers a PaaS called App Engine that sits on top of its Compute Engine.

Choosing a Boot Loader

After you have selected and installed your Linux distribution, wherever you have chosen to install it, the next step is to think about the boot loader and how you may best secure it. If Linux is your primary operating system, there are two primary boot loaders that typically come with a Linux distribution. (Of course, you can use other boot loaders to boot to a Linux system.) They are as follows:

- Linux Loader (LILO)
- Grand Unified Bootloader (GRUB)

Linux Loader

These days, the Linux Loader (LILO) is not nearly as popular as it used to be. One of the biggest problems with LILO is that it doesn't currently have complete support for GUID Partition Tables (GPTs) and universally unique identifier device (UUID) identifiers. These are far more common with newer hardware because of the size of the drives found on these devices. The GPT, compared to the master boot record (MBR), supports far more and larger partitions. The LILO boot loader can be configured to require a password before the operating system loads. The following is an example of the LILO configuration file, commonly found at /etc/lilo.conf.

```
default="Linux"
boot=/dev/hda
map=/boot/map
prompt
timeout=100
message=/boot/message
image=/boot/vmlinuz
    password=MyPassw0rd
    label="Linux"
    root=/dev/hda1
    initrd=/boot/initrd.img
    append="acpi=ht splash=verbose"
    vga=788
    read-only
```

The **password=** line under the image specification gives the password that must be entered before Linux will boot. This password can also be specified at the top level of the configuration file. If you were booting multiple operating systems, like dual-booting Windows and Linux, you could require the password before any operating system booted. If you wanted to require a password across all operating systems, you would put the **password=** line in the top-level configuration. Anything above the **image=** lines would result in requiring the password no matter which operating system was selected.

Grand Unified Boot Loader

At this time, most distributions use the Grand Unified Bootloader (GRUB). GRUB offers some advantages over LILO, including the ability to alter the boot parameters at boot time by editing the settings in the boot menu live. You also don't have to reinstall the boot loader into the master boot record when you make a change to the configuration file. Instead, GRUB will read the configuration file as part of the boot process, determining where it needs to look next for the final stage of the boot—loading the kernel. GRUB will also allow you to use a password to protect the boot process. This is another area where GRUB has an advantage over LILO. With LILO, anyone with access to the system while it is running can get the boot password. GRUB makes it a bit harder by protecting the password using a cryptographic hash. You can implement the password using a message digest 5 (MD5) hash. As an example, this is a line you can put into your GRUB configuration file:

```
title linux lock password -md5 90faa7a45e286dd869945be21a802f89
```

FYI

A *hash* is a complex mathematical function that generates a value of a specific length no matter what the input length is. This is not encryption, but because hash values are commonly used to provide verification for encryption functions, they are closely related. A hash function should theoretically not generate the same value twice given two separate input sources.

That line will add a password to the operating system named `linux` to ensure that only people who have the password can complete the boot process. You'll notice that the password is not in cleartext. Instead, it is an MD5 hash of the password. This can also be broken, of course, but it takes some effort. Depending on the password used, this can be time consuming. If it's a long and complex enough password, it might require a brute-force attack rather than a **dictionary attack**. With a brute-force attack, all possible passwords have to be checked, one at a time. With a dictionary attack, a set of possible passwords is checked using a word list or dictionary. Any password not in the word list is not checked. Another option similar to a dictionary attack is **rainbow tables**. With rainbow tables, instead of computing the hash value of each password in the dictionary whenever the password cracker is running, all the hashes are already computed and the password hash is compared with the precomputed hash in the rainbow table.

GRUB has a problem similar to LILO, however, because it doesn't support GPT disks. If you want to use GPT disks, you have to use GRUB2. Some distributions, like Red Hat Enterprise Linux, are moving to using GRUB2. However, GRUB2 uses a completely different configuration system. Instead of manually editing a single configuration file, GRUB2 uses a set of configuration files. It then uses a utility, `grub2-mkconfig`, to build the final configuration that is used. You can set a password on a specific operating system or you can set a global password on the configuration. With GRUB2, you have to set a superuser in the configuration. After you have set the superuser, you can set passwords. A configuration fragment that had a password in it would look like the following.

```
set superusers="myuser"
password myuser MyPassword123
```

You'll notice that the password is in plaintext. You can also create a hashed password using GRUB2. You can create the hashed password using the utility `grub2-mkpasswd-pbkdf2`, which will generate a line that you can use in your configuration file. This uses the Secure Hash Algorithm (SHA) for the hash rather than the MD5 hash algorithm. The hash generated from the utility is as follows:

```
[kilroy@localhost ~]$ grub2-mkpasswd-pbkdf2
Enter password:
Reenter password:
```

PBKDF2 hash of your password is grub.pbkdf2.sha512.10000.834CCAR356A
↳ 9FE65C063263D81BAEE6206D2F7B636DAFC829A0C3943440FE08E883A229FA
↳ 871F4317BCA53D69CAA14776849CEF554FD296636F59BBD5EDFA1.F182ED50
↳ D3A01FED9C641ECC5739EBA6CFF97ABA70E7592ACCD6F8896569229D23C876
↳ 5C908CBB56B093CDF48725BEDE0CD496AD48CEAFBA3A5F218A32F2C700

After the system has booted up and the kernel has loaded into memory, control transfers to the initialization system. On many Linux distributions, this is the init program, which was the process inherited from Unix. The init program becomes the super parent process for every program that ever runs on the system. Every service that starts is started by the init process. The GUI is started by the init process. Everything that ever happens on the system through its running life can be traced back to the init process. Some Linux distributions, like Red Hat Enterprise Linux, have migrated to a newer initialization system called systemd. As with init, systemd becomes the process to which all others trace back. If you were to print out a process tree, as in [Figure 3-3](#), you will see that the top-level process is systemd from a RHEL system. Other processes have children of their own, but the super parent is the systemd process.



NOTE

One of the advantages of systemd is that, unlike with init, services can be started up in parallel. While there have been other attempts to make processes start up in parallel to help the system boot faster, systemd is the latest example. This is not just an upgrade on init. It's a completely different way for the system to start up that uses different utilities, configuration files, and management techniques.

Services

One thing init and systemd do is start up all the services. **Services** are autonomous processes that generally start up at boot time and run in the background while the operating system is running and operational. While it's easy to think of services as being programs that listen for network connections, there are a number of other services in constant operation as well. In many Linux distributions—though not all—you can find all the service startup scripts in `/etc/init.d/`. That will tell you the services that you can start up on the system. [Figure 3-4](#) shows the listing of that directory on a fairly standard install of Linux Mint, which is pretty similar to an Ubuntu installation.



FIGURE 3-3

Process tree.

```
kilroy@hodgepodge ~ ls /etc/init.d/
acpid           friendly-recovery    pppd-dns      sendsigs
anacron          grub-common        prltoolsd     single
avahi-daemon    halt              prl-x11       skeleton
binfmt-support  hddtemp          procps        smbd
bluetooth       irqbalance       pulseaudio    speech-dispatcher
brltty          kerneloops       rc            sudo
casper          killprocs        rc.local      udev
console-setup   kmod             rcS           umountfs
cpufrequtils   lm-sensors       README        umountnfs.sh
cron            loadcpufreq      reboot        umountroot
cryptdisks     mdm             resolvconf   unattended-upgrades
cryptdisks-early mintsystem      rsync         urandom
cups            networking       syslog        virtualbox-guest-utils
cups-browsed   nmbd           samba         virtualbox-guest-x11
dbus            ondemand        samba-ad-dc  x11-common
dns-clean       postfix        saned
```

FIGURE 3-4

Service startup scripts in the `/etc/init.d/` directory on a fairly standard install of Linux Mint.

There are utilities that will provide you the list of services as well. One of these is the `service` command. If you run the `service --status-all` command, you will see a list of all the services on the system as well as the status of each. Running `service` on the same Linux Mint system as the one from which the preceding directory listing came generates the following output. Most of the scripts in `init.d` enable you to check the status of the service. The `Service` command will get the status from each of the services. If the service is running, `service` will output `+`. If it isn't running, `service` will output `-`. If the status check either doesn't exist or didn't return anything, `service` responds with `?`.

```
[ + ] acpid
[ - ] anacron
[ + ] avahi-daemon
[ ? ] binfmt-support
[ + ] bluetooth
[ - ] brltty
[ - ] casper
[ ? ] console-setup
[ ? ] cpufrequtils
[ + ] cron
[ ? ] cryptdisks
[ ? ] cryptdisks-early
[ + ] cups
[ + ] cups-browsed
[ - ] dbus
[ ? ] dns-clean
[ + ] friendly-recovery
[ - ] grub-common
```

```
[ - ] hddtemp
[ ? ] irqbalance
[ + ] kerneloops
[ ? ] killprocs
[ ? ] kmod
[ - ] lm-sensors
[ ? ] loadcpufreq
[ + ] mdm
[ ? ] mintsystem
[ ? ] networking
[ + ] nmbd
[ ? ] ondemand
[ - ] postfix
[ ? ] pppd-dns
[ ? ] prl-x11
[ ? ] prltoolsd
[ - ] procps
[ - ] pulseaudio
[ ? ] rc.local
[ + ] resolvconf
[ - ] rsync
[ + ] rsyslog
[ + ] samba
[ - ] samba-ad-dc
[ + ] saned
[ ? ] sendsigs
[ + ] smbd
[ ? ] speech-dispatcher
[ - ] sudo
[ - ] udev
[ ? ] umountfs
[ ? ] umountnfs.sh
[ ? ] umountroot
[ - ] unattended-upgrades
[ - ] urandom
[ - ] virtualbox-guest-utils
[ ? ] virtualbox-guest-x11
[ - ] x11-common
```

One of the goals of any hardened installation is to limit the number of services running as well as the number of services installed. Even if a service isn't running, it can be started if it's installed. A number of vulnerabilities may allow an attacker to execute a program that's on the system. It's harder to get the package installed and then get the service started. Obviously, if an attacker gets login access or just shell access, almost anything becomes possible. The goal, though, is to reduce the surface area as much as possible to limit potential attacks. The **service** command enables an administrator to manipulate services. You can stop and start services using the **service** command if you know the name of the service, which you can get from the **--status-all** function. As an example, the following starts the Samba service:

```
kilroy@hodgepodge ~ $ sudo service smbd start
smbd start/running, process 759
```

The script that manages the Samba service responds with the process identifier from the new process that was launched by the `service` script. Of course, you can accomplish the same result by executing the script directly: `-- /etc/init.d/smbd start`. This is the old Sys V init style. Other systems use different mechanisms, including, as noted, the new systemd style Red Hat uses. You can't access the list of services in the same way as you did before, nor can you start services up the same way. Additionally, the definition of services has been expanded. You can see a partial listing of the services to get an idea of this expansion in [Figure 3-5](#). With systemd, device drivers also fall into the category of services that can be managed. You can see the list of devices that are operating at the top of the list. Further down, although not shown in this figure, are the different services like CUPS, Samba, Apache, and other more traditional services.

Runlevels

The init program uses a series of *runlevels* to determine which services it may run when the system starts up. A runlevel provides information to the init program about what it should be doing as it starts the system up. Each runlevel introduces a different set of functionality. Different systems may decide what different runlevels mean. [Table 3-1](#) outlines what each runlevel means to most Linux systems.

UNIT	LOAD	ACTIVE	SUB	DESCRIPTION
proc-sys-fs-binfmt_misc.automount	loaded	active	waiting	Arbitrary Executable File Formats File S
sys-devices-p...-virtio0-net-eth0.device	loaded	active	plugged	Virtio network device
sys-devices-p...:0-block-sda-sda1.device	loaded	active	plugged	CentOS_Linux-0
sys-devices-p...:0-block-sda-sda2.device	loaded	active	plugged	LVM PV 4e6kve-HTF2-90io-s12P-2uHf-rP8r-6
sys-devices-p...:2:0:0:0-block-sda.device	loaded	active	plugged	CentOS_Linux-0
sys-devices-p...:3:0:0:0-block-sr0.device	loaded	active	plugged	Virtual_DVD-ROM_1
sys-devices-p...:1f.4-sound-card0.device	loaded	active	plugged	82801BA/BAM AC'97 Audio Controller
sys-devices-p...:serial8250-tty-ttyS0.device	loaded	active	plugged	/sys/devices/platform/serial8250/tty/tty
sys-devices-p...:serial8250-tty-ttyS1.device	loaded	active	plugged	/sys/devices/platform/serial8250/tty/tty
sys-devices-p...:serial8250-tty-ttyS2.device	loaded	active	plugged	/sys/devices/platform/serial8250/tty/tty
sys-devices-p...:serial8250-tty-ttyS3.device	loaded	active	plugged	/sys/devices/platform/serial8250/tty/tty
sys-devices-virtual-block-dm\x2d0.device	loaded	active	plugged	/sys/devices/virtual/block/dm-0
sys-devices-virtual-block-dm\x2d1.device	loaded	active	plugged	/sys/devices/virtual/block/dm-1
sys-devices-virtual-block-dm\x2d2.device	loaded	active	plugged	/sys/devices/virtual/block/dm-2
sys-module-configfs.device	loaded	active	plugged	/sys/module/configfs
sys-module-fuse.device	loaded	active	plugged	/sys/module/fuse
sys-subsystem-net-devices-eth0.device	loaded	active	plugged	Virtio network device
-mount	loaded	active	mounted	/
boot.mount	loaded	active	mounted	/boot
dev-hugepages.mount	loaded	active	mounted	Huge Pages File System
dev-mqueue.mount	loaded	active	mounted	POSIX Message Queue File System

FIGURE 3-5

Partial output from the `service` command.

TABLE 3-1 Runlevel definitions.

RUNLEVEL DEFINITION	
0	System powered off. There is no system activity at this runlevel.
1	Single user. This can be used to make changes to a system without having multiple users logged in.
2	Multiple users, no Network File System (NFS). This may also mean no networking.
3	Multiple users, networking functionality. This may be a default runlevel on servers.
4	Definable.

- | | |
|---|---|
| 5 | Multiple users with a GUI. Desktops may use this as a default runlevel. |
| 6 | Reboot. |

You can set the runlevel by using the `init` command and passing the runlevel to it. To reboot, for example, you could use `reboot`, use `shutdown`, or just type `init 6` as a root user. To halt the system, you could use `init 0` to set the runlevel to 0 and shut the system down. Each system has a default runlevel that may be set in the `/etc/inittab` file, although different distributions may use different files to set the default runlevel.

Different Linux distributions also use a different directory structure to handle runlevels. The common structure, based on AT&T's Sys V, is `/etc/rcn.d` where the `n` is the runlevel. Inside of each of these directories—`/etc/rc0.d`, `/etc/rc1.d`, `/etc/rc2.d`, and so forth—are links to startup scripts. Each entry in the directory is likely a symbolic link to the actual startup script in `/etc/init.d`. The links have different names, depending on what they are expected to do. Because there is no parallel startup and no way in the common init startup structure to specify dependencies, the naming makes a big difference. Service startup scripts have names that begin with an S followed by a two-digit number. The number is important because it specifies the order in which the service will start up. The operating system will read through all the files beginning with an S in ASCII order. So, S00 will come before S01, then S10, and so forth. This enables you to ensure that one service starts before another if there is a dependency where one relies on the other to start up cleanly. The following entries show the numbered scripts.

```
kilroy@hodgepodge ~$ ls /etc/rc3.d
README           S20rsync          S99grub-common
S05loadcpufreq   S20speech-dispatcher  S99ondemand
S19cpufrequtils  S20virtualbox-guest-utils  S99prltoolsd
S20hddtemp       S50saned
S20kerneloops   S70dns-clean
S20postfix       S70pppd-dns
kilroy@hodgepodge ~$ ls /etc/rc5.d
README           S20rsync          S99grub-common
S05loadcpufreq   S20speech-dispatcher  S99ondemand
S19cpufrequtils  S20virtualbox-guest-utils  S99prltoolsd
S20hddtemp       S50saned
S20kerneloops   S70dns-clean
S20postfix       S70pppd-dns
```

Notice the scripts that start with a K. These scripts stop the services. The K is for kill, since the S for stop was already in use. Each script is aware of how it has been called, so it knows whether to launch the start or stop function it contains. Again, these scripts will be called in ASCII order, so the numbers make a difference if one service needs to be stopped before another to prevent errors or failures. Checking the different runlevel directories is another way to determine the services that are running in that particular runlevel.

Wrappers

Services may not have built-in abilities to restrict network access. Because of that, Wietse Venema wrote the **TCP Wrapper** program. TCP Wrapper intercepts the network traffic destined for the network service behind the **wrapper**. A wrapper wraps existing service programs, offering them network-level protections. The wrapper makes a determination based on the source IP address as to whether the connection attempt will be allowed to pass through to the network service behind.

These access controls occur at the Application Layer but are based on the Network Layer. This is different from a traditional firewall, which operates much lower in the networking stack. The entire network communication has been parsed by the time it gets to the TCP Wrapper program. This means the lower layers of the network communication have been pulled off of the frame that arrived at the network interface. The source and destination IP addresses have been parsed and the destination port has been determined to identify which application the information should be passed to.

TCP Wrapper uses a pair of files in the /etc directory to make determinations. Not surprisingly, hosts that are allowed to communicate with the system are entered in the /etc/hosts.allow file. The second file, /etc/hosts.deny, is a list of all the addresses that are not allowed to communicate. Between these two files, you end up with a set of rules that control access to the network service that sits behind the wrapper.

inetd and xinetd

Network services that are typically started up by init are not the only network services that will run on a system. There are others that rely on another service to start up. The super server, **inetd**, has always been responsible for starting up what are sometimes called simple services. This might be the character generator (chargen) or the echo service. It may also be more complex services like an FTP server. Instead of each of the programs being responsible for these services running all the time, the super server runs all the time and then launches each of these other servers only as they are needed as network connections come into the appropriate port.

The inetd super server has been around for a long time to manage these services. However, it had some security issues. For one, it didn't offer good ways to restrict who could get access to some of these services. In addition, some of the services could be exploited. The chargen and echo services, for example, can easily be exploited to generate traffic to other systems.

An upgraded inetd called **xinetd** was developed to provide better functionality and more control over the services. With xinetd, you have a configuration file for each service, defining whether it is enabled and other configuration settings. A sample configuration file follows:

```

service chargen
{
    disable          = yes
    type             = INTERNAL
    id               = chargen-dgram
    socket_type     = dgram
    protocol        = udp
    user             = root
    wait             = yes
}

```

You can specify access control lists using xinetd by creating both **whitelists** and **blacklists**. A whitelist is a list of things you want to allow. In this case, it's a list of addresses or systems that you want to allow to get access to your system. A blacklist is, by contrast, a list of things you *don't* want to allow, as in addresses or systems. The easiest thing to do is to create a whitelist allowing only the systems that you want to allow and denying all other systems from getting access by default. It's harder to create a blacklist because the list of things you don't want to allow is commonly a lot longer than the list of things you do want to allow.

technical TIP

Whitelists and blacklists are used across security mechanisms, so this won't be the last time you run across them. They are very important concepts.

To specify a list of systems from which you want to allow access, you use the **allow_from** key in the configuration file. After **allow_from**, you add your whitelist of addresses that you want to allow access to the service. You can create a blacklist using the **no_access** key. It works the same way as the **allow_from** setting. You create a list of values with spaces between them including hostnames or IP addresses from which you want to deny access. In some cases, you might discover you want to allow access from everywhere, but eventually you run across someone who is abusing your system. This is where you may want to use a blacklist.

R-services

One of the major advantages of networked systems, especially from a business perspective, is the ability to share information and computing power. Unix-like systems have long had a set of services that allow users to gain access, copy files, and perform other tasks on remote systems. These are commonly called *r-services* because all of the programs and protocols start with an *r*, for *remote*. If a user wanted to run a command on a remote server, for example, he or she might use rsh. If the user wanted to copy files, he or she could use rcp. To ensure files are the same on one system as another, as in the case of backing up one system to another, the user could use rsync.

The r-services are managed by inetd and xinetd. They aren't network **daemons**—standalone, always-running programs that are responsible for the application functionality and handle connections for those services. Instead, the super server is used to handle the connections and determine whether systems can connect or not based on the inetd or xinetd configuration.

These services have long been considered to be unsecure because authentication for them is done at the host rather than at the user. The file /etc/hosts.equiv provides a list of hosts that are allowed to use the r-services on one system. When a system has been given equivalency permissions, anyone on that system can connect to the r-services and use them. No further authentication is necessary—no passwords and no keys. Additionally, these services are plaintext, meaning that any information that is passed back and forth, like files and commands, will be in the clear. While it has long been considered bad practice to enable these services, there are still

organizations that use them, in part due to legacy applications. After you have invested years in an application that works well for you, it can be difficult to make a case to redo the application just because the services it uses are weak and unsecure.



CHAPTER SUMMARY

It can't be said enough that security is a process. You have to think holistically to provide the best protection for your systems. This means starting by selecting the right Linux distribution and the best delivery platform for you. Of course, these choices are based on more factors than just security. Cost is a major factor, which often leads businesses to choose virtualization and cloud computing. Any time you move your corporate data outside your direct control, however, there are issues around who can access your sensitive information. You must weigh security and cost benefits against each other to make the best decision about whether to use a local physical system, a set of virtual machines, or a cloud-based solution to host your system and all the applications and data that will be stored on it.

After you have selected the delivery platform, you must start thinking about physical security controls, including whether the system is so well protected that you don't need a password just to boot it. In some cases, you may wish to require a password before booting. Other times, you may believe that your system is adequately protected through keys, cards, and other physical authentication mechanisms, eliminating the need for a password to protect the boot process. Every system you install, though, will need to be hardened, meaning that you will have to determine the least number of packages and services that are installed. You'll need to know how to determine what services are installed and how to control those services.

There are a couple ways to deliver network services. One is to use daemons, which are standalone programs that continuously run in the background. Another is to use a super server that will launch the required program as soon as a request for it comes in. No matter what you do, you may want to use some Application Layer controls to protect your network services. This can be done through the use of wrappers for standalone services or through specific configuration settings if you are using the inetd or xinetd super server.



KEY CONCEPTS AND TERMS

Application programming interface (API)

Attack canvas

Attack surface

Blacklists

Brute-force attack

Cloud computing

Crack

Daemons

Dictionary attack

inetd

Infrastructure as a Service (IaaS)

Packages

Password cracker

Platform as a Service (PaaS)

Rainbow tables

Services**Software development kit (SDK)****TCP Wrapper****Threat vectors****Translation lookaside buffer (TLB)****Virtual machines****Virtual memory****Whitelists****Wikis****Wrapper****xinetd**

CHAPTER 3 ASSESSMENT

- 1.** Which package manager does Red Hat Enterprise Linux use?
 - A. apt
 - B. SlackPack
 - C. rpm
 - D. tgz

- 2.** In which directory can network services commonly be found?
 - A. /etc/netserv/
 - B. /etc/init.d/
 - C. /etc/rc5/
 - D. /srv/init.d/

- 3.** Which of the following is the first process that runs after the kernel loads on most Linux systems?
 - A. init
 - B. inetd
 - C. xinetd
 - D. superserv

- 4.** Which of the following is the super server that launches applications based on connection attempts?
 - A. init
 - B. superserv
 - C. xinetd
 - D. init.d

- 5.** Ubuntu and Linux Mint derive from which of the following Linux distributions?
 - A. Red Hat
 - B. Xubuntu
 - C. Gentoo
 - D. Debian

- 6.** What kind of attack is it when you attempt every possible character combination to crack a password?
 - A. Brute force
 - B. Dictionary
 - C. Wordlist

- D. Iterative
- 7.** Runlevel 5 is commonly used for what purpose?
- A. Single user
 - B. Multiuser, no network
 - C. Multiuser, GUI
 - D. Multiuser, network, no GUI
- 8.** Which of the following is the piece of software used to manage virtual machines?
- A. Super server
 - B. Hypervisor
 - C. Central processor
 - D. Inetserv
- 9.** Which setting would you use to give certain systems access to a network service through the super server?
- A. `allow_from`
 - B. `open`
 - C. `acl_allow`
 - D. `open_from`
- 10.** What would you use to provide access control at the network level without using the super server?
- A. Network libraries
 - B. UDP ACLs
 - C. TCP Wrapper
 - D. TCP ACLs
- 11.** Given four services named S01service4, S02service3, S99Service2, and S10service1, which service would start first?
- A. S02service3
 - B. S99service2
 - C. S01service4
 - D. S10service1
- 12.** Which of the following describes a service that starts up and stays running in the background?
- A. Super server
 - B. init
 - C. systemd
 - D. Daemon

CHAPTER

4 User Privileges and Permissions

AFTER A SYSTEM IS BOOTTED AND RUNNING, it lives to serve users. Linux systems are designed to have multiple users. This may mean that multiple users could be logged in at the same time, both on the system directly as well as remotely. You may have only a single user for the system if it's a desktop, but there are probably quite a few additional users added by certain software packages to handle the services that are running. The thing about users, though, is that you probably want to be able to protect one from another. Everyone has secrets, after all, and sometimes you just want to keep someone from inadvertently altering files that belong to someone else. With Linux, permissions are based on users and groups. Although it seems fairly simplistic compared to more complex access control lists, there are still various ways to control access to files on the system. User information is stored in the **shadow password suite**. The files of the shadow password suite make up the local Linux password authentication database. Several of these files are worth looking at in more detail.

Of course, some users are more trusted than others. Those users may be given a variety of administrative privileges. Linux has logging services that allow you to track who logs in—and who tries and fails to log in—to monitored systems. The pluggable authentication modules (PAMs) offer a number of ways for users to be authenticated on a Linux system. PAMs allow users not only to be authenticated with local password stores, but also by way of network authentication—using facilities like **Network Information Service (NIS)** and the **Lightweight Directory Access Protocol (LDAP)**. (Both NIS and LDAP are directory services for network-based authentication.) LDAP PAMs also allow implementation of password policies like password length and complexity. **Polkit** (formerly PolicyKit) allows for systemwide control of permissions.

Chapter 4 Topics

This chapter covers the following topics and concepts:

- What the shadow password suite is
- Which user privileges are available
- How to secure groups of users
- How to configure the hierarchy of administrative privileges
- What regular and special permissions are
- How authorized and unauthorized access can be tracked through logs
- What the essentials of pluggable authentication modules (PAMs) are
- How the polkit can authorize access
- Which tools exist for network authentication
- What best practices are for user privileges and permissions

Chapter 4 Goals

When you complete this chapter, you will be able to:

- Configure regular and special user privileges and permissions
- Set up users with varying levels of administrative rights
- Use fine-grained administrative controls including PAMs and the polkit
- Understand options for network authentication

The Shadow Password Suite

For more than 20 years, Unix systems used the `/etc/passwd` file as a way to store both users and their associated passwords. To that point, most users on multiuser systems were trusted users, so it didn't matter so much that the `passwd` file was available to all users. It had to be, given the number of utilities that needed read access to the `passwd` file to acquire information about the system's users.



NOTE

While the Message Digest 5 (MD5) cryptographic hash function used by the shadow password suite is often used as part of a cryptographic system, it is not encryption. MD5 provides a one-way value, meaning if you have the hash value, you can't restore the original value from it.

As more systems became attached to networks and more users who may not have been strictly trusted began using these systems, it became necessary to separate the passwords out to a separate file. Even when encrypted, it's not a good practice to store passwords in a world-readable file. Additional restrictions on passwords can also enhance security. Those are some of the reasons for the development of the shadow password suite, which add the `/etc/shadow` and `/etc/gshadow` files to the database. The `gshadow` file is the file that may store a hashed password for a group. The file that stores all the public information about the system's groups is `/etc/group`.

The files of the shadow password suite are fundamental to Linux. You may have read about them before in other texts. Understanding the use of all these files is important. A number of system utilities use them, including the PAM system and the login mechanism. The `passwd` and `group` files, in particular, are also critical for giving permissions to all the files on the system.



NOTE

Technically, the shadow password suite as originally developed also includes the current functionality of the `login`, `passwd`, and `su` commands. (The `su` command can connect with the privileges of another user.)

`/etc/passwd`

The `/etc/passwd` file contains basic information for each user account. It defines the users configured on the local system. It may seem a bit misnamed, however. As you can see from most current versions of this file, it no longer contains even a salted hashed version of the actual password. Even so, it is the starting point for individual user accounts. A cryptographic hash is used to generate a fixed-length value that can't be restored to the original value. The salt is a value that is used to introduce some randomness into the function to help protect against a dictionary attack where a list of words is similarly hashed to compare the resulting hash value with the stored value. If the two values match, the password can be determined.

Each `/etc/passwd` file contains seven columns of information, delineated by colons. [Table 4-1](#) describes these columns from left to right.

Many accounts in the user-authentication database include a standard login shell in the last column, such as /bin/bash or /bin/sh. A malicious user who somehow gets access to that account would get access to a regular shell equipped with a full array of commands.

Changing the default shell for these nonstandard users can enhance security. Two fake shells available for this purpose are /bin/false and /sbin/nologin. In some distributions, the nologin shell is found in the /usr/sbin/ directory.

/etc/group

The /etc/group file contains basic information for each group account. It defines the groups configured on the local system. It's a simple file that includes user members of each group. Users who are members of certain groups may have special privileges or may be part of a common project. They may have privileges to certain hardware components. They may have some level of administrative privileges. Several of these groups are described later in this chapter. Each /etc/group file contains four columns of information, delineated by colons. [Table 4-2](#) describes these columns from left to right.

TABLE 4-1 Data in /etc/passwd, by column.

COLUMN	DESCRIPTION
Username	Login name.
Password	May be set to x or *. An x in a standard password column refers to /etc/shadow for the actual password. An * is shown if the account is disabled.
User ID	A numeric identifier for the user.
Group ID	A numeric identifier for the primary group of the user.
User information	Comments for the user. Sometimes known as the <i>GECOS field</i> , based on its development as the General Electric Comprehensive Operating System.
Home directory	The directory accessed when logging into the given account.
Login shell	The start shell for the user.

TABLE 4-2 Data in /etc/group, by column.

COLUMN	DESCRIPTION
Groupname	Most Linux systems are configured with users as members of their own group. Other groups may be administrative and more.
Password	Normally set to x, which refers to /etc/shadow for any password.
Group ID	A numeric identifier for the group.
Group members	A comma-delineated list of users who are members of the group.

/etc/shadow

User access to the /etc/shadow file is limited to the root administrative user. Even then, default access to the administrative user is limited to read-only. In some cases, access may be allowed to users who are members of the shadow group. It adds detailed password information to users defined in the /etc/passwd file. Per the shadow password suite, that password is modified by a salted hash, normally using the MD5 algorithm. An /etc/shadow file contains eight columns of information, delineated by colons. [Table 4-3](#) describes these columns from left to right.



NOTE

Because the MD5 algorithm is generally considered to be weak and potentially breakable, more and more systems are using the Secure Hash Algorithm (SHA1), which generates a longer hash value.

TABLE 4-3 Data in /etc/shadow, by column.

COLUMN	DESCRIPTION
Username	Login name.
Password	A password that has normally been changed to a salted hash. If it starts with \$1\$, it's been modified with the MD5 algorithm.
Date of last password change	Date of last password change, in number of days after January 1, 1970.
Minimum password life	Number of days a password must be retained.
Maximum password life	Number of days the same password can be retained.
Password warning period	Number of days before password expiration when a warning is given.
Inactive period	Number of days after password expiration when an account is made inactive.
Disabled period	Number of days after password expiration when an account is disabled.

The password-related options in /etc/shadow are especially important. If an unauthorized user breaks into a system and manages to get a copy of /etc/shadow, it'll take a little time for that person to decipher those salted hashed passwords. If you change passwords before the unauthorized user has time to use a rainbow table, your systems will be that much more secure.



Because rainbow tables can be used to decipher hashed passwords, the standard MD5 hash includes a salt. A rainbow table is a set of precomputed stored hashes that are mapped to the plaintext password. A **salt** is a 56-bit key added to the hash to make it more difficult to set up a manageable rainbow table. The salt adds randomness into the password creation. Because rainbow tables can't easily take into account every possible salt, a salt is one way of defeating rainbow-table attacks.

TABLE 4-4 Data in /etc/gshadow, by column.

COLUMN	DESCRIPTION
Groupname	Most Linux systems are configured with users as members of their own group. Other groups may be administrative and more.
Password	Normally set to ! if there's no password.
Group administrator	A user who is authorized to add users to the group.
Group members	A comma-delineated list of users who are members of the group.

/etc/gshadow

User access to the /etc/gshadow file is generally limited to the root administrative user. On some distributions, access may be allowed to users who are members of the shadow group. The /etc/gshadow file is analogous to the /etc/shadow file except it's for groups. Group administrators and an associated salted hashed password may be added to this file. The difference between /etc/shadow and /etc/gshadow is that with /etc/gshadow, passwords are rarely configured; when they are, /etc/gshadow doesn't specify password lifetime parameters.

Each /etc/gshadow file contains four columns of information, delineated by colons. [Table 4-4](#) describes these columns from left to right.

Passwords in /etc/gshadow can be set with the **gpasswd** command. A group administrator is not required. For example, the following command allows you to set the password for the project group, which can be used to authenticate the **newgrp** or **sg** commands. (The **sg** command can connect with the privileges of another group.)

```
# gpasswd project
```

Defaults for the Shadow Password Suite

When you create new users and groups, settings in the /etc/login.defs file are used to generate some of the values, including specifying defaults. If you're interested in more rigorous security, you may consider setting or changing the configuration directives shown in [Table 4-5](#).

A number of directives associated with PAMs, described later in this chapter, have superseded a number of directives traditionally used in /etc/login.defs. For newly created users, the contents of the /etc/skel/ directory are copied to that user's home directory with appropriate permissions.

TABLE 4-5 /etc/login.defs security-related directives for new users and groups.

DIRECTIVE	DESCRIPTION
FAILLOG_ENAB	Failed login attempts are collected in the binary /var/log/faillog file.
LOG_OK_LOGINS	Successful logins are collected in a log file defined by /etc/syslog.conf.
SYSLOG_SU_ENAB	Uses of the su command are logged.
SYSLOG_SG_ENAB	Uses of the sg command are logged.
FTMP_FILE	Login failures collected in an associated file.
PASS_MAX_DAYS	Maximum number of days a password can be used.
PASS_MIN_DAYS	Minimum number of days a password must be retained.

PASS_MIN_LENGTH	Minimum password length.
LOGIN_TIMEOUT	Maximum time for a console login.

Privileged and Unprivileged User and Group ID Numbers

Every user and group has a **user ID (UID)** and **group ID (GID)** number. This is the number associated with the username or group name in Linux, respectively. In 2003, the number of available UIDs and GIDs was raised from a 16-bit number to a 32-bit number. In other words, the highest legal UID and GID numbers are somewhere over 4 billion. That should be more UID and GID numbers than are needed in any modern enterprise. UIDs and GIDs are further limited, however. The `/etc/login.defs` file defines standards for those numbers with the **UID_MIN** and **GID_MIN** directives. Depending on the distribution, the minimum UID and GID number for a regular user or group may be 100, 500, 1,000, or some other relatively low number. UIDs and GIDs below that minimum are reserved for administrative, service, and other privileged accounts.

You'll also find maximum UIDs and GIDs defined in `/etc/login.defs`. These are arbitrary numbers. UIDs and GIDs above that number are often used by other authentication systems such as NIS, LDAP, and Microsoft authentication through Samba.

Some special high UID numbers are associated with fewer privileges than a regular user has. Two prime examples are the users nobody and nfsnobody. They may be assigned the second-to-last possible 16- or 32-bit UID. For example, the nobody account on an Ubuntu system may be assigned a UID number of 65534, the second-to-last possible 16-bit UID. In contrast, the nfsnobody account on a Red Hat system may have a UID number of 4294967294, the second-to-last possible 32-bit UID. These accounts have matching GID numbers.

While accounts with such UID and GID numbers are supposed to have fewer privileges than regular users, that's documented only in the information in the user-authentication database.

TABLE 4-6 Standard user- and group-management commands.

COMMAND DESCRIPTION	
useradd	Adds users to the shadow password suite based on defaults in <code>/etc/login.defs</code> except when modified by <code>useradd</code> command options.
usermod	Modifies user settings in the shadow password suite.
userdel	Deletes users. By itself, the command retains the user home directory.
groupadd	Creates a new group.
groupmod	Modifies group information.
groupdel	Deletes an existing group.
groups	Lists group membership of the current user.
chage	Revises aging information for a user's password.

Shadow Password Suite Commands

A variety of commands and tools are available to create, modify, and delete users and groups. Generally, the commands shown in Table 4-6 are well covered in more elementary Linux texts. As a security administrator, be

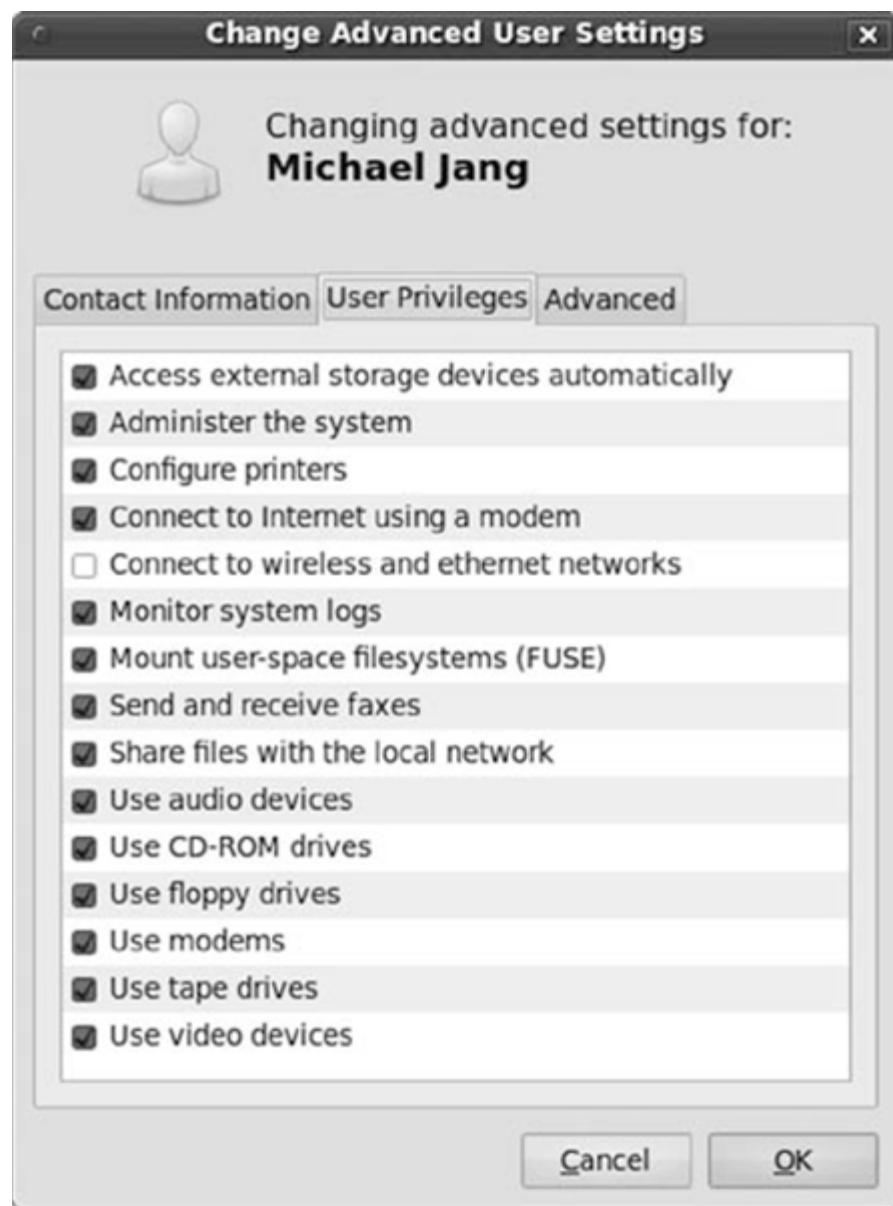
prepared to use these commands frequently to make sure users and groups are created with appropriate levels of security.

Although many excellent GUI user- and group-management tools are available, as a Linux administrator, you need to be prepared to administer users and groups from the command line.

Available User Privileges

Although the privileges discussed in this section are partially administrative in nature, they relate to privileges that users may want on regular workstations. The permissions in question support access to hardware such as modems, sound cards, printers, and scanners.

On Ubuntu distributions, user privileges to special hardware are implemented through group memberships. While those groups are listed in /etc/group, a clearer description is available in the Ubuntu Users Settings tool, accessible in the GUI with the **users-admin** command. Depending on the release, you'll click Properties or Advanced Settings and then select the User Privileges tab. The window that appears is shown in [Figure 4-1](#). You'll see a description of some of the groups associated with special user privileges.

**FIGURE 4-1**

User privileges as a member of special groups

If you select one of the options shown in Figure 4-1, the Users Settings tool applies the **usermod** command to make the subject user a member of the noted group. Of course, as a Linux administrator, you may choose to apply the **usermod** command more directly. Alternatively, you can add desired users to the noted groups even more directly by editing the /etc/group and /etc/gshadow files. After all, these are text files and can be modified in any text editor if you have appropriate permissions to change them.

Table 4-7 may help you identify groups associated with the descriptions shown in Figure 4-1. The groups in the table are the group names found in the /etc/group and /etc/gshadow files. Some of the descriptions in the User Settings window may be less than perfect. For example, “ethernet” should not be lowercase, and sharing files with the local network should not be limited to Samba. As development proceeds, the descriptions in the User Settings window shown in Figure 4-1 may change.

**TIP**

Don't modify a text-configuration file with a binary editor such as [OpenOffice.org](#) Writer. Changes written to those files are likely to render them unusable. Changes to key configuration files may even lead to unbootable systems. If you want to edit the files of the shadow password suite, use the `vipw` and `vigr` commands. These commands temporarily override the read-only permissions of the `/etc/shadow` and `/etc/gshadow` files.

TABLE 4-7 Groups with special permissions for users.

GROUP NAME	DESCRIPTION
plugdev	Access external storage devices automatically.
admin	Administer the system.
lpadmin	Configure printers.
dip	Connect to the Internet using a modem.
netdev	Connect to wireless and ethernet networks.
adm	Monitor system logs.
fuse	Mount user-space filesystems (FUSE)*.
fax	Send and receive faxes.
sambashare	Share files with the local network.
audio	Use audio devices.
cdrom	Use CD-ROM drives.
floppy	Use floppy drives.
dialout	Use modems.
tape	Use tape drives.
video	Use video devices.

* FUSE is an acronym for Filesystems in USErspace.

The groups described in this section apply only to Ubuntu systems. Different Linux distributions use different groups or different group names to achieve the same goal. It's certainly possible to configure other Linux distributions with such group privileges, however. To do so, you'll need to do the following:

- If the specified groups don't already exist on the target distribution, in both the `/etc/group` and `/etc/shadow` configuration files, you may need to create them.
- On an Ubuntu system, identify the files owned by the noted group. One method is with the `find` command. For example, the following command identifies those files owned by the `dip` group. On Ubuntu systems, user members of that group are allowed to dial out through a telephone modem.

```
# find / -group dip
```

- Check the octal permissions associated with the noted files.

- If those same files exist on the target distribution, use commands like `chgrp` and `chmod` to modify ownership and permissions accordingly. In some cases, you may need to create the target files.

Securing Groups of Users

Administrators need to be able to set up users in special groups, with dedicated directories. Administrators can give those users rights and privileges on dedicated directories. The noted directories can be shared by groups of workers. Before configuring such groups, you need to understand how the **user private group scheme** works in Linux.

User Private Group Scheme

Linux users are typically assigned to at least one group, which is based on that user's account. For example, most current Linux systems with user `kim` also have a group named `kim`. Of course, user `kim` may be a member of other groups with access to hardware, administrative privileges, and more.

Every user has a primary group. To identify that group, review the third and fourth fields in the `/etc/passwd` file. For most user accounts, the numbers in these fields are the same. To verify the group name associated with each number, check the `/etc/group` file.

Linux distributions that don't implement the user private group scheme may assign all users to the same group. That group is typically named `users`, with a GID of 100.

Create a Special Group

If you're creating a group for a special set of users, you'll probably want to set up a directory for their exclusive use. The following instructions describe how you can set up a `/home/special` directory for some series of users in a group named `project`. Assume the user members of that group are users `larry`, `kim`, `kate`, `mike1`, and `mike2`. Assume those user accounts have already been created. Further assume there is a user `nobody` with a nonprivileged UID.

1. Run a command like `mkdir /home/special` to create the directory for the special group.
2. Create the group named `project`. Make sure to assign that group a GID that can't otherwise be assigned. To be sure, check the `/etc/login.defs` file and make sure the GID number is greater than the `GID_MAX` setting. If the local system uses network authentication, make sure the GID number isn't assignable in any of those systems. For example, to create a group named `project` with a GID of 100000, run the following command:

```
# groupadd -g 100000 project
```

3. Assign the noted users to that `project` group. Although you can do so with the `usermod` command, one user at a time, you can also directly edit the `/etc/group` and `/etc/gshadow` files to add the noted users. Given a GID of 100000, the applicable lines in these files would be as follows:

```
project:x:100000:larry,kim,kate,mike1,mike2
project:!:larry,kim,kate,mike1,mike2
```

4. Set up appropriate ownership in the newly created directory. The following command ensures that the user can change permissions in the `/home/special` directory:

```
# chown nobody.project /home/special
```

Alternatively, if you wanted to give user `larry` ownership of the directory, you could run the following command:

```
# chown larry.project /home/special
```

5. Finally, use the following command to set up special octal permissions on the directory.

```
# chmod 2770 /home/special
```

As described later in this chapter, the `2` in the `chmod` command assigns the so-called **set group ID (SGID) bit** to the directory. (The SGID bit is a special permission commonly applied to directory.) Alternatively, if you just wanted to set the SGID bit, run the `chmod g+s /home/special` command.

Once configured, all members of the project group will be able to copy files to the `/home/special` directory. The SGID bit in that directory assigns group ownership to any files copied to that directory. Thus, all users who are members of that project group will have group-ownership access rights to files in that directory.

If you run these steps from a GUI command-line console, you'll have to log completely out of the GUI and log back in again before the noted ownership and permissions take effect.

Configuring the Hierarchy of Administrative Privileges

You could always run administrative commands as the root user. That can be dangerous, however, because the actions required to undelete a file are complex. Although it's common to delete groups of directories and subdirectories, that's a risky business. Say, for example, you wanted to delete the hypothetical `/usr/src/someone-source/` directory along with its files and subdirectories. You could run the following command as the root administrative user:

```
# rm -rf /usr/src/someone-source
```

That is a risky command, however. An accidental space added in the wrong place in that command would delete every file on the local system. That's one reason most Linux administrators prefer to log in and even administer a system from a regular account.

This section explains how that's possible—how regular accounts can be configured as administrators for certain servers. It explains the use of the `su` and `sg` commands with super-user and super-group privileges. It further explains how to run fine-grained rights to administrative tools through the `sudo` command, authorized through the `/etc/sudoers` file. (Using the `sudo` command, you can connect as the administrative user.)

Administrative Privileges in Services

Some services support the configuration of administrative privileges. Perhaps the prime example is the Common Unix Printing System (CUPS). In its main configuration file, `/etc/cups/cupsd.conf`, you may find a `SystemGroup` directive. That directive can be configured to assign a group such as `lpadmin` or `sys` as a print administrator group. Users who are members of that group will have print administrative rights to CUPS. When CUPS prompts for an administrative user and password, that print administrator can enter his or her regular username and password.

Administrative privileges to other services are more subtle, based on group ownership of special files. For example, Ubuntu systems authorize access to many log files for users who are members of the `adm` group. One way to verify the files owned by the `adm` group is with the following command:

```
# find / -group adm
```

Assuming that the group owner of such files has at least read privileges, users who are members of the `adm` group will be able to read those files.

The `su` and `sg` Commands

The `su` and `sg` commands allow users to assume the identity of others. You can set it up for one command or you can assume the identity of another user or group until you log out of that new user or group. You can use the `su` command by itself to log in from a regular account into the root administrative account. That action prompts for the root password.

Because logging in as the root administrative user is dangerous, the right way to use the **su** command is with the **-c** switch. This applies administrative privileges for that one command. For example, the following command opens the noted /dev/sda drive in the fdisk utility.

```
$ su -c '/sbin/fdisk /dev/sda'
```

Before the command is executed, you're prompted for the root administrative password. When changes are complete, the **su** command returns the shell to your regular account.

You can also use the **su** command to log into a different account. For example, if you have the password of the user named humu, you can log into his or her account with the following command:

```
$ su - humu
```

The **sg** command allows a user to join a group on a temporary basis. It works only if there's a group password in the /etc/gshadow file. For example, if you have a regular account named humu and have the group password for the project group described earlier in this chapter, you can use the **sg** command to access that directory. The following command would copy the noted file from user humu's home directory:

```
$ sg project -c 'cp /home/humu/mycontribution.doc /home/project'
```

Just remember, the use of such passwords can be a security risk. Because the root administrative password is all-powerful for a system, it should be shared with as few people as possible. In addition, if you send an administrative password over a network, that's a risk, even over an encrypted connection. That's one reason for the **sudo** command, configured in the /etc/sudoers file.

Options with **sudo** and /etc/sudoers

The Linux way to configure limited administrative permissions is based on the **sudo** command, configured in the /etc/sudoers configuration file. With **sudo**, authorized users need only enter their own regular passwords to run configured administrative commands. In the following sections, you'll look at basic options in /etc/sudoers, take a more extensive view of the commands that can be delegated through /etc/sudoers, and see how the **sudo** command works with this file.

The **sudo** command is so important in the Ubuntu distribution that you won't even find a root administrative password on Ubuntu releases. Ubuntu disables logins to the root account. The first user on an Ubuntu system is given membership in the admin group. A regular user who is a member of the admin group can run administrative commands with full privileges (if that group is appropriately configured in /etc/sudoers). As described later, all that user needs to do is to enter his or her own password when prompted.



NOTE

Most Linux distributions make it difficult to log in directly to the root account. It is possible to boot into single user mode, which will give you root access. When you have root access, you can create a password for the root user using the **passwd** utility. When the root user has a password, he or she can be allowed to log in.

Basic Options in /etc/sudoers

The standard /etc/sudoers file contains the following entry, which gives the root administrative user full privileges through **sudo**:

```
root ALL=(ALL) ALL
```

The format of this line is as follows (**if run_as_username** is not included, it's assumed that the command that follows is run as root):

```
user system=run_as_username command
```

The user can be any regular user account in /etc/passwd. Alternatively, a name with a % in front specifies a group. So the following entry configures full administrative privileges for members of the admin group:

```
%admin ALL=(ALL) ALL
```

If your account is listed as a member of the admin group in /etc/group, you can run any administrative command. From your regular account, the following command would open the /dev/sda hard drive in the fdisk utility. The **sudo** command prompts for your user password, not the root administrative password:

```
$ sudo /sbin/fdisk /dev/sda
```

Of course, any unauthorized user who finds the username and password of someone in the admin group would have full administrative privileges. Fortunately, though, because the username is something other than root, the unauthorized user might not know what he or she has.

Nevertheless, that's one reason why you as a security administrator can configure /etc/sudoers with more fine-grained administrative privileges. It's a file worth protecting. By default, it's accessible only to the root administrative user and is configured with read-only privileges. As such, you can't edit it in just any text editor. To edit it, you need to run the **visudo** command as the root user. As the name implies, it opens /etc/sudoers in the vi editor, requiring some knowledge of that editor. Many (if not most) entry-level Linux texts describe the use of the vi editor in detail.

You can allow a user access to run very specific programs as a superuser. This provides additional restrictions rather than just allowing access to all commands. For example, you may want a user to be able to mount devices but not perform any other actions. The following line allows members of the users group from all systems to run the commands shown. Note how the full path to each command is required. This is a simple example.

```
%users ALL=/bin/mount /mnt/cdrom, /bin/umount /mnt/cdrom
```

You can try to add the preceding line to something more realistic in your system, changing the options shown to include some relevant commands. For example, the following replacement code allows members of the users group to use fdisk on the /dev/sda drive (and only that drive) and to use the yum command to update and install packages:

```
%users ALL=/sbin/fdisk /dev/sda, /usr/bin/yum
```

More Detailed Options with /etc/sudoers

The power of /etc/sudoers comes earlier in the Red Hat version of this file, with the aliases. You can set up aliases for a group of users, systems, and administrative commands. Say you want to set up an administrator with the right to reprioritize or kill a process. The following command alias directive assigns appropriate administrative tools to the PSMGMT alias:

```
Cmnd_Alias PSMGMT= /bin/nice,  
↳ /bin/kill, /usr/bin/kill, /usr/bin/killall
```

You can then assign that to a group of users with the authority to use these commands on all systems with the following directive:

```
%psusers ALL = PSMGMT
```

TABLE 4-8 Groups of commands in /etc/sudoers.

VARIABLE	COMMANDS
NETWORKING	route, ifconfig, ping, dhclient, net, iptables, rfcomm, wvdial, iwconfig, mii-tool

```

SOFTWARE    rpm, up2date, yum
SERVICES   service, chkconfig
LOCATE     updated
STORAGE    fdisk, sfdisk, parted, partprobe, mount, umount
DELEGATING sudo, chown, chmod, chgrp
PROCESSES  nice, kill, killall
DRIVERS    modprobe

```

A more extensive example of administrative tools that have been assigned to command aliases is shown here, taken from a Red Hat /etc/sudoers file:

```
# %sys ALL = NETWORKING, SOFTWARE, SERVICES, STORAGE, DELEGATING,
PROCESSES, LOCATE, DRIVERS
```

If activated, this line would enable privileges to an extensive series of administrative commands. [Table 4-8](#) lists the commands associated with each variable. It should also give you a good idea of what basic administrative commands are really important in Linux.

If you know Linux in detail, you may realize this list is not complete. For example, you could set up commands like `iwconfig`, `iwlist`, and `iwspy` in their own wireless networking group. You may want to add commands like `insmod` and `rmmmod` to the DRIVERS group.

Use the `sudo` Command

Regular users may be configured with permissions in the /etc/sudoers file. Members of the aforementioned admin group can run administrative commands from regular accounts. For example, if your account is a member of the admin group, you could open the second SATA drive on the local system for editing with the following command:

```
$ sudo /sbin/fdisk /dev/sdb
```

The first time such a trusted user prefacing an administrative command with the `sudo` command, he or she will see the following response:

```
We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:
```

- #1) Respect the privacy of others.
- #2) Think before you type.
- #3) With great power comes great responsibility.

Password:

When you see this message, enter the regular password for your account. The root administrative password, if it exists, should not work here. If that root password does work, then your regular and root passwords are identical, and that's a different security risk.

If your user account is not authorized through /etc/sudoers, you'll see the following message:

```
userx is not in the sudoers file. This incident will be reported.
```

The **sudo** command then generates a log entry in the appropriate log file, based on the configuration files for the logging service used.

Regular and Special Permissions

Linux files and directories have a user and a group owner. Permissions are split into three groups: the user who owns the file, the users in the group that owns the file, and all other users.

Regular permissions are read, write, and execute. Special permissions go beyond execute bits and can extend the executable permissions. In general, these special permissions, especially the **set user ID (SUID) bit**, may be a security risk. (The SUID bit is a special permission that allows others to execute the given file with the rights of the user owner of the file.) If there's a binary that's vulnerable to the **ptrace** system call, however, the SUID or SGID bit may stop that system call in its tracks, preventing an unauthorized user from taking control of the process. The **chmod** command is what is used to set permissions on files and directories.

The Set User ID Bit

A file with the SUID bit allows other users to run that command, with the permissions assigned to that user owner. One command with the SUID bit is `/usr/bin/passwd`. When you apply the `ls -l` command to this file, you'll see the following output:

```
-rwsr-xr-x 1 root root 27768 Jul 17 2015 /usr/bin/passwd
```

The **s** in the fourth alphanumeric character position is the SUID bit. As a result, the `/usr/bin/passwd` command is accessible to all users. Without the SUID bit, a normal user running `passwd` would not be able to actually change their password because the `passwd` binary needs to have write access to the `/etc/shadow` file. When a normal user runs the `passwd` program, it runs with the permissions of that user. The SUID bit is necessary to be able to write changes out to the `/etc/shadow` file. That's limited by its PAM configuration, however.

There are two methods to set the SUID bit on an executable file. The following command just sets the SUID bit on the `/home/michael/filename` file:

```
# chmod u+s /home/michael/filename
```

Alternatively, the following command sets the SUID bit with read and write permissions for the user and group owners of the file:

```
# chmod 4660 /home/michael/filename
```



WARNING

The SUID bit tells the operating system to run the program as the user who owns the file. If this program has a bug in it that can be exploited to run code specified by an attacker, that code will be run as the user who owns the file. This makes files that are SUID root particularly dangerous.

The SUID bit can be a security issue. An unauthorized user may be able to use it from any account. While the `/usr/bin/passwd` command is carefully protected in a PAM configuration file, other SUID files may not be so well protected. One way to identify files on the local system with the SUID bit is with the following command:

```
# find / -perm +4000
```

One standard executable command with the SUID bit is `/bin/ping`. If you believe regular users should not be given access to the `ping` command or are paranoid about users starting a **ping storm**, you should unset the SUID bit on this file. You may also hear this referred to as a *ping flood*. A ping storm or ping flood is a denial of

service attack with large volumes of Internet Control Message Protocol (ICMP) messages sent to a target with the goal of taking the target offline. To unset the SUID bit from the ping program, run the following command:

```
# chmod u-s /bin/ping
```



NOTE

The `ping -f` command can flood a system with Internet Control Message Protocol (ICMP) packets. Hypothetically, without an interval, the `ping -f` command would send 100 packets per second to the given IP address. Because that can overwhelm a system, that ability is limited to the root user.

The Set Group ID Bit

As described earlier in this chapter, one use for the SGID bit is for shared directories. In that example, the SGID bit was set on the `/home/special` directory. Ownership of files written by user members of that project group was modified. The SGID bit makes the project group the owner of all such files written to the `/home/special` directory. One way to identify files on the local system with the SGID bit is with the following command:

```
# find / -perm +2000
```

The SGID bit is also frequently found on executable files intended to be run by groups. However, it's a way to protect certain binaries from attacks using the `ptrace` system call. One file protected by the SGID bit is the `ssh-agent` command. Without the SGID bit, a `ptrace` system call could bypass a passphrase-based Secure Shell (SSH) connection to a remote system.

The Sticky Bit

Like the SGID bit, the **sticky bit** is also normally applied to a shared directory. It allows any user to add files to and delete files from that directory. Unlike the SGID bit, the sticky bit doesn't change the ownership of any files added to that directory. Perhaps the most common use for the sticky bit is the `/tmp` directory. Users who are logging into the GUI require the ability to write to that directory. However, users do not have the ability to overwrite or delete files written there by other users.

While other users may be able to read what is written to `/tmp`, they won't be able to delete or overwrite those files. If you want to set the sticky bit on a different directory, the `chmod` command can help. The following command sets just the sticky bit on the `/newtmp` directory:

```
# chmod o+t /newtmp
```

Alternatively, the following command sets the sticky bit with read, write, and execute permissions for all users on the `/newtmp` directory:

```
# chmod 1777 /newtmp
```

Tracking Access Through Logs

Linux supports logging for services and the kernel. Until recently, this was configured in two different service daemons, `syslogd` and `klogd`, in the `/sbin/` directory. The latest versions of Linux include a successor, the `/sbin/rsyslogd` daemon. The functionality has not changed. The system and kernel logs are so intertwined, they're usually part of the same package, `rsyslog` or `sysklogd`. In either case, Ubuntu and Red Hat take different approaches to logging configurations from this file.

This section focuses on tracking access—that is, finding those log files that record login attempts and especially login failures. These services classify log messages in a number of areas. The relevant categories are **auth** and **authpriv**. In the language of the log-service configuration files, **auth** and **authpriv** are known as *facilities*. While both transmit the same messages, the **authpriv** facility is normally associated with a more secure file.

This section addresses the basics of tracking access through logs. If you're an administrator for multiple systems, however, you may prefer a system that sends you an alert when something goes wrong. That's a log-monitoring system.

Authorization Log Options

Many Linux distributions are in transition from the sysklogd to the rsyslog package. The main sysklogd configuration file is /etc/syslog.conf. The main configuration file from the rsyslog package is /etc/rsyslog.conf. On Ubuntu systems, it refers to the 50-default.conf file in the /etc/rsyslog.d/ directory.

Despite the changes to the names and locations of the configuration files, the basic directives haven't changed. Specifically, for Debian-based systems such as Ubuntu, the following directive sends information on login attempts to the /var/log/auth.log file:

```
auth,authpriv.* /var/log/auth.log
```

For Red Hat-based systems, the following directive sends information on login attempts to the /var/log/secure file:

```
authpriv* /var/log/secure
```

Authorization Log Files

The aforementioned log files are important tools in the battle to keep a system secure. Courtesy of the /etc/logrotate.conf file and a regular **cron** script, logs are rotated on a weekly basis. (**cron** is a service for running administrative jobs on a regular basis.) Linux systems configured in this way typically include several weeks of logs. Logs from previous weeks have a .? extension, where the ? is a number. (The ? is a standard Linux single alphanumeric character wildcard.)

The size of these log files may be important. A big jump in the size of any log file indicates increased activity. While this increased activity may just be a result of user-based **cron** jobs that are run more frequently, it may also reflect a large number of external login attempts. Look at the /var/log/secure.1 file in [Figure 4-2](#).

The log file excerpt in [Figure 4-2](#) displays messages associated with an attempt to log into a system named RHELserver, first as user test and then as user test1. It also provides information on the protocol, the IP address of the attacker, and the result.

```

Feb 21 10:49:28 RHELserver sshd[27551]: Invalid user test from 60.217.234.134
Feb 21 10:49:28 RHELserver sshd[27552]: input_userauth_request: invalid user test
Feb 21 10:49:28 RHELserver sshd[27551]: pam_unix(sshd:auth): check pass; user unknown
Feb 21 10:49:28 RHELserver sshd[27551]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=60.217.234.134
Feb 21 10:49:28 RHELserver sshd[27551]: pam_succeed_if(sshd:auth): error retrieving information about user test
Feb 21 10:49:30 RHELserver sshd[27551]: Failed password for invalid user test from 60.217.234.134 port 41495 ssh2
Feb 21 10:49:30 RHELserver sshd[27552]: Received disconnect from 60.217.234.134: 11: Bye Bye
Feb 21 10:49:32 RHELserver sshd[27553]: Invalid user test1 from 60.217.234.134
Feb 21 10:49:32 RHELserver sshd[27554]: input_userauth_request: invalid user test1
Feb 21 10:49:32 RHELserver sshd[27553]: pam_unix(sshd:auth): check pass; user unknown
Feb 21 10:49:32 RHELserver sshd[27553]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=60.217.234.134
Feb 21 10:49:32 RHELserver sshd[27553]: pam_succeed_if(sshd:auth): error retrieving information about user test1
/var/log/secure.1

```

FIGURE 4-2

Failed remote login attempts.

Pluggable Authentication Modules

PAMs are used primarily to regulate access to administrative tools and commands. It works as an additional layer of security for users. A substantial number of services and systems are PAM-aware. In other words, with the right modules, access to those services and systems can be regulated by rules defined in PAM configuration files in the /etc/pam.d/ directory.

To review available PAM modules for your distribution, search the applicable archives. On a Red Hat-based system, you can search with the `yum search pam` command. On an Ubuntu system, you can search with the `apt-cache search libpam` command. Just the titles in the output shown in [Figure 4-3](#) provide a feel for the full capabilities of PAM.

PAM is configured in specific files in the /etc/pam.d/ directory. Those files, such as `passwd` and `sshd`, accurately depict the command or service that is regulated.

The Structure of a PAM Configuration File

The structure of a PAM configuration file is based on its modules, available from the /lib/security/ directory. If you don't find a desired module here, you may need to install another PAM-related package. Each line in a PAM configuration file is set up in the following format:

module_type control_flag module_file (arguments)

```
michael@LucidServerVM:~$ apt-cache search libpam
libpam-ck-connector - ConsoleKit PAM module
libpam-cracklib - PAM module to enable cracklib support
libpam-doc - Documentation of PAM
libpam-gnome-keyring - PAM module to unlock the GNOME keyring upon login
libpam-krb5 - PAM module for MIT Kerberos
libpam-ldap - Pluggable Authentication Module for LDAP
libpam-modules - Pluggable Authentication Modules for PAM
libpam-mount - PAM module that can mount volumes for a user session
libpam-opie - Use OTPs for PAM authentication
libpam-p11 - PAM module for using PKCS#11 smart cards
libpam-radius-auth - The PAM RADIUS authentication module
libpam-runtime - Runtime support for the PAM library
libpam-smbpass - pluggable authentication module for Samba
libpam0g - Pluggable Authentication Modules library
libpam0g-dev - Development files for PAM
opie-server - OPIE programs for maintaining an OTP key file
update-motd - superceded by pam motd in libpam-modules
ldapscripts - Add and remove user and groups (stored in a LDAP directory)
libpam-afs-session - PAM module to set up a PAG and obtain AFS tokens
libpam-alreadyloggedin - PAM module to skip password authentication for logged users
libpam-apparmor - changehat AppArmor library as a PAM module
libpam-blue - PAM module for local authentication with bluetooth devices
```

FIGURE 4-3

PAM-related packages on Ubuntu.

Older versions of PAM required the full path to the module. Current versions assume the module is in the /lib/security/ directory. If installed, more information on a *module_file*, such as the arguments used for a module, may be available from a man page. That leaves the *module_type* and *control_flag* options. First, there are four different module types available:

- **auth**—The **auth** module type authenticates users by verifying passwords, group memberships, and even Kerberos tickets. Also known as *authentication management*.
- **account**—The **account** module type checks the validity of the account based on expiration dates, time limits, or configuration files with restricted users. Also known as *account management*.
- **password**—The **password** module type controls changes to user passwords. It may also control the number of login attempts. Also known as *password management*.
- **session**—The **session** module type makes the connection work. It may mount appropriate directories and send information to system logs. Also known as *session management*.

There is an implicit fifth module type, **include**, which incorporates the configuration directives from another file.

Next, there are four different control flags available. Success or failure in the flag determines the next step for the configuration file:

- **required**—The **required** flag indicates that the module in the current line must work before PAM proceeds to the next line in the file. If the module doesn't work, the authentication attempt fails. However, PAM proceeds to test the lines that follow.
- **requisite**—The **requisite** flag indicates that the module in the current line must work before PAM proceeds to the next line in the file. If the module doesn't work, the authentication attempt fails and PAM does not proceed to the lines that follow.

- **sufficient**—Assuming no previous **required** or **requisite** control flag has failed, success in the **sufficient** flag means the request for access is approved.
- **optional**—The **optional** flag is normally ignored unless no other control flags have returned success or failure.

With that information in hand, look at some directives in files in the /etc/pam.d/ directory. A couple of important password **module_type** directives from Red Hat's system-auth file are shown here:

```
password requisite pam_cracklib.so try_first_pass retry=3
```

The first command uses the **pam_cracklib** module. When a new password is entered, that module checks the new password against dictionary words. The **try_first_pass** argument looks for and uses any password that was previously entered. The **retry=3** argument allows a user to try to enter the password three times.

```
password sufficient pam_unix.so md5 shadow nullok
  ↗ try_first_pass use_authtok
```

This next command uses MD5 hashes for passwords, in the context of the shadow password suite. The **nullok** option allows null passwords, which is sometimes required for accounts that run PAM-protected services.

PAM Configuration for Users

You can further configure a PAM configuration file to limit access to certain services or commands to specified users. That's made possible by the **pam_listfile** module. One example that puts that module into effect is the Red Hat version of the very secure FTP daemon, vsftpd. Access to this service is limited in the /etc/pam.d/vsftpd configuration file, courtesy of the following directive:

```
auth required pam_listfile.so item=user sense=deny
  ↗ file=/etc/vsftpd/ftpusers onerr=succeed
```

This line relates to authentication management. The **required** means any access attempt must pass this test, based on the **pam_listfile** module. The **sense=deny** option denies access to users listed in the noted file, /etc/vsftpd/ftpusers. If that file doesn't exist, the **onerr=succeed** option means any limitations are ignored.

Authorizing Access with the Polkit

The polkit, developed by the freedesktop.org group, is a way of solving the administrative permission problem. When programs need to temporarily escalate privileges, the polkit is a framework to provide the higher-level function without having to have the whole program gain administrative access.

The polkit is written with authentication agents and applications that use the authentication agents. There is an API for the framework that allows applications to be written to use the polkit implementation. Where escalation of privileges works well on the command line with **sudo** or **su**, when it comes to graphical programs, there aren't really the same mechanisms that work well. You can run the entire program as root, which exposes the system to the potential for exploit. But it's better to escalate only during the functions that specifically need that administrative access. This is what polkit is available for.

FYI

The leading developers of the polkit are affiliated with freedesktop.org. The freedesktop.org group was created to manage interoperability projects on different Linux X Window System desktop environments. That group is loosely

associated with the Free Standards Group at the Linux Foundation, which also happens to be the current sponsor of the work of Linus Torvalds on the Linux kernel.

How the Polkit Works

When a user in the admin group opens an administrative tool, that user doesn't get immediate access. Instead, the user needs to click an Unlock button and then enter his or her regular password. At least, that's based on default polkit settings.

The polkit can be configured in a number of different ways. With implicit authorizations, you can leave the access rules to admin users in /etc/sudoers. And you can go further. Such access can be limited to the local console.

Furthermore, with explicit authorizations, you can regulate the access to administrative tools by user. That access can last for a single local section or may be set permanently, even after a reboot. Needless to say, such a policy would make the associated account a more appealing target to a malicious user.



NOTE

Earlier versions of the polkit, called PolicyKit at the time, were so restrictive, they did not allow access by the root user to GUI administrative tools unless root was made a part of the admin group. That restriction has been since lifted.

Polkit Concepts

The polkit assumes privileged tools are associated with two distinct processes: policies and mechanisms. In this context, a policy defines rules for executing an administrative tool. A mechanism runs in privileged mode.

The mechanism includes three parts: the subject, the object, and the action. The subject is the administrative tool in question. The object is the device or file to be modified. That object may be a device or a configuration file. The action specifies how the device or configuration file is to be modified. For example, a polkit action may format a USB key through its object configuration file, using a formatting administrative tool.

Early versions of the PolicyKit, as it was called then, were configured in part in the /etc/PolicyKit/ directory. The PolicyKit also included rules on subject administrative tools in the /etc/dbus-1/system.d/ directory. Once processed, you could find rules in the /var/lib/PolicyKit/ (or /var/lib/polkit-1/) directory.

Later versions of the PolicyKit, with the name now changed to polkit, replace the PolicyKit/ subdirectory with the polkit-1/ subdirectory.

Polkit works in conjunction with **dbus**. Dbus is an application programming interface (API) that allows processes to communicate with one another. Over the lifetime of Unix, there have been a number of mechanisms allowing interprocess communication. Dbus was developed by the freedesktop.org team. You will find it in most Linux distributions.

The Polkit and Local Authority

With local authority features, the polkit can provide one more layer of protection in case a black-hat hacker is able to connect to your system remotely. You can configure policies to require local access. Actual physical local access may be difficult for servers in racks and in remote locations, however.

The relevant commands are part of a console kit package; the name may vary by distribution. Three key commands include **ck -history**, **ck -list -sessions**, and **ck -launch -session**. These commands are of interest by themselves because they can help identify logged-in users in detail. If you see information about a session that you don't recognize, these commands may be able to help you identify the user and source.

With respect to the polkit, these commands can help identify the user and type of session. The polkit can use that information to determine whether to accept authentication from that user for the configured administrative tool.

The `ck-history` Command

The `ck-history` command can provide extended information about recent users—not only regular users, but also GUI access with the GNOME Display Manager (GDM). Useful options include the following:

- `--frequent`—The `--frequent` option for the `ck-history` command lists users by the frequency of their recent logins.
- `--last`—The `--last` option for the `ck-history` command lists last-logged-in users by user, session, seat, and time. The `--last-compat` switch may provide a more readable format.

The `ck-list-sessions` Command

The `ck-list-sessions` command can provide extended information about current login sessions. When run on a remote system, it would display a login at the console, in the GUI, and from a remote location as three different sessions. The command includes details such as the time when the session was started, the remote host, the UID, and even the GUI display device if applicable.

The `ck-launch-session` Command

The `ck-launch-session` command is useful for remote administrators. The polkit can be used to limit access to specified administrative tools based on whether the console is local or remote. The `ck-launch-session` command can start a local session from a remote connection.

Network User Verification Tools

It's all fine and good to know how to authenticate users on a local system. But networks don't work very well unless users can log into not just a single system but an entire network. Linux includes two basic options for network-based authentication: NIS and LDAP. These are known as *directory services*.

Of course, you may not want to enable root administrative authentication to all systems with one network login. Even the relatively unsecure NIS system makes it difficult to set up a single root administrative account for a network.

Users on Linux systems can also be configured on Microsoft-based authentication systems. Their usernames and passwords can be translated to Linux usernames and passwords with the right plug-in. If you're working with a Microsoft LDAP database, the associated Samba server plug-in is `idmap_ldap`.

Linux can be configured, using PAM, to authenticate users against an Active Directory server. Using the file-sharing program Samba, Linux can also be configured to be an Active Directory domain controller to not only participate in a Windows network as a member system, but also to behave as though it were the controlling server on a network of Windows clients.

Alternatively, you can configure Linux with Samba as an old-style Microsoft Primary Domain Controller (PDC), based on the authentication database first created for the Windows NT 4 server operating system.

Although Microsoft no longer supports Windows NT 4 with security updates, Samba developers will continue to support PDC features for the foreseeable future.

NIS If You Must

If you want to set up a network-authentication scheme and already understand the shadow password suite, NIS is a relatively simple option. It allows you to use the standard shadow password suite files as the authentication database for the local network. Configuration is a straightforward process. Once the NIS server is configured, all you need to do is point NIS clients to the server for authentication in `/etc/yp.conf` and `/etc/nsswitch.conf`.

NIS transmits data, including password hashes, over the network without encryption. Any unauthorized user who gets hold of these password hashes can eventually decrypt such passwords. It's recommended that you avoid NIS. But if you run NIS, be sure to configure behind a secure network firewall. Even then, the security of your passwords will depend on the integrity of every one of your users. In addition, if any system on the local network is ever taken over by an unauthorized user, you should assume that account passwords would be compromised.



NOTE

NIS has been superseded by NIS+, which resolved several of the issues from the implementation of NIS.

One step you can take is to enable the `/var/yp/securenets` file. Unfortunately, this file can limit access only by IP address. If the `portmap` package is version 5 or above, it is known as a *secure portmapper*, and cannot be used in concert with `/var/yp/securenets`.

Of course, you can further secure a system for NIS. If you have firewalls on the NIS server, review the output to the `rpcinfo -p` command. It'll provide information on the Transmission Control Protocol/Internet Protocol (TCP/IP) ports used by NIS, along with the protocol.

Not all NIS ports are automatically static. Fortunately, they can be fixed with options in the `/etc/default/nis` or `/etc/sysconfig/network` configuration files. [Table 4-9](#) includes suggested port options for various NIS services. The noted port numbers are not currently assigned.

TABLE 4-9 Suggested standard port numbers for NIS services.

SERVICE	DESCRIPTION	SUGGESTED PORT	COMMAND
ypserv	NIS server	834	YPSERVARGS="-p 834"
ypbind	Binder between NIS server/client	835	YPBINDARGS="-p 835"
yppasswd	Password management	836	YPPASSWDDARGS="-p 836"
ypxfrd	Map transfer server	837	YPXFRDARGS="-p 837"

LDAP Shares Authentication

If you're configuring authentication on a network, LDAP may be the more secure choice. Because it can also be used to authenticate users on Microsoft and Apple operating systems, it may be a better choice for a network with multiple operating systems. It supports encryption using both the Secure Sockets Layer (SSL) and its successor, Transport Layer Security (TLS).

Such encryption requires a digital certificate. Without an encryption certificate, LDAP would also transmit passwords over a network in cleartext. You can always purchase a digital certificate from an official authority such as VeriSign, Comodo, or GoDaddy. Alternatively, you can create an unofficial digital certificate with the `openssl` command. This is sometimes known as a *self-signed certificate*.

If you're looking to install LDAP on a local network, the Linux implementation is known as *OpenLDAP*. Most of the names of the client and server packages start with `openldap`. In addition, you may want additional tools to connect the LDAP database with PAM and the name server switch configuration file in `/etc/nsswitch.conf`. Typical names for such packages include `libpam_ldap` and `nss_ldap`.

While the format of a user account in LDAP differs from that of the files of the shadow password suite, that shouldn't be a handicap. If the migration tools aren't already included in your Open LDAP server package, look for a `migrationtools` package for your distribution.

LDAP doesn't have the same network port issues as NIS, as it normally uses port number 3306 for its communication.

Best Practices: User Privileges and Permissions

This section is designed as a “lessons learned” review of what the chapter has just covered. If you already know the basics of a particular topic, the notes from this section may be all that you need.

First, it's important to protect the files of the shadow password suite. Because the /etc/passwd and /etc/group files are world-readable, /etc/shadow and /etc/gshadow should be readable only by the root administrative user. As a Linux administrator, you need to know how to create, delete, and otherwise manage user accounts from the command line.

You can give users two kinds of special privileges. There are the standard privileges associated with hardware components such as CDs, telephone modems, and audio devices. In addition, there are the administrative privileges associated with system configuration, file sharing, and print management.

With the Linux user private group scheme, users are members of their own special group. As an administrator, you can create a special group of users. With the help of the SGID bit, that directory can be shared by those users, secure from perusal by others. The SUID, SGID, and sticky bits are special permissions that support different kinds of access by regular users.

Linux provides a hierarchy of administrative privileges. Access to such privileges is available with the **su**, **sg**, and **sudo** commands. Some administrative privileges are tied to membership in certain Linux groups. The **sudo** command is especially important because it supports special privileges for regular users as configured in /etc/sudoers.

Linux logs are system and kernel logs. Loggable authentication events are based on the **auth** and **authpriv** options in related configuration files. Such logs are stored in the /var/log/ directory and are rotated on a weekly basis. Attempts by unauthorized users to log into your systems may be found in files such as auth.log and secure in the noted directory.

PAM modules are used to further regulate access to administrative commands. PAM configuration files can be found in the /etc/pam.d/ directory, using modules stored in the /lib/security/ directory. The four different module types are **auth**, **account**, **password**, and **session**. The success or failure of the PAM module depends on the control flag that applies, which may be **required**, **requisite**, **sufficient**, and **optional**.

For GUI tools, the polkit goes further. It regulates access between the dbus system and components that you may want to control in the GUI. It can regulate access to a variety of tools by user. It can regulate access by whether the user is local or remote.

If you have a network-authentication database, security issues are there as well. The two Linux-based authentication databases are NIS and LDAP. NIS is easier to set up because it can almost directly use the files of the shadow password suite. In their default configuration, both services send passwords in cleartext. LDAP can be protected with a certificate authority from a third party such as VeriSign or GoDaddy, or a self-signed certificate created with the **openssl** command. NIS has no such option for encryption.



CHAPTER SUMMARY

The focus of this chapter was user security. It started with the files of the shadow password suite. Within that structure, Linux administrators can assign standard and administrative privileges based on group memberships. You can further customize administrative privileges in the /etc/sudoers file. SUID, SGID, and sticky bits may be used to set up specialized privileges for executable files and shared directories.

Login access can be tracked through logs collected in the /var/log/ directory. PAM modules can further customize user access to key administrative tools. Two Linux-based options are available for network-based authentication: NIS and LDAP.



KEY CONCEPTS AND TERMS

account
auth
chage
cron
Dbus
--frequent
groupadd
groupdel
Group ID (GID)
groupmod
--last
Lightweight Directory Access Protocol (LDAP)
Network Information Service (NIS)
optional
password
Ping storm
Polkit
required
requisite
Salt
session
Set group ID (SGID) bit
Set user ID (SUID) bit
sg
Shadow password suite
Sticky bit
su
sudo
sufficient
User ID (UID)
User private group scheme
useradd
userdel
usermod



CHAPTER 4 ASSESSMENT

1. Which of the following files is *not* normally readable by all users?
 - A. /etc/passwd
 - B. /etc/shadow
 - C. /etc/group
 - D. /etc/login.defs
2. Which of the following files contains information about time limits on a password?
 - A. /etc/passwd
 - B. /etc/shadow
 - C. /etc/group
 - D. /etc/gshadow
3. Which of the following commands can be used to revise expiration information on a user password?
 - A. **useradd**
 - B. **passwd**
 - C. **groupmod**
 - D. **chage**
4. The _____ command searches for all files owned by the group named audio. Assume you're logged into the root administrative account.
5. Which of the following statements is true with the user private group scheme?
 - A. There are no private groups in Linux.
 - B. User information in the group is private.
 - C. The primary UID for the user is the same as the primary GID for the user.
 - D. Users are members of the same private group.
6. Members of which of the following groups are frequently set up as printer administrators? (Select two.)
 - A. admin
 - B. adm
 - C. lpadmin
 - D. sys
7. Which of the following commands only requires the password of a configured standard user?
 - A. **sudoers**
 - B. **sudo**
 - C. **su**
 - D. **sg**
8. Enter the _____ command to open and edit the /etc/sudoers file in a command-line console.
9. Which of the following special permissions is associated with a shared directory? That directory is *not* accessible to others who are *not* members of the group owner of that directory.
 - A. SUID
 - B. SGID
 - C. Sticky bit
 - D. Executable bit
10. Which of the following options in a log configuration file collects information on login attempts and failures?
 - A. **auth**
 - B. **sys**

- C. `log`
 - D. `user`
11. Which of the following PAM modules is least related to login information?
- A. `auth`
 - B. `account`
 - C. `password`
 - D. `session`
12. Enter the _____ directory for PAM modules.
13. Which of the following polkit concepts is associated with configuring access rules to special desktop tools by user?
- A. Implicit authorizations
 - B. Explicit authorizations
 - C. Administrative authorizations
 - D. Polkit authorizations
14. Which of the following polkit commands can be used to identify user logins by session? (Select two.)
- A. `ck-history`
 - B. `ck-list-sessions`
 - C. `ck-launch-session`
 - D. `ck-logins`
15. Which of the following commands can help identify network ports used by NIS through the portmapper?
- A. `nismap -p`
 - B. `ypbind -p`
 - C. `rpcinfo -p`
 - D. `portmap -p`

CHAPTER

5 Filesystems, Volumes, and Encryption

A

S A LINUX EXPERT, you already know how to set up partitions, logical volumes, and redundant array of independent disks (RAID) systems. You know how to configure those systems on selected directories. You know how to format those filesystems. But there are choices that can enhance security or at least limit the risk to important data. These choices relate to how such volumes are organized, how they're mounted, and how they're formatted. Of course, encryption of such filesystems can also enhance security.

Although file and folder permissions are basic to Linux, they are the starting point for security on Linux. You'll examine these permissions when they're local and when they're mounted from remote systems. With the right settings, selected filesystems can be configured and mounted with quotas and access control lists. This chapter examines these topics in detail.

Chapter 5 Topics

This chapter covers the following topics and concepts:

- How to organize filesystems
- What options for journals, formats, and file size affect security
- How to use encryption
- What basic concepts for local file and folder permissions are
- What basic concepts for networked file and folder permissions are
- How to configure and implement quotas on a filesystem
- How to configure and implement access control lists on a filesystem
- What best practices exist for filesystems, volumes, and encryption

Chapter 5 Goals

When you complete this chapter, you will be able to:

- Configure Linux with more secure filesystems
- Encrypt files, folders, and volumes
- Work with local and networked file permissions
- Configure access control lists
- Set up quotas on a filesystem

Filesystem Organization

In Linux, all data is stored as files. A **filesystem** is a protocol that specifies how those files are stored, marked, and retrieved. As you'll see in this chapter, filesystems can be local or remote. In this section, you'll review some

basics of a filesystem, the filesystem hierarchy standard (FHS), how filesystems can be organized, and when it is practical to configure a Linux filesystem in read-only mode.

Filesystem Basics

At its core, a filesystem is a database. Filesystems are configured to store data. In Linux, filesystems are applied to a particular device. That device might be named /dev/sda1, which is a standard device file for the first partition on a standard first hard drive. Or that file might be set to something like /dev/md0, which is a standard device file for the first configured local **software RAID** array. (Software RAID is a version of RAID [redundant array of independent disks] that uses partitions instead of disks as components of the array.) Alternatively, that device could be /dev/VolGroup00/LogVol00, which is a standard device file for a logical volume.

FYI

Software RAID on Linux is slightly misnamed, as it is based on an array of partitions, not disks. But software RAID is more flexible. Although it's most efficient to set up software RAID arrays based on partitions of equal sizes, that's not required. Of course, to take advantage of the redundancy associated with RAID, you need to make sure the partitions that make up the RAID array are from different physical hard drives. Otherwise, the failure of a single physical disk could lead to the loss of multiple elements of the RAID array, potentially leading to the loss of all data on the array.

Files on a device are accessible only when they're mounted on a Linux directory. The way volumes are mounted on different Linux directories is documented in the /etc/fstab configuration file. In the next section, you'll review the FHS. You can mount a volume on just about any of those directories (or subdirectories).

The word *filesystem* can have a couple of different meanings. The first involves the formatting of the volume or partition. At a low level, that refers to the data structures on the disk that allow files and directories to be stored and retrieved easily. Different operating systems will use a different means of storing all of the metadata that is associated with the files and directories.

The second use of the word applies particularly to Linux and describes the layout of particular directories. Unlike other operating systems, all directories and files under Linux fall under a single root directory. Specific directories are used to store specific types of information. The word *filesystem* sometimes refers to the entire arrangement of directories on the disk.



NOTE

Incidentally, *filesystem* is sometimes also written as two words: *file system*.

The Filesystem Hierarchy Standard

The **Filesystem Hierarchy Standard (FHS)** is the way files and directories are organized on a Linux system. Because the current version of the FHS was defined back in 2004, most Linux distributions include their own variations. These directories may be dedicated to different functions such as the boot process, user files, logs, command utilities, and more. [Table 5-1](#) defines the major directories of the FHS. It includes only first-level subdirectories; in other words, it does not describe second-level subdirectories such as /etc/X11/. It includes notes on those directories that are practical to mount separately, along with directories that you could choose to mount in read-only mode.

As shown in [Table 5-1](#), some directories just belong with the top-level root directory. They include /bin/, /dev/, /etc/, /lib/, and /sbin/. If they're mounted on a different filesystem and there's a problem with that filesystem such as corruption, that could keep a system from booting.

Some of you may see additional subdirectories such as /proc/, /selinux/, /sys/, /net/, /smb/, and /tftpboot/. The first three are virtual filesystems, created during the boot process in local memory. The /net/ and /smb/ directories are standard mount points for networked filesystems using the automounter. The /tftpboot/ directory is a common location for Trivial File Transfer Protocol (TFTP) server files, which may include terminal-based systems such as those associated with the Linux Terminal Server Project (LTSP). This directory would exist only on systems running a TFTP server.

TABLE 5-1 Important filesystem hierarchy standard directories.

DIRECTORY DESCRIPTION	
/	Top-level root directory, always mounted separately
/bin/	Basic command-line utilities; should always be a part of /
/boot/	Linux kernel, initial RAM disk, bootloader files; frequently mounted separately
/dev/	Device files for hardware and software; should always be part of /
/etc/	Configuration files; should always be part of /
/home/	Home directories for every regular user; frequently mounted separately
/lib/	Program libraries; should always be part of / (/lib64/ may exist on 64-bit systems)
/media/	Standard mount point for removable media such as CD/DVD drives and universal serial bus (USB) keys; may also be used for other volumes such as a directory formatted to a Microsoft file system
/mnt/	Common legacy mount point for removable media and temporary filesystems
/opt/	Common location for some third-party applications; may be empty and can be mounted separately
/root/	The home directory for the root administrative user; should always be part of /
/sbin/	Primarily for administrative commands; should always be part of /
/srv/	Directory commonly used for network services such as those that share using FTP and HTTP; may be helpful to mount separately
/tmp/	Common location for temporary files; if the /tmp/ filesystem is full, users cannot log into the GUI
/usr/	Small programs generally accessible to all users; could be mounted separately and read-only
/var/	Log files, print spools; some distributions use it for network service files; may be helpful to mount separately

FYI

Linux uses the word **root** in several contexts. By itself, root is the name of the standard Linux administrative user. The top-level root directory is symbolized by the forward slash (/). In contrast, the home directory of the root user is /root/, which is a subdirectory of the top-level root directory (/). And in the configuration file for the Grand Unified Bootloader (GRUB), the `root (hd0, 0)` directive specifies the partition with the files normally mounted on the /boot/ directory.

Good Volume Organization Can Help Secure a System

With a Linux system, everything falls under the root directory. There is a single filesystem where everything is located. On some operating systems, additional partitions or volumes become accessible as separate entities from the primary hard drive. On Windows systems, for example, if you were to add an additional drive, you would get another drive letter through which you could get to all of the contents of the drive. On a Linux system, there is no such thing as a separate device that appears outside the filesystem contained under the root directory (/).

Instead, if you want to add another device, like a volume or partition, you mount it into the filesystem at a mount point. From the standpoint of the user and the operating system, everything is all connected. A **mount point** is just a directory that effectively becomes a shortcut to an external device. Rather than pointing to a place within the existing volume, the metadata in the filesystem will point to the external volume or device.



NOTE

The sections that follow describe first-level directories configured as separate mount points. Directories at lower levels, such as /var/ftp/ and /home/michael/ can also be configured as separate mount points.

There is a difference, though perhaps sometimes subtle, between a partition and a volume. A *partition* is a way to segment a drive. You may have multiple partitions on a hard drive. A *volume*, on the other hand, is a single entity that can be formatted with a filesystem. This might be a partition, but it may also be a portion of a multi-disk storage device like a RAID—which would also likely be partitioned—or a storage area network (SAN). A SAN is a large cluster of storage devices that can be carved up across multiple systems, acting like locally attached storage devices.



NOTE

The directory you use as a mount point doesn't have to be empty. If you mount an external device to a directory with content in it, the directory will display what is on the external device. This doesn't mean the original files in the directory go away. When you unmount the volume, the original content is accessible again.

Historically, one of the reasons for using additional volumes was the small size of the disks available. This is not as much the case anymore, but there are still long-time Unix and Linux users who rely on mount points. The use of mount points can provide additional security because some volumes can be mounted as noexec, meaning nothing can execute from it. This helps protect the system from the exploitation of a service and having a program execute from a file being uploaded into a directory like /var/, which should only contain files that would never be executed.

Although you can set a Linux system up with one very large partition without adding any additional mount points, some common directories are used as mount points. There is one issue to be aware of, however. When the system first starts up, any directory that has to be mounted from a separate volume won't be accessible. This can cause problems for the system startup. For example, if you were to make /bin/ and /sbin/ mount points, the system wouldn't start because there are programs in those directories that the system requires to start everything up. As an example, you wouldn't be able to mount anything because the mount program is in /bin/. This means you couldn't mount /bin/ to get to mount. It's a nice little paradox. There are some directories that are safe to use as mount points, however. The following sections explain those directories and what they are used for.

The /boot/ Directory

By default, Red Hat-based distributions set up the /boot/ directory on a separate filesystem, always on a separate partition. It's not realistic to set up the /boot/ directory as a logical volume, as it contains the Linux kernel and initial RAM disk files needed before Linux can read a logical volume. While it is possible to set up /boot/ on a

RAID array, a /boot/ filesystem is typically pretty small at 100 to 500 MB, is infrequently changed, and is therefore easier to back up.

The 100 or 500 MB that Red Hat configures for /boot/ is typically plenty of space for four or five versions of the Linux kernel and associated initial RAM disks. Older versions of the Linux kernel can typically be deleted. Generally, only developers have a need for more versions of the Linux kernel on any single system.

Using /boot/ as a mount point enables you to unmount it after the kernel is loaded or at least make it read-only. This can help protect the kernel from unexpected, unintentional, or malicious changes. If the kernel file were to be overwritten with a binary that didn't work or the file were just simply erased, the system wouldn't boot again until the kernel file was fixed. As a result, there are security advantages to using /boot/ as a separate volume that is only mounted when it's needed.

The /home/ Directory

You'll probably have more reasons to set up the /home/ directory as a separate mount point. It's easier back up user files configured in that way. It'll be easier to upgrade a distribution to a later release with much less risk to user files. Given the importance of user files, it may be helpful to set up the /home/ directory on a RAID array. If your system includes a growing number of users, it may be helpful to set up the /home/ directory on a logical volume. Of course, with some extra work, you can set up a RAID array from two (or more) logical volumes.

Having /home/ as a separate volume enables you to more easily upgrade the storage space for your users without having to reinstall the entire operating system. You can back up the /home/ volume and then restore it to a larger drive and move the larger volume into place on the system. It will still cause some downtime, but it will be quite a bit easier than doing an entire OS reinstall just to get more storage on your /home/ volume.

The /opt/ Directory

While /opt/ is not always used, it is a standard location for third-party applications. While those applications may take advantage of standard Linux configuration files in /etc/ and libraries in /lib/, the applications in /opt/ are generally not important in the actual Linux boot process. Examples of Linux applications that use this directory include the following:

- Adobe Acrobat
- Cross-over Office
- [OpenOffice.org](#) Suite (when installed directly from a downloaded package from [openoffice.org](#))
- RealPlayer for Linux

Because the /opt/ directory is not often checked, it may be helpful to set it up as a separate mount point. In fact, it's an excellent candidate for a logical volume. If the number of third-party applications on the local system grows, a configuration as a logical volume makes it easier to expand the space allocated to the /opt/ directory.

The /srv/ Directory

The /srv/ directory is designed to contain files associated with some FTP and HTTP services. Some distributions set up the same services in subdirectories of the /var/ directory. The same issues that apply to /srv/ also apply to those /var/ subdirectories.

Because FTP servers are often used to allow uploads to the system, making /srv/ a mount point to a separate partition can be helpful because you don't run the risk of filling up your system partition. If someone chooses to upload a library of multi-gigabyte movie files to your FTP server, a separate mount point would act as a barrier, keeping that upload from affecting the rest of the system.

The /tmp/ Directory

The /tmp/ directory is a common location for temporary files such as shared downloads. It's also a location for files associated with logins to the GUI. If the /tmp/ mount point or partition is full, users who try to log into the GUI are "stuck" with the command line.

If your systems use the /tmp/ directory for shared downloads, it may be helpful to set it up on a separate mount point on a separate volume for that directory. That way, users who download too much do not overload the local system.

The /var/ Directory

Files in the /var/ directory can easily grow quite large. Some distributions use subdirectories of /var/ to contain files associated with some servers. If there are no upload limits or if those limits are compromised, the available space on the drive can quickly be filled, which may result in the system halting.

In addition, the /var/ directory contains log files. The logs for enterprise-level Web sites can easily grow to several gigabytes every day. If such a directory of logs is not maintained, this array of large files could quickly overload just about any system.

The configuration of the /var/ directory on a separate volume can limit such risks.

Read-Only Mount Points

It is often possible and even desirable to set up a few mount points in read-only mode. This is another advantage of using mount points rather than directories on the primary volume. You can mount a volume or partition in read-only mode but you'd have to make permission changes to the entire directory tree to accomplish the same result in a directory that's on the primary volume. When configured correctly, such filesystems can make it just a bit more difficult for malicious users to do real damage to your systems. Three directory candidates for mounting in read-only mode are /boot/, /opt/, and /usr/. Of course, if you were to update a kernel or install a new package, you'd have to remount the filesystem in read-write mode so the kernel or package files could be written to the appropriate directories.

The following is a possible read-only configuration directive for the /boot/ directory in the /etc/fstab configuration file:

```
/dev/sda1 /boot ext2 ro,exec,auto,nouser,async 1 2
```

If you're familiar with the format of the /etc/fstab file, you should already know that an option like **defaults** would mount the noted filesystem in read-write mode. Of course, because you'll also have to mount that filesystem in read-write mode when updating related components such as the Linux kernel, it may be helpful to have the read-write configuration line in comments in the same file. That means that /etc/fstab file might also include the following line for the /boot/ directory filesystem:

```
# /dev/sda1 /boot ext2 defaults 1 2
```

Not all directories should be mounted in read-only mode. Users who log into the GUI need write access to the /tmp/ directory. Log information is not available unless the system can write to the /var/ directory. New packages can't be installed unless the system has write access to the /etc/, /usr/, and /lib/ directories. Of course, you could remount a volume in read-write mode when installing or updating software.

Closely related to the concept of a read-only filesystem is the use of a live CD/ DVD-based distribution. When you boot from one of these live distributions, the storage is read-only. So the operating system has to create a storage device from which to run. Rather than use physical storage, it creates a RAM disk, which is a virtual storage device that exists in memory. All the regular directory structures are there, but they don't exist on any hard disk or even removable media, like a USB stick. Some people use this as a security mechanism because every change to files or directories is wiped away when the system is rebooted. Nothing in the RAM disk is retained.

How Options for Journals, Formats, and File Sizes Affect Security

As odd as it sounds, the choices made with respect to journals, formats, and file sizes can affect security with respect to availability. To that end, the selection of a filesystem can affect how well it resists an attack. A filesystem journal keeps track of transactions that have been applied to the filesystem. These transactions can be

checked against the data in the filesystem to ensure integrity and to apply any changes that are missing. For this reason, a **journaled filesystem**—that is, a filesystem that keeps track of changes to be written—is more likely to survive a catastrophic event such as an unexpected power outage. Not all the filesystems Linux supports are journaled, however. The primary filesystem in use by most Linux distributions is ext2/3/4; it supports journaling starting with ext3.

Linux can handle an impressive array of filesystem formats. If you have any doubts, take a look at the Linux version of the **fdisk** drive-management tool. [Figure 5-1](#) shows a list of available filesystem partition type identifiers (IDs).

If you’re reading carefully, you may already realize that a filesystem partition identifier and a filesystem format are not the same thing. When fdisk creates a partition, it assumes you’re using the standard Linux partition ID of 83. If you’re setting up a different kind of partition, you’ll need to know different partition types. No matter which partition type you specify, you can format the partition with any filesystem you want.



NOTE

It is possible to use Linux’s fdisk to create partitions suited to a wide variety of operating systems. But if you want to create a partition associated with a different operating system, it’s generally best to do so with tools native to that operating system.

1	FAT12	24	NEC DOS	81	Minix / old Lin	bf	Solaris
2	XENIX root	39	Plan 9	82	Linux swap / So	c1	DRDOS/sec (FAT-
3	XENIX usr	3c	PartitionMagic	83	Linux	c4	DRDOS/sec (FAT-
4	FAT16 <32M	40	Venix 80286	84	OS/2 hidden C:	c6	DRDOS/sec (FAT-
5	Extended	41	PPC PReP Boot	85	Linux extended	c7	Syrix
6	FAT16	42	SFS	86	NTFS volume set	da	Non-FS data
7	HPFS/NTFS	4d	QNX4.x	87	NTFS volume set	db	CP/M / CTOS / .
8	AIX	4e	QNX4.x 2nd part	88	Linux plaintext	de	Dell Utility
9	AIX bootable	4f	QNX4.x 3rd part	8e	Linux LVM	df	BootIt
a	OS/2 Boot Manag	50	OnTrack DM	93	Amoeba	e1	DOS access
b	W95 FAT32	51	OnTrack DM6 Aux	94	Amoeba BBT	e3	DOS R/O
c	W95 FAT32 (LBA)	52	CP/M	9f	BSD/OS	e4	SpeedStor
e	W95 FAT16 (LBA)	53	OnTrack DM6 Aux	a0	IBM Thinkpad hi	eb	BeOS fs
f	W95 Ext'd (LBA)	54	OnTrackDM6	a5	FreeBSD	ee	EFI GPT
10	OPUS	55	EZ-Drive	a6	OpenBSD	ef	EFI (FAT-12/16/
11	Hidden FAT12	56	Golden Bow	a7	NeXTSTEP	f0	Linux/PA-RISC b
12	Compaq diagnost	5c	Priam Edisk	a8	Darwin UFS	f1	SpeedStor
14	Hidden FAT16 <3	61	SpeedStor	a9	NetBSD	f4	SpeedStor
16	Hidden FAT16	63	GNU HURD or Sys	ab	Darwin boot	f2	DOS secondary
17	Hidden HPFS/NTF	64	Novell Netware	b7	BSDI fs	fd	Linux raid auto
18	AST SmartSleep	65	Novell Netware	b8	BSDI swap	fe	LANstep
1b	Hidden W95 FAT3	70	DiskSecure Mult	bb	Boot Wizard hid	ff	BBT
1c	Hidden W95 FAT3	75	PC/IX				

Command (m for help):

FIGURE 5-1

Linux partition type identifiers.

Partition Types

The focus of this section is on current Linux partition types. It does not address partitions normally associated with any other operating system. In most cases, the partitions on your Linux systems will be one of five different partition types, as follows:

- **82**—The Linux swap partition is used for partitions dedicated to swap space.
- **83**—The Linux partition is used for standard partitions with data.
- **85**—The Linux extended partition can contain logical partitions.
- **8e**—The Linux LVM partition configures a partition that can be used as a component of a logical volume.
- **fd**—The Linux raid auto partition specifies partitions that can be used as components of a RAID array.

Most Linux partitions with data are of type 83. If you want to format a partition to the ext2, ext3, ext4, reiserfs, or xfs filesystems, you'll want to configure a partition of that type.

The Right Format Choice

Linux format commands apply to a partition, a logical volume, or a RAID array. For example, when you run a command such as the following on an associated device file, the partition, logical volume, or RAID array identified by the /dev/sda1 device file is formatted and ready for use:

```
# mkfs.ext3 /dev/sda1
```

But what format should you select for a filesystem? It depends. Start with the standard Linux filesystem formats: ext2, ext3, and ext4. These represent the second, third, and fourth extended filesystems. (The original Linux extended filesystem, ext, is obsolete.)

- **ext2**—The second extended filesystem, **ext2**, does not include journaling. Because the partition associated with the /boot/ directory can be small, it may be well suited to the ext2 filesystem. Writes to that partition are infrequent.
- **ext3**—The third extended filesystem, **ext3**, writes data to a journal before writing it to the actual filesystem. Because the system writes to file twice, performance is slower.
- **ext4**—The fourth extended filesystem, **ext4**, is suited for filesystems with some large files. It includes a defragmentation tool.

Other major Linux filesystems include variations on journaling. The **xfs** filesystem developed by the company originally known as Silicon Graphics is suited for systems that require the transfer of larger files. The ext4 filesystem is fairly new and also supports larger files. The **reiserfs** filesystem is a common option on the SUSE distribution. Based on the concept of balanced trees, it's suited for filesystems with a few large and many very small files.

Available Format Tools

It's easy to find the format commands available on a local system. Just about all of them are variations on the **mkfs** command. The **mkfs** command is the command to format and build a Linux filesystem. If command-completion features are active on your shell, all you need to do is type **mkfs** and press the Tab key twice. With the right packages, the **mkfs** command can be used to format all the filesystems just described. It also can be used to format Microsoft VFAT and NTFS filesystems.

You can use these commands to format the partition, logical volume, or RAID array of your choice. For example, the following commands are two different ways to format the /dev/md0 RAID array:

```
# mkfs.ext3 /dev/md0
# mkfs -t ext3 /dev/md0
```

A slight variation on the `mkfs` command is designed to format swap partitions; that's the `mkswap` command.

Using Encryption

Linux supports a number of options for encrypting files, directories, and volumes. In this section, you'll look at the wide variety of encryption tools available for Linux. But this section is just a snapshot; it's not meant to be a comprehensive list. You'll also see an example of how each of these data types can be encrypted. The tool selected is based on the default tool used for Ubuntu and Red Hat distributions.

Generally, you don't need to encrypt everything. The files of an open source operating system are well known. However, user files are often sensitive and are therefore excellent candidates for encryption. To that end, perhaps the prime candidate for encryption is any partition, volume, or RAID array associated with the `/home/` directory. Alternatively, you might just encrypt the home directory of a specific user or just a few selected files.

Encryption Tools

Linux supports a wide variety of encryption tools. Such tools can be split into two categories: **kernel-space tools** and **user-space tools**. Kernel-space tools depend primarily on modules that can be loaded with the Linux kernel. User-space tools can more easily utilize the libraries loaded in the `/lib/` directory. In general, while kernel-space encryption is faster, user-space encryption is easier to implement.

Most Linux kernels support a wide variety of encryption options. To get a sense of what's included, run the following command, which returns all lines with the term `CRYPT` in the configuration of the currently loaded kernel.

```
# grep CRYPT /boot/config-`uname -r`
```

Options for encryption tools are described in the sections that follow. Some tools listed in one category may be used to encrypt data in two or all three categories.

Encrypted Files

Perhaps the most common standard for file encryption on Linux is based on the GNU Privacy Guard (GPG) command, `gpg`, a user-space tool that can encrypt and add digital signatures to a file. But that's just one of many Linux tools available for file encryption. Several other Linux tools are shown in [Table 5-2](#).

Now look at two ways that the `gpg` command can be used. By default, it uses the CAST-128 block cipher encryption algorithm.

Encryption With a Passphrase

With the `-C` switch, you can encrypt any file with a passphrase. The first time the `gpg` command is run, it'll create a standard public key in the user's home directory, in the `.gnupg/` subdirectory. But that public key isn't used in this case. The user is prompted for a passphrase.

The selected file is saved in encrypted format, with the `.gpg` extension. Of course, if you're concerned about someone reading files in the local directory, you should delete the original unencrypted file. You can then send the encrypted file to others as desired.

Users can then decrypt the file with the `gpg` command. The user who runs that command is prompted for the original passphrase.

Encryption with a Public/Private Key Pair

Before you can protect a file with a GPG-based public/private key pair, you need to create those keys. To do so, run the `gpg --gen-key` command. You're prompted to choose from three different options:

TABLE 5-2 Linux file-encryption commands.

COMMAND DESCRIPTION	
bcrypt	Uses the blowfish encryption algorithm, based on a passphrase
ccrypt	Uses the U.S. Advanced Encryption Standard (AES); uses passphrases
gpg	Users may select from different encryption algorithms; may use passphrases or public and private key pairs
ncrypt	Users may select from different encryption algorithms; passphrases are hashed
pad	Uses one-time pad encryption

- **Elgamal**—A probabilistic encryption scheme, **Elgamal** was developed by Taher Elgamal. When running the **gpg** command, this scheme is paired with DSA for digital signatures.
- **DSA**—The **Digital Signature Algorithm (DSA)** is used by the U.S. government for digital signatures.
- **RSA**—A public-key algorithm, **RSA** is named for its developers, Rivest, Shamir, and Adleman.

Despite the available options, the documentation from the developers of GPG suggests that you should choose the first option because the other options simply provide digital signatures. The latest version of Ubuntu includes a fourth option, which includes RSA-based encryption and digital signatures. If this is the first time you've used the **gpg --gen-key** command, you'll see the following messages:

```
gpg: directory `/home/michael/.gnupg' created
gpg: new configuration file `/home/michael/.gnupg/gpg.conf' created
gpg: WARNING: options in `/home/michael/.gnupg/gpg.conf'
      ↴ are not yet active during this run
gpg: keyring `/home/michael/.gnupg/secring.gpg' created
gpg: keyring `/home/michael/.gnupg/pubring.gpg' created
```

After selecting a key, be prepared to select the following:

- An encryption-key size of between 1,024 and 4,096 bits
- A key lifetime in days, weeks, months, or years; alternatively, choose the default setting of 0, which means the key does not expire
- A name, e-mail address, and comment
- A passphrase

The key depends in part on a random number generator. If you get a **not enough random bytes available** message, you'll need to run a command that reads or writes a lot. One simple method is to open up a second terminal and run the following command a few times:

```
$ cat /boot/initrd* > /dev/null
```

When successful, you'll see messages similar to the following:

```
gpg: /home/michael/.gnupg/trustdb.gpg: trustdb created
gpg: key CA58E7F6 marked as ultimately trusted
      public and secret key created and signed.
```

To confirm the creation of a private and public key pair, run the `gpg --list-keys` command. This assumes you've set up the `gpg` command on both local and remote systems. The first two steps export and copy a public-private key pair on system A. The user on system B can encrypt a file and then send it to user A. Finally, user A can use his or her private key to decrypt the file.

1. On system A, export the public key with the following command. The name you use should reflect the name shown in the output from the `gpg --list-keys` command. The `gpg.pub` filename is arbitrary; you can use the filename of your choice. Quotes are not required around the name.

```
$ gpg --export Michael Jang > gpg.pub
```

2. Copy the public key file (`gpg.pub` in this example) to system B. It may be sent by e-mail or it could be copied by other means such as the `scp` command.

3. On system B, import the public key with the following command:

```
$ gpg --import gpg.pub
```

If successful, you'll see messages similar to the following:

```
gpg: key CA58E7F6: public key "Michael Jang (This is
  ↪ an encryption test)
  <michael@example.org>" imported
gpg: Total number processed: 1
gpg: imported: 1 (RSA: 1)
```

4. On system B, verify the import with the `gpg --list-keys` command. The last key shown should reflect the information from the same command run on system A.
5. On system B, encrypt the file of your choice. Quotes around the name, taken from the `gpg --list-keys` command, are required. This command encrypts the `wagthedog.odt` file in the file named `wagthedog.enc`.

```
$ gpg --out wagthedog.enc --recipient "Michael Jang"
  ↪ --encrypt wagthedog.odt
```

6. Copy the encrypted file to system A.
7. On system A, run the following command to decrypt the noted file:

```
$ gpg --out wagthedog.odt --decrypt wagthedog.enc
```

8. Still on system A, you'll be prompted for the passphrase created earlier in this section with messages similar to those shown here. The correct passphrase decrypts the noted file with information on the encryption key.

```
You need a passphrase to unlock the secret key for
user: "Michael Jang (This is an encryption test)
  <michael@example.org>" 1024-bit RSA key,
ID 33A09C78, created 2010-03-21 (main key
ID CA58E7F6)
Enter passphrase:
```

Ubuntu has recently made it easy to set up an encrypted directory for a single user during the installation process. It makes sense to encrypt the directory of that first user, especially because that user has implicit administrative privileges defined through the /etc/sudoers configuration file. While the standard Ubuntu method is with the **enterprise cryptographic filesystem (eCryptfs)**, a system adapted by Ubuntu to encrypt directories, other options are available. One alternative uses loopback encrypted filesystems based on the pluggable authentication modules (PAMs) mount module. The eCryptfs system includes both kernel-space and user-space tools.

One advantage of the Ubuntu system is that the password for the user's account is also used as a passphrase for that user's encrypted directory. So unlike with many encrypted partitions, you don't have to enter a passphrase during the boot process.

If you've chosen to encrypt a home directory during the Ubuntu installation process, you'll see the hidden .ecryptfs/ and .Private/ subdirectories in that home directory. You'll also see something similar to the following output from the **mount** command, which suggests that the encrypted files are actually in the .Private/ subdirectory.

```
/home/michael/.Private on /home/michael type ecryptfs
```

Those files are decrypted when they're mounted on the user's home directory using the **ecryptfs** option. One relatively simple way to set up the same type of encryption for user directories is with the **encrypt-setup-private** command. Try this command on a system where a home directory is not encrypted. You don't even need (or want) root administrative access in this case, as the encrypted directory is presumably owned by the user in question. When the **encrypt-setup-private** command is entered, the user is prompted for two things:

- The login passphrase, which is really that user's login password
- A mount passphrase, which can be used to manually remount the directory from a rescue system

If successful, you'll see the hidden directories just described in the noted user's home directory. If there are errors, they may be related to the encryption modules that are or are not loaded. For example, the **lsmod** command run on an Ubuntu system where directory encryption is enabled includes the following output:

Module	Size	Used by
sha256_generic	11191	2
aes_i586	7268	6
aes_generic	26863	1 aes_i586
dm_crypt	11131	1

The loaded modules related to encryption include the following:

- **Secure Hash Algorithm (SHA)** with 256 bits (sha256_generic)
- **Advanced Encryption Standard (AES)** for x86 CPUs (aes_i586)
- **Disk encryption subsystem** (dm_crypt)

One command that may load all these modules simultaneously is the following:

```
# modprobe dm_crypt
```

Encrypted Partitions and Volumes

Although the latest encryption schemes are often associated with more recent kernels, that is not always the case. Red Hat has enabled filesystem encryption during the installation process starting with the release of Red

Hat Enterprise Linux version 5.3. While the previous section focused on an Ubuntu method for encryption, this section focuses on the Red Hat method for encryption, with guidance from the *Red Hat Enterprise Linux 7 Installation Guide*. To that end, Red Hat uses the **Linux unified key setup (LUKS)** disk-encryption specification. It'll work if the `dm_crypt` module is loaded, which can be verified in the output from the `lsmod` command. LUKS is also a user-space tool.

The basic steps are as follows:

1. Create the desired partition, logical volume, or RAID array to be used as an encrypted partition or volume.
2. Based on the device file to be used, fill the device with random data. The `/dev/urandom` device is a Linux-based random number generator. For example, if the partition device is `/dev/vda1`, you'd run the following command:

```
# dd if=/dev/urandom of=/dev/vda1
```

This command may take some time. One test suggests 10 minutes per gigabyte of space being randomized. The reason for filling the drive with random data is to ensure there is nothing left on the drive from any previous storage that might be recovered.

3. Format the device with the **`cryptsetup`** command using the noted LUKS extension. (The **`cryptsetup`** command is used to encrypt entire partitions using the device mapper.)

```
# cryptsetup luksFormat /dev/vda1
```

If you see an error message related to the VIA padlock, the `dm_crypt` and related cryptographic modules may not be loaded.

4. To verify detailed encryption information for the noted device, run the following command:

```
# cryptsetup luksDump /dev/vda1
```

5. The noted partition or volume should now be encrypted. The next step is to set up a device name to be used when the volume is decrypted during the boot process, which you can do with the following command (substitute as desired for `/dev/vda1` and `secret`):

```
# cryptsetup luksOpen /dev/vda1 secret
```

If desired, you can use the `cryptsetup luksUUID /dev/vda1` command to set up a universally unique identifier (UUID) for the encrypted volume.

6. You should now be able to find more information on the noted volume with the `dmesg /dev/mapper/secret` command.
7. Format the device as desired. The following example formats the noted device to the ext3 filesystem format:

```
# mkfs -t ext3 /dev/mapper/secret
```

8. You're now ready to mount the noted `/dev/mapper/secret` device on the directory of your choice. Two more steps are required to make the mount survive a reboot. For the purpose of this exercise, create a directory named `/encrypted`.

9. Create or add information from the partition or volume to the `/etc/crypttab` configuration file. For the parameters given, you'd add the following line to that file:

```
secret /dev/vda1 none
```

If a UUID is used, you can substitute that for the `/dev/vda1` device file. Use the `UUID=uuidnumber` format, where `uuidnumber` is the UUID number created with the `cryptsetup luksUUID /dev/vda1` command.

10. Implement the result in the /etc/fstab configuration file. In this case, you can use the /dev/mapper/secret or the /dev/mapper/luks-*uuidnumber* device filenames.

Local File and Folder Permissions

While the concepts of file ownership and permissions are basic to Linux, these concepts make up the standard discretionary access control system. File permissions are an elementary part of security on Linux. If you're already familiar with file permissions, this section should be a simple review.

Strictly speaking, there are no separate folders in Linux. Everything in Linux is a file. Folders that might exist in file browsers such as Nautilus and Konqueror are based on Linux directories. Directories are just a special kind of file. There are other special kinds of Linux files.

To summarize, just about all you need to know with respect to file and folder permissions is shown in the output from the `ls -l` command. The first letter or dash in each line of output indicates the type of file. For our purposes, you need be aware of three types of files. A dash (-) indicates a regular file; a d indicates a directory, and an l indicates a soft-linked file.

These basic file and folder permissions are known as one method of discretionary access control. In this case, file and folder permissions meet discretionary access control requirements as access is regulated by file and by the identity of users. A second level of discretionary access control is based on access control lists (ACLs), discussed later in this chapter.

Basic File Ownership Concepts

Every file has two owners: a user and a group. In many cases, those owners look like the same user. Based on the user private group scheme, every user is a member of a group of the same name. But any other user can be a member of any group. Just to be clear, take a look at this output from the `ls -l /var/log/cups/access_log` command:

```
-rw----- 1 root lp 10387 Mar 10 14:46 access_log
```

The user owner of this file is root, and the group owner of this file is lp.

File ownership in Linux can be modified with the `chown` and `chgrp` commands. These commands are straightforward. The `chown` command changes the identity of the user who owns a file. The `chgrp` command changes the identity of the group that owns a file. The `chown` command is a bit more capable, as it can change both the user and the group owner of a file simultaneously. For example, the following command assigns the user named donna and the group named users as the owners of the noted file:

```
# chown donna.users somefile
```

For both commands, the `-R` switch changes ownership recursively. For example, if you have to set up user temp1 as a replacement for user worker, you can change the name and ownership of all files in user worker's directory with the following commands:

```
# mv /home/worker /home/temp1
# chown -R temp1.temp1 /home/temp1
```

Basic File-Permission Concepts

Standard Linux file permissions can be assigned in three categories: for the user owner, the group owner, and all other users on the local system. In short, these categories are known as user, group, and others. More Linux permissions are associated with ACLs, described later in this chapter.

Each file may be configured with read, write, and execute permissions for the user owner, the group owner, and every other user. Examine the output from an `ls -l /home` command on a hypothetical directory:

```
drwxr-xr-- 1 root project 100387 Mar 10 12:43 project
```

The first `d` confirms this is a directory. The nine characters that follow specify the permissions for the user, group, and others.

The first trio of characters is `rwx`, which specifies that the user owner, the root administrative user, has read, write, and execute permissions on the `/home/project` directory.

The second trio of characters is `r-x`, which specifies that the group owner, the project group, has read and execute permissions on that directory. The permissions assigned to the group apply to members of that group. Therefore, every user who is a member of the project group has those same read and execute permissions on the `/home/project` directory.

Finally, the last trio of characters is `r--`, which is supposed to provide read permissions to all other users on that directory. But that's not too helpful to those other users. If your account is not root or a member of the project group and you try to run the `ls -l /home/project` command, all you'd see are filenames with question marks where you might expect to see information on owners and permissions.

If you set this up on your system, try the `chmod 751 /home/project` command. (The `chmod` command modifies permissions for a file.) Now log in as a regular user who is not a member of the project group. You should now be able to read the files in the `/home/project` directory with just those executable permissions.

In other words, if you're going to allow anyone but the root administrative user to read a directory, you'll need to make sure the executable bit is set. You'll review how the `chmod` command can be used to modify permissions shortly.

First, look at the output from the `ls -l /etc/grub.conf` command on a Red Hat system:

```
lrwxrwxrwx 1 root root 22 Feb 1 10:44 /etc/grub.conf ->
  ./.boot/grub/grub.conf
```

The first letter in the output confirms this file is a soft link. That's further confirmed by the end of the output, where the `/etc/grub.conf` file actually is redirected to the actual file, `/boot/grub/grub.conf`.

The permissions may appear too generous. You don't want everyone to be able to modify or even see the contents of the local GRUB configuration file. Fortunately, that's not the case. The actual permissions are based on those set on the actual file, `/boot/grub/grub.conf`. On Red Hat systems, permissions are read-write just for the user owner of the file.

Changing File Permissions

The focus of this section is the `chmod` command. The most efficient way to use `chmod` is based on an octal representation of the permissions of the user owner, group owner, and other users. Each type may or may not have read, write, and execute permissions. That's three bits. In a binary system, $2^3 = 8$, which is why such permissions can be represented octally, as described in [Table 5-3](#).

For example, a file with read, write, and execute permissions is given a value of 7, obtained by simply adding together the numerical representations of the desired permissions ($4 + 2 + 1 = 7$).

TABLE 5-3 Permissions represented in octal format.

PERMISSION	NUMERICAL REPRESENTATION
r	4
w	2
x	1

The user owner of a file can use the **chmod** command to change permissions on a file. To change the permissions for the user, group, and others in a file, you need three numbers. For example, look at the following command, where **localfile** is some arbitrary file in the local directory:

```
$ chmod 754 localfile
```

The first **7** represents read, write, and execute permissions for the user owner of the file. The **5** that follows represents read and execute permissions for the group owner of the file. The final **4** represents just read permissions for all other users.

The **chmod** command can also represent and change the set user ID (SUID), set group ID (SGID), and sticky-bit permissions. These special bits are also represented numerically, in octal format, as shown in [Table 5-4](#).

TABLE 5-4 Special permissions represented in octal format.

SPECIAL PERMISSION	NUMERICAL REPRESENTATION
SUID	4
SGID	2
Sticky bit	1

technical TIP

Default permissions depend on the current value of the **umask**, available in the output from the **umask** command. For example, the default value of **umask** for regular Ubuntu users is **0022**. Despite the description of the **chmod** command and special permissions, the first number is ignored. The second number applies to the regular user owner, and presumably masks nothing.

With this **umask**, user owners of a new directory get full read, write, and execute permissions. However, user owners of new files get only read and write permissions. Execute permissions on new regular files are disabled by default.

If you're working in octal format, the **chmod** command can be used not only to set permissions for all users, but also to set special permissions for all users. These special permissions become the first number in a **chmod** command. So hypothetically, if you were to run the following command:

```
$ chmod 5764 testfile
```

It would set the SUID and sticky bit on the **testfile**. It would assign read, write, and execute permissions to the user owner. It would assign read and write permissions to the group owner. Finally, it would assign read-only permissions to all other users.

If you just want to change permissions for a single user or group, or just for other users, a different format of **chmod** switches can help. As this is just another way to set permissions, here are a couple of examples:

```
# chmod u+wx filename
```

The noted command adds write and executable permissions to the noted filename.

```
# chmod o+t newdirectory
```

The noted command adds the sticky bit to the directory named **newdirectory**.

Networked File and Folder Permissions

Whatever permissions exist on the local and client systems, the actual permissions that are transmitted and available are based on those configured in the server configuration file. In Linux, three major services that network files and folders (which are really directories) are the Network File System (NFS), Samba, and the File Transfer Protocol (FTP). In fact, the file and directory permissions that you see from shared directories may be overridden by those file and directory permissions configured in those services.

NFS Issues

While three major versions of NFS are still in common use, two are inherently unsecure. NFS versions 2 and 3 do not require any sort of user authentication when a client connects to a shared directory. In other words, the only access controls for NFS versions 2 and 3 services are based on domain names and/or IP addresses. All an unauthorized user has to do is find an open port on a hub, switch, or router; connect his or her laptop; find an appropriate IP address; and connect to your NFS shared directory.

Basic NFS Security Issues

By default, all current versions of NFS prevent some security issues, such as root administrative access to files and directories. However, that is not assured if the wrong options are set in the /etc/exports configuration file.

Some special configuration may be required to minimize risks. For example, while unknown users (and the local root administrative user) are supposed to be redirected to an account like **nobody** (the default account for users on certain configured file-sharing servers) or **nfsnobody** (the default account for unauthorized users who connect to an NFS file-sharing server), the UID associated with that “safe” account may vary by distribution. In some cases, that UID may be 65534; in other cases, that UID may be 4294967294.

In either case, you should check that account in the files of the password authentication database. For example, this excerpt from one version of /etc/passwd may be a risk, as an unauthorized user who gains access to the nfsnobody account will see a shell if a login is allowed.

```
nfsnobody:x:4294967294:4294967294:  
→ Anonymous NFS User:/var/lib/nfs:/bin/sh
```

Additional problems relate to different user-authentication databases. For example, while the first user on an Ubuntu system has a UID of 1000, the first user on a Red Hat system has a UID of 500. So even if those users have identical names and passwords, the user on one client won’t be able to access his or her files on a shared directory from a second client.

NFS Permissions and Authentication

All versions of NFS set up shares with read-only or read-write access. Whatever access is allowed in the NFS /etc/exports configuration file supersedes any access of actual files. In fact, when you run the `ls -l` command on an NFS-mounted directory, the permissions you see don’t have to match the permissions configured in the /etc/exports file.

NFS version 4 (NFSv4) systems, when properly configured, are somewhat more secure. They can be used with a Kerberos ticket server to authenticate users.

NFSv4 also supports ACLs. As with regular ACLs on Linux, NFS-based ACLs can override standard read, write, and executable permissions for files on shared directories.

Samba/CIFS Network Permissions

Samba is the Linux implementation of Microsoft networking. As Microsoft has added to its systems with the Common Internet File System (CIFS), Samba developers have kept pace. As a result, current Linux systems can share directories and printers on a Microsoft-based network. It’s even possible to have a Samba-based network of

nothing but Linux computers. The focus of this section is the main Samba configuration file, **smb.conf**, in the `/etc/samba/` directory. That configuration file is split into two sections, “Global Settings” and “Share Definitions.”

As with NFS, the configured options in the Samba configuration file supersede any regular local permissions for the shared files and directories.

The standard share in Samba is of user home directories. It’s associated with the [homes] stanza. In general, default options in this stanza allow an authenticated user to access his or her home directory on the local system. However, that directory is not shared—and should not even be visible to other users, courtesy of the following directive:

```
browseable = no
```



NOTE

Some Samba directives may appear to be misspelled. Sometimes, multiple spellings are allowed; for example, **browsable** and **browseable** have the same meaning in a Samba configuration file.

Some directives are functionally identical. For example, you may see one of the following default directives in the [homes] stanza:

```
read only = yes
writable = no
```

Of course, if you want to allow users write access to their home directories, these directives can be changed. If you’ve set up write access in your Samba configuration file, the following directives apply to the creation of files and directories, respectively:

```
create mask = 0700
directory mask = 0700
```

The **create mask** value sets up read and write permissions for the user owner of new files. The **directory mask** value sets up read, write, and execute permissions for the user owner of new directories.

While the [homes] share is visible only to the owner of a home directory, anyone can still connect to that home directory by default. You can prohibit such access with the following directive:

```
valid users = %S
```

Other directives associated with shared directories that you might consider or watch out for include those listed in [Table 5-5](#).

Several Samba directives can be configured to point to users. Those directives can also point to groups. For example, the following directive limits write access to the users who are members of the admin group:

```
write list = @admin
```

Alternatively, the following directive would prohibit access from a group that is often given administrative access on some systems:

```
invalid users = @wheel
```

TABLE 5-5 Samba directives related to shared directories.

SAMBA DIRECTIVE	DESCRIPTION

acl check permissions	Checks ACL-based permissions from a Microsoft client
acl group control	Allows users of a group owner to change permissions and ACLs
admin users	Lists users with full administrative privileges
guest ok	Configures the share so no password is required; synonymous with public
hosts allow	Sets hostnames or IP addresses of allowed systems; opposite of hosts deny
locking	Supports a lock file to prevent simultaneous access to the same file
path	Sets a directory path for the share
printer admin	Lists users allowed to administer printers
write list	Lists users allowed read-write access to a share

Network Permissions for the vsftpd Daemon

Several FTP services are commonly used on many Linux systems. However, the very secure FTP daemon is acknowledged as a secure option. To keep things simple, this section focuses on file-sharing security options associated with that service. Although this section focuses on vsftpd, many of the same principles apply to other FTP services.

The main vsftpd configuration file is **vsftpd.conf**. Depending on the distribution, it may be found in the /etc/ or /etc/vsftpd/ directory. Most of the standard directives in this file have a direct bearing on how files are shared. While most FTP servers allow anonymous access, the following directive suggests that it is disabled by default:

anonymous_enable=NO

If you do enable anonymous access to an FTP server, it becomes especially important to isolate the directories associated with that server. As described earlier, it's an excellent idea to isolate the directory with the FTP server in a separate partition or volume. One advantage is that it configures a chroot jail by default for anonymous users. The chroot jail directory is configured as the home directory of the ftp user, as defined in the /etc/passwd file. A *chroot jail* is a special way of confining a program to a specific part of the filesystem. If anyone were to compromise the FTP server application, that person could never get to any of the system files because they would be locked within that particular directory. The *chroot* in chroot jail means change root. It suggests that you are changing the root of the filesystem to be a new directory, effectively capping you within that one directory tree.

FYI

The chroot jail can help prevent malicious users or attackers from getting to the key files on your systems. However, it isn't perfect. Some FTP developers believe that any sort of chroot jail presents its own kinds of risks. The directory used for the chroot jail could conceivably be used as the root for configuration files, commands, and libraries, enabling that user to create his or her own commands that may be able to escape from the chroot jail. Disabling executable access within a chroot jail can help prevent the use of such commands.

You can also set up the FTP server to allow users access to their home directories. This directive enables this:

local_enable = YES

If you enable local access, it's important to set up a chroot jail for those users. That's made possible with the following directive:

```
chroot_local_user = YES
```

Other relevant directives from standard versions of the vsftpd.conf file are shown in [Table 5-6](#).

Configuring and Implementing Quotas on a Filesystem

Quotas can limit the resources taken by a user or by a group of users. When appropriate, quotas are enabled. Limits are configured on the amount of space based on blocks and on the number of files based on **inodes**. (An inode is a data structure associated with a file.) Quotas can protect the space available on critical directories. Quotas can give you time if there's a legitimate reason to expand the space allocated to a directory such as /home/. Quotas can help limit the damage if one of the accounts on your system has been compromised.

Poorly configured logs or message systems can create huge numbers of files. For example, systems configured in debug mode create logs for every event. Systems configured to create, say, pictures every second can create tens of thousands of files every day. Such systems could easily affect the life of hard-drive media.

Although quotas may not directly enhance system security, they can certainly help limit damage should there be a security breach. In the following sections, you'll see how to configure quotas on a selected filesystem, the commands used to manage quotas, and how to get quota reports on your users.

TABLE 5-6 vsftpd security-related service directives.

DIRECTIVE	DESCRIPTION
anon_upload_enable	Allows uploads by anonymous users; can be set to NO
anon_mkdir_write_enable	Allows anonymous users to create directories; can be set to NO
chroot_list_file	Sets up a list of users who are limited to their home directories
local_umask	Configures permissions for created files and directories
pam_service_name	Specifies the PAM configuration file in the /etc/pam.d/ directory
rsa_cert_file	Specifies the RSA certificate file
rsa_private_key_file	Specifies the file with the RSA private key
secure_chroot_dir	Configures a directory that should not be writable by the ftp user
userlist_enable	Configures a list to allow or deny users
write_enable	Allows users to write to a server; can also apply to anonymous users

The Quota Configuration Process

To set up a quota on a filesystem, you'll need to mount it appropriately. User and group quota options for the **mount** command are **usrquota** and **grpquota**. Mount options are configured in the /etc/fstab configuration file. Any filesystem that may be written to by users is a good candidate for quotas. In that respect, the prime user filesystem is associated with the /home/ directory.

If you haven't configured a separate filesystem for the /home/ directory, the following steps apply to the top-level root directory (/). In the /etc/fstab file, you'd add the **usrquota** and **grpquota** options to the fourth column. As an example, assume you have a system where /home/ is mounted on the /dev/sda2 partition. The appropriate line in /etc/fstab would look similar to the following:

```
/dev/sda2 /home ext4 defaults,usrquota,grpquota 1 2
```

If you've just made this change and don't want to reboot the system, run the following command, which refers to /etc/fstab for required options to the `mount` command:

```
# mount -o remount /home
```

If successful, you'll see it in the output from the `mount` command. For this particular configuration, you'd see the following output:

```
/dev/sda2 on /home type ext4 (rw,usrquota,grpquota)
```

Next, you'll need to create the actual quota configuration files with the `quotacheck -cugm /home/` command. (The `quotacheck` command creates, checks and repairs quota-management files.) That command combines four switches that (-C) create new quota files, (-U) apply the quota files to users, and (-G) apply the quota files to groups, (-M) without remounting the filesystem. If successful, it'll add `aquota.user` and `aquota.group` files to the /home/ directory. These are the group quota-management and user quota-management files, respectively. The quotas that you create will be stored in these files.

Quota Management

Linux quotas are based on the concept of soft and hard limits. Although the soft limit is a permanent limit, users and groups are allowed to exceed that limit for a grace period. In contrast, users are not allowed to exceed a configured hard limit.

After you've configured the basic quota files, you're ready to edit the quota of a specific user. As an administrative user, run the `edquota -u username` command. The `edquota` command is used to check and edit user quotas. The -u stands for user and is the default for the `edquota` command. If you want to edit the quota for a group, run the `edquota -g groupname` command.

The command as shown opens a quota configuration file in the default editor for the distribution. If this is the first time you've configured quotas, the screen may appear confusing, as the columns are wrapped. [Figure 5-2](#) displays quotas in a slightly expanded screen. In the case shown, user michael has files of nearly 700,000 blocks. Because each block is 1 KB, that user currently has about 700 MB of files on 305 inodes on the /dev/sda2 partition.

If you wanted to limit user michael to 1 GB of space and 10,000 files, you could set up the following quotas:

Disk quotas for user michael (uid 500):

Filesystem	blocks	soft	hard	inodes	soft	hard
/dev/sda2	698172	1000000		0	305	10000

In this case, the soft limit becomes the implicit hard limit. But that's something you may want to do unless the rules are crystal clear. After all, people make mistakes. So you may want to set hard limits as well, with a number a bit higher than the soft limit.

Disk quotas for user michael (uid 500):

Filesystem	blocks	soft	hard	inodes	soft	hard
/dev/sda2	698172	0	0	305	0	0

-

-

FIGURE 5-2

Editing a user quota.

```
GNU nano 2.0.7           File: /tmp//EdP.aZ2Kq3z           Modified

Grace period before enforcing soft limits for users:
Time units may be: days, hours, minutes, or seconds
Filesystem          Block grace period    Inode grace period
/dev/sda1           7days                  7days

^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is   ^V Next Page  ^U UnCut Text ^T To Spell
```

FIGURE 5-3

Quota grace periods.

One example is shown here, where the hard limit is 10 percent above the soft limit:

Filesystem	blocks	soft	hard	inodes	soft	hard
/dev/sda2	698172	1000000	1100000	305	10000	11000

A user is allowed to exceed the soft limit for a grace period. After the changes are saved, you can review the current grace period with the `edquota -t` command. The result is shown in [Figure 5-3](#) and is almost self-explanatory. The default grace period is seven days. A user is allowed to exceed the soft limit, but not the hard limit, for those seven days. After the grace period has passed, the user has to be back under the soft limit or that user will no longer be able to write to disk. You can change the grace period as shown, in days, hours, minutes, or seconds.

If you're satisfied with the limits set on the first user, you can copy those quotas to other users. The following command uses the quota configured for user michael as a (-p) prototype for the users who follow in the list:

```
# edquota -up michael kim mike larry kate
```

If you've set up quotas for one group and want to transfer those same quotas to more groups, the command would be similar. The following command uses the quotas set for group1 and copies them for the groups named group2 and group3:

```
# edquota -gp group1 group2 group3
```

Quota Reports

Once quotas have been put into place, administrators will want to get periodic reports. You may want to warn any users who are near their quotas. Alternatively, if lots of users are near their quotas, you may consider getting more storage and raising the limits. To review the current status for all users, run the following command, which leads to output similar to that shown here:

```
# repquota -a
*** Report for user quotas on device /dev/sda2
Block grace time: 24:00; Inode grace time: 24:00
                                Block limits                      File limits
User        used   soft   hard   grace   used   soft   hard   grace
dickens   +-     124    100    110    1days   15     0     0     1days
```

As shown, user dickens has exceeded her soft and hard limit. The grace period shown is 24 hours. If you receive a call from that user asking why she can't save files, this information tells you why.

How to Configure and Implement Access Control Lists on a Filesystem

One area where Linux did not initially include the same level of security as some other operating systems was in ACLs, which allow you to set different permissions for specific users and groups. Linux developers started to address this flaw back in 2002, so ACL support in Linux is mature. It's available for the standard Linux filesystems, including ext2, ext3, ext4, reiserfs, and xfs. As suggested earlier, it's also available for directories shared through NFS. The ACLs available for Samba are somewhat different, as they are based on the functionality developed for Microsoft operating systems.

ACLs provide a second level of discretionary access control. The ACL options you set can override the basic discretionary access control options described earlier in this chapter. In other words, ACL settings can supersede basic read, write, and execute permissions for the user and group owners of files and directories.

In this section, you'll see how filesystems can be configured with ACLs, review available ACL commands, and learn how to use these commands to set ACLs on files and directories.

Configure a Filesystem for ACLs

The process to configure a filesystem for ACLs is similar to the process for configuring a filesystem for quotas. The appropriate **mount** option to configure ACLs is **acl**. To implement ACLs permanently, configure it on appropriate filesystems in the /etc/fstab configuration file. Any filesystem that requires custom access by certain users is a good candidate for ACLs. The prime candidate for such customization is the /home/ directory.

If you haven't configured a separate filesystem for the /home/ directory, the following steps apply to the top-level root directory (/). In the /etc/fstab file, add the **acl** option to the fourth column. As an example, suppose you have a system where /home/ is mounted on the /dev/sda2 partition. The appropriate line in /etc/fstab would look similar to the following:

```
/dev/sda2 /home ext4 defaults,acl 1 2
```

If you've just made this change and don't want to reboot the system, run the following command, which refers to /etc/fstab for required options for the **mount** command:

```
# mount -o remount /home
```

If successful, you'll see it in the output from the **mount** command. For this particular configuration, you'd see the following output:

```
/dev/sda2 on /home type ext4 (rw,acl)
```

ACL Commands

The two standard ACL commands are **getfacl** and **setfacl**. These commands are almost self explanatory, as they get and set the ACL settings for the current file. These commands work only in filesystems mounted with the **acl** option.

If you've configured the /home/ directory filesystem with ACL support, you should be able to run the **getfacl** command on the file or subdirectory of your choice in that filesystem. If you want to go farther, the following applies the **getfacl** command recursively on the home directory of user michael:

```
$ getfacl -R /home/michael
```

**TIP**

If you need to collect files from an ACL-enabled filesystem in an archive, consider the `star` command. It is sort of an enhanced `tar` command that can collect and restore ACL information to and from an archive.

Next, there's the `setfacl` command. It includes two basic options: `-m` to set or modify ACL rules and `-x` to remove ACL rules. ACL rules can be configured in four categories:

- **`u:user:permissions filename`**—Sets ACL permissions for the specified user, by username or UID number
- **`g:group:permissions filename`**—Sets ACL permissions for the specified group, by groupname or GID number
- **`m:permissions filename`**—Sets the standard effective rights mask for ACL permissions for users and groups
- **`o:permissions filename`**—Sets ACL permissions for users who are not members of the group that owns the file

Configure Files and Directories with ACLs

In this section, you'll see how ACLs can be configured, and how they can make a difference. You'll use the `setfacl` command to override standard discretionary access control permissions and ownership. Before running the steps in this section, make sure of the following:

- The `acl` package is installed. The package has the same name on both Red Hat and Ubuntu systems.
- The partition, volume, or RAID array with the `/home/` directory is mounted with ACLs, as described in the previous section.
- At least two regular users are configured, with files owned by each user in their own home directories.

First, run the `ls -l` and `getfacl` commands on the file or directory of your choice. Unless you've already set an ACL, the output from both commands should confirm the same information.

Now select a different user—someone who does not own the file or directory just reviewed. Make sure that user is not a member of the group that owns the file. For example, the following command provides read, write, and execute permissions on the `ITTIS418/` subdirectory:

```
$ setfacl -m u:donna:rwx ITTIS418/
```

Now you'll have to give user `donna` ACL execute permissions on the underlying directory—in this case, `/home/michael/`:

```
$ setfacl -m u:donna:x /home/michael/
```

These commands can be run by the user owner of the `ITTIS418/` subdirectory. No system-administrative privileges are required. You can confirm these ACL rules with the `getfacl` command, applied to the noted directories. Now you can further protect the `ITTIS418/` subdirectory. The following command limits access to the user who owns that directory:

```
$ chmod 700 ITTIS418/
```

Despite the limitation in standard user privileges, the configured ACLs mean that user `donna` has full read, write, and execute access to the files in the `/home/michael/ITTIS418/` directory. If you want to remove those ACL rules, run the following commands:

```
$ setfacl -x u:donna ITTIS418/
$ setfacl -x u:donna /home/michael/
```

Best Practices: Filesystems, Volumes, and Encryption

If you understand the FHS, you'll know which directories can be mounted using mount points on separate volumes. More importantly, you'll know which directories should be mounted on separate mount points to help protect the system in case a malicious user manages to compromise your system. Candidate directories for separate mount points include `/boot/`, `/home/`, `/opt/`, `/srv/`, `/tmp/`, and `/var/`. Appropriate subdirectories can also be considered for separate mount points.

The right filesystem format can also help protect the system in case of a break-in. Journaling filesystems such as `ext3`, `ext4`, `xfs`, and `reiserfs` can speed recovery in case of problems. However, journaling takes extra space and may therefore be unnecessary for smaller filesystems.

In Linux, you can encrypt individual files, directories, and entire filesystems. Encryption tools can be split into user-space and kernel-space tools. Three of many available Linux encryption tools, or commands, are `gpg`, `cryptfs`, and `cryptsetup`. In this chapter, you saw how to use these tools to encrypt files, directories, and filesystems, respectively.

Local file and folder permissions provide one level of discretionary access control. The associated read, write, and execute permissions apply to the file's user owner, the file's group owner, and all other users. You can change ownership with the `chown` and `chgrp` commands. You can change permissions with the `chmod` command.

File and folder permissions differ when shared over a network. In fact, the file and folder permissions that you see may not work, as they may be overridden by the configuration of the network sharing service. NFS sharing has other security issues, unless you use NFSv4 with Kerberos-based authentication. Samba shared directories include a number of options for configuring how files are shared. Directories shared with the vsftpd daemon service can be further protected with a chroot jail.

If an unauthorized user does compromise one of your systems, quotas can limit damage. Once configured on a filesystem, quotas can be configured by user or group. Quotas can limit the amount of space or number of inodes taken by a user or group. You can configure hard and soft limits in each category, buffered by a grace period. You can review what users and groups do with regular quota reports.

With ACLs, you can take discretionary access controls to another level. ACLs can be configured using the same files and commands as quotas. Once configured, you can review current ACLs with the `getfacl` command. If you configure ACL rules for a specific user, you can further limit access to key files and directories to other users.



CHAPTER SUMMARY

In this chapter, you examined the FHS with a focus on what directories to configure on separate filesystems. You should now know more about Linux options for filesystem formats, especially in the context of journaling. You skimmed the surface of available Linux encryption tools for files, directories, and filesystems.

You also learned about the differences between filesystems, partitions, volumes, and mount points. These are sometimes subtle in nature. Although there are other definitions, the primary definition of a filesystem is the format used to keep track of files and directories and all the metadata associated with them. A partition is a portion of a hard drive. A volume is more of a logical term because it refers to any entity that can be used to contain a filesystem, meaning any single entity that can be formatted. This might be a single partition on a hard drive but it might also be a RAID device or some other type of multi-device storage.

Discretionary access controls on Linux start with regular file and folder ownership and permissions. The file and folder permissions for networked directories can be customized on NFS, Samba/CIFS, and vsftpd services. Any damage to a system can be limited by user and group quotas on space and inodes. Discretionary access controls can be taken a step further with ACLs.



KEY CONCEPTS AND TERMS

Advanced Encryption Standard (AES)
aquota.group
aquota.user
chgrp
chmod
chown
cryptsetup
Digital Signature Algorithm (DSA)
Disk encryption subsystem
cryptfs
edquota
Elgamal
Enterprise cryptographic filesystem (eCryptfs)
ext2
ext3
ext4
fdisk
Filesystem
Filesystem Hierarchy Standard (FHS)
getfacl
gpg
grpquota
Inodes
Journaled filesystem
Kernel-space tools
Linux unified key setup (LUKS)
mkfs
Mount point
nfsnobody
nobody
quotacheck
reiserfs
root
RSA
Secure Hash Algorithm (SHA)
setfacl
smb.conf
Software RAID
User-space tools
usrquota
vsftpd.conf

xfs

CHAPTER 5 ASSESSMENT

- 1.** Which of the following directories are suitable for separate mount points? (Select two.)
 - A. /etc/
 - B. /home/
 - C. /lib/
 - D. /var/
- 2.** Which of the following directories typically includes files associated with third-party applications?
 - A. /etc/
 - B. /home/
 - C. /opt/
 - D. /usr/
- 3.** Which of the following directories is most well suited as a read-only mount point?
 - A. /boot/
 - B. /home/
 - C. /mnt/
 - D. /srv/
- 4.** Which of the following filesystem formats is best suited for a smaller volume?
 - A. ext2
 - B. ext3
 - C. ext4
 - D. reiserfs
- 5.** The command that lists currently loaded GPG keys is _____.
- 6.** Which of the following directories contain GPG private and public keys?
 - A. .gpg
 - B. .gpgkeys
 - C. .gnupg
 - D. .keys
- 7.** Which of the following commands is associated with the Linux unified key setup disk encryption specification?
 - A. **dcrypt**
 - B. **ecryptfs**
 - C. **gpg**
 - D. **cryptsetup**
- 8.** Which of the following commands prohibits access from all users except the user owner and members of the group that owns the file named filename?
 - A. **chmod 770 filename**
 - B. **chmod 707 filename**
 - C. **chmod 077 filename**
 - D. **chmod 007 filename**
- 9.** Which of the following commands sets the SUID bit on the file named filename?
 - A. **chmod 1770 filename**

- B. `chmod 2750 filename`
 - C. `chmod 4555 filename`
 - D. `chmod 3777 filename`
- 10.** If you try to change files remotely on a shared NFS directory as the root administrative user, what happens?
- A. The change fails because the root user on one system is the nobody user on another system.
 - B. The change is successful.
 - C. The change is successful even if the NFS directory is shared in read-only mode.
 - D. The change fails unless you log in with the root administrative password from the remote system.
- 11.** Which of the following Samba directives specify permissions of files created on a shared network directory?
- A. `create_octal`
 - B. `create_mask`
 - C. `create_options`
 - D. `create_write`
- 12.** Which of the following directories is appropriate for quotas?
- A. `/etc/`
 - B. `/home/`
 - C. `/opt/`
 - D. `/usr/`
- 13.** Which of the following commands lists quota usage by user?
- A. `quota`
 - B. `repquota`
 - C. `quotacheck`
 - D. `quotarep`
- 14.** What configuration file is used to configure ACLs for a filesystem?
- A. `/etc/fstab`
 - B. `/etc/acl`
 - C. `/etc/pam.d/acl`
 - D. `/etc/filesystems`
- 15.** The command that lists the current ACL rules on the local file named test1 is _____. Assume your user account is the owner of file test1.

CHAPTER

6 Securing Services

O

NCE UPON A TIME, when you booted up your computer, you could run a single program. Of course, you had a choice of which program you wanted to run, but you could run only one program at a time. When that program was finished, you could run another or run the same one all over again—whichever you chose.

At some point, both large-scale computers and personal computers reached the point where they could run terminate-and-stay-resident programs. These programs would launch and then release the operating system back to the user to run another program, leaving the terminate-and-stay-resident program sitting in memory. This type of system was very rudimentary.

Today, modern operating systems allow users to run a number of programs at any given point. This is made possible by **multitasking**. Multitasking operating systems allow multiple programs to appear to be running simultaneously. This is aided by the use of multiple processors or multiple cores that look like multiple processors within a single central processing unit (CPU). In reality, time in the CPU is shared across all programs, but the CPU switches between the programs so quickly that it appears that the programs are all running at the same time. It's impossible to distinguish between this type of multitasking and having an individual CPU for every program that is running.

But having multiple processors does mean that more than one process can run at any given time. Both user- and system-level processes can take up time in the processor because as far as the processors are concerned, each one looks like the others.

For decades, Unix-like operating systems like Linux have used specialized programs called *daemons* to support the system's functionality. Daemons (pronounced *demons*) provide this functionality to users or other programs while running in the background. As an example, a Web server provides a service to remote users, sending them Web pages and other related content. The Web server runs in the background without any direct interaction with a user logged into the system on which it's running. These services can have vulnerabilities. Like any software, they could expose you to risk, particularly if they are running.

Chapter 6 Topics

This chapter covers the following topics and concepts:

- How to start a hardened system
- How to manage services
- How to harden services
- How to use mandatory access controls
- What the differences are between servers and desktops
- How to protect against development tools

Chapter 6 Goals

When you complete this chapter, you will be able to:

- Install a Linux distribution with minimal software
- Enable and disable services
- Determine whether to use a GUI
- Minimize the use of development tools to test systems

Starting a Hardened System

Hardening is the process of locking down a system to protect it and any resources that may reside on it. These resources include the data, the services, and the system itself.

A common goal of malicious users is simply to acquire more systems for their computing power. Attackers may be looking to get access to one of your systems simply to have it under their control. Some Linux distributions have a minimal installation option, which should be the bare minimum when it comes to software packages. This would be just enough for you to be able to log in, perform basic functions, add software, and administer the system. You won't be able to do a lot with a system with a minimum number of software packages, but there will be very few ways for an attacker to access it.

The advantage of a minimal installation is that you can always add software after the fact. What is considerably harder is paring down software after you install a complete system. When you add a piece of software, it may depend on other types of software, so those other software packages must be installed before the software you want will work. As an example, the graphical version of the Vi iMproved (Vim) editor has a number of dependencies that are not installed on the Linux Mint system. You can use the apt-get utility to install additional software and to check dependencies. As you can see in the following code, six additional software packages must be installed with Vim. In addition, a handful of additional software packages are recommended.

```
kilroy@hodgepodge ~ $ sudo apt-get install vim-gtk
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  libruby1.9.1 libtcl8.6 libyaml-0-2 ruby ruby1.9.1 vim-gui-common
  ↗ vim-runtime
Suggested packages:
  tcl8.6 ri ruby-dev ruby1.9.1-examples ri1.9.1 ruby1.9.1-dev
  ↗ ruby-switch
  cscope vim-doc
The following NEW packages will be installed:
  libruby1.9.1 libtcl8.6 libyaml-0-2 ruby ruby1.9.1 vim-gtk
  ↗ vim-gui-common
  vim-runtime
0 upgraded, 8 newly installed, 0 to remove and 242 not upgraded.
Need to get 9,630 kB of archives.
After this operation, 45.4 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

In the preceding example, the apt-get utility is used to install the package with the `install` keyword. The apt-get utility determines that a number of additional packages must be installed for vim-gtk to work. It notes that it's going to install six additional packages, as indicated by the message "The following extra packages will be installed." After that, it offers some suggestions for other packages that you might consider installing—packages

related to the package you are installing. Finally, apt-get displays the entire list of packages that will be installed before asking you to confirm that this is what you want to do.

If you were to remove the vim-gtk package, none of the other packages would be removed along with it. After all, just because you uninstall vim-gtk doesn't necessarily mean you don't want to keep the other packages. As an example, the ruby package was installed as part of the installation of vim-gtk. Uninstalling vim-gtk will leave ruby installed on the system.

Most users don't keep track of all the dependencies that come with a piece of software. If you were to remove vim-gtk, the only software that would be uninstalled would be vim-gtk, as you can see in the following code.

```
kilroy@hodgepodge ~ $ sudo apt-get remove vim-gtk
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be REMOVED:
  vim-gtk
0 upgraded, 0 newly installed, 1 to remove and 242 not upgraded.
After this operation, 2,623 kB disk space will be freed.
Do you want to continue? [Y/n]
```

On Red Hat operating system installations like Fedora Core, Red Hat Enterprise Linux, and CentOS, there are other means of installing packages. In the early years of Red Hat, rpm was the name of the program used to install **Red Hat Package Manager (RPM)** packages. RPM is a format used not only to include all the files associated with a program, but also to include the metadata associated with the package's contents and any dependencies the package may have.

One problem with the rpm program, though, is that it doesn't handle the automatic installation of dependencies. If you were to install an RPM package, the rpm utility would merely tell you what you were missing. It would be up to you to download and install any packages that were required for the package you want to work.



NOTE

Yellowdog is a Linux distribution based on Red Hat, meaning it uses the same package-management system and service-management features. Introduced in 1999, it's still maintained, but it focuses on the PowerPC processor rather than the Intel processor on which Red Hat focuses. The original Yellowdog utility on which yum is based is Yellowdog Updater (yup).

Instead of adding dependency installation to the rpm utility, Red Hat introduced a utility taken from another Linux distribution and altered to add functionality. The **Yellowdog Updater, Modified (yum)** utility downloads the RPM files from a network site, identifies the packages that must be installed before the package you really want is installed, and then installs everything, in the correct order.

When you install a package on a Red Hat-based system using yum, you end up with a list of all the packages you need to ensure the one package you want functions properly. This can have a cascading effect, however. If package A has a dependency for package B, but package B requires both packages C and D, the installation of package A will require the installation of packages B, C, and D.

As an example, installing Tomcat on a very limited installation of CentOS or Red Hat Enterprise Linux will result in the list of packages in [Figure 6-1](#), which must be installed. **Tomcat** is a Java application server used to handle Web-based requests and then hand them to an application written in Java that the Tomcat server runs. You can see that the list of dependencies is extensive, all based on a single package.

```

Dependencies Resolved (base : http://www.LinuxCentOS.com/13.0.37-generic-x86_64)
=====
Package          Arch    Version           Repository  Size
=====
Installing:
tomcat           noarch  7.0.54-2.el7_1   updates      85 k
Installing for dependencies:
apache-commons-collections  noarch  3.2.1-21.el7   base        506 k
apache-commons-daemon        x86_64  1.0.13-6.el7   base        54 k
apache-commons-dbcp          noarch  1.4-17.el7    base       167 k
apache-commons-logging        noarch  1.1.2-7.el7   base        78 k
apache-commons-pool          noarch  1.6-9.el7    base       113 k
avalon-framework            noarch  4.3-10.el7   base        88 k
avalon-logkit               noarch  2.1-14.el7   base        87 k
ejc                  x86_64  1:4.2.1-8.el7  base       1.4 M
geronimo-jms              noarch  1.1.1-19.el7  base        31 k
geronimo-jta               noarch  1.1.1-17.el7  base        20 k
java-1.8.0-openjdk          x86_64  1:1.8.0.45-30.b13.el7_1  updates   213 k
java-1.8.0-openjdk-headless x86_64  1:1.8.0.45-30.b13.el7_1  updates   31 M
javamail                   noarch  1.4.6-8.el7   base       758 k
libXfont                  x86_64  1.4.7-2.el7_0  base       144 k
libfontenc                x86_64  1.1.1-5.el7   base        29 k
log4j                     noarch  1.2.17-15.el7  base      443 k
tomcat-el-2.2-api          noarch  7.0.54-2.el7_1  updates   76 k
tomcat-jsp-2.2-api         noarch  7.0.54-2.el7_1  updates   90 k
tomcat-lib                 noarch  7.0.54-2.el7_1  updates   3.7 M
tomcat-servlet-3.0-api     noarch  7.0.54-2.el7_1  updates   207 k
ttmkfdir                  x86_64  3.0.9-41.el7  base        47 k
xalan-j2                  noarch  2.7.1-23.el7  base       1.9 M
xerces-j2                  noarch  2.11.0-17.el7_0  base       1.1 M
xml-commons-apis            noarch  1.4.01-16.el7  base      227 k
xml-commons-resolver        noarch  1.2-15.el7   base       108 k
xorg-x11-font-utils         x86_64  1:7.5-18.1.el7  base       87 k
xorg-x11-fonts-Type1        noarch  7.5-9.el7    base      521 k

Transaction Summary
=====
Install 1 Package (+27 Dependent packages)

Total download size: 43 M
Installed size: 118 M
Is this ok [y/d/N]: 

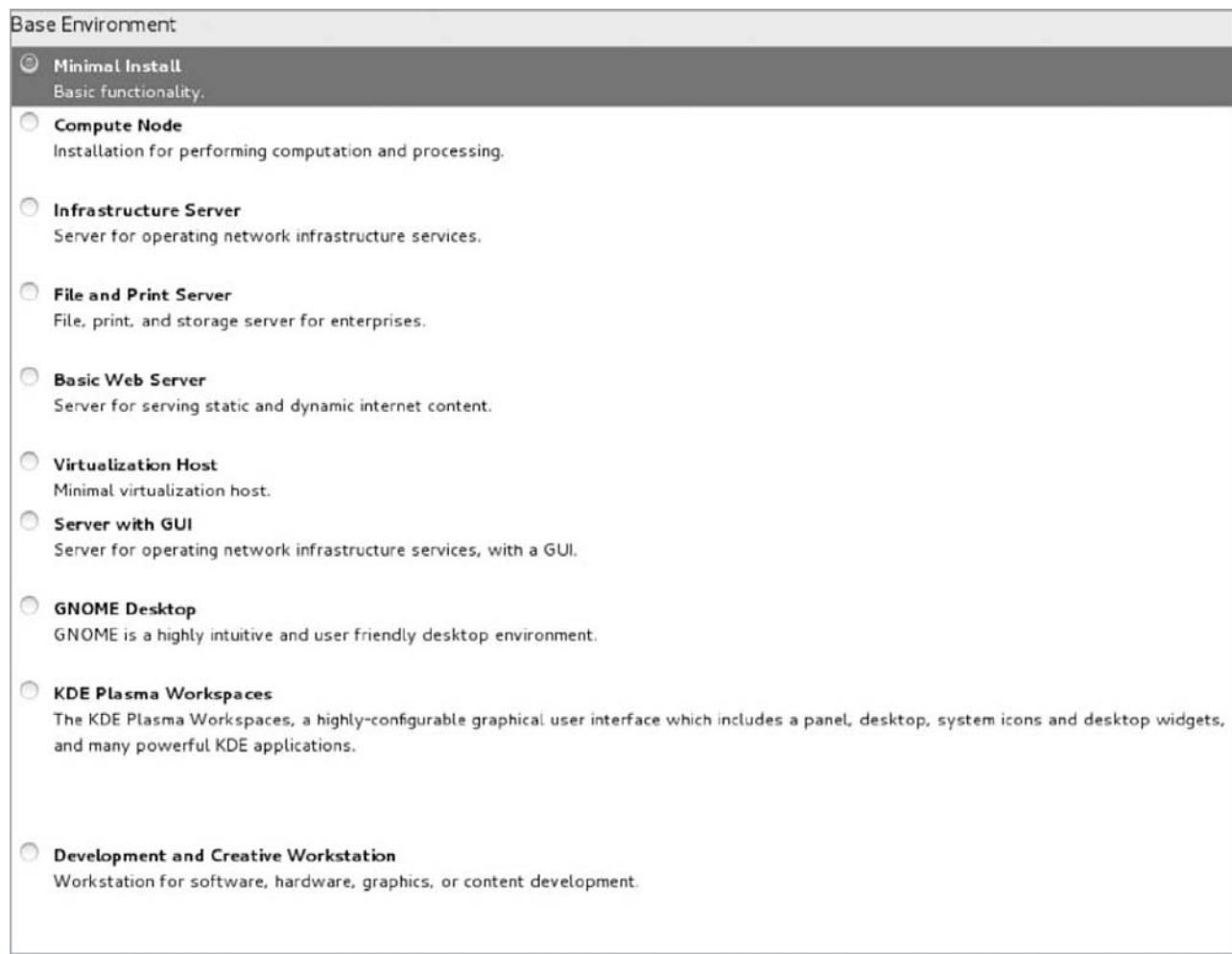
```

FIGURE 6-1

Tomcat dependencies.

What does all this mean? It means that when trying to protect yourself, especially on systems that face the Internet, the best approach is to start with a minimal installation and add only software that is absolutely necessary. Once you have added software or started with a larger installation base, it's considerably harder to scale back to a smaller number of software packages.

So the short answer is to start with a minimal installation if you are going to create a server. Install the smallest number of software packages that you can. On a Red Hat Enterprise Linux or CentOS system, select Minimal Install, as shown in Figure 6-2. There are a number of other options, of course, but if you select any of these, the number of packages you will start with will increase.

**FIGURE 6-2**

Select Minimal Install.

With a minimal installation, you get no graphical user interface. When you log in, everything is text-based. Everything happens on the command line. You can remotely access the system using Secure Shell (SSH) to get a command line and do everything remotely that you can on the console. Of course, even with a minimal installation, nearly 300 packages will be installed. On a RHEL/CentOS 7 system, that number is 296, as shown here:

```
[root@thornhump ~]# rpm -qa | wc -l
296
```

Installing **GNOME**, the standard desktop environment in RHEL/CentOS and many other Linux distributions, will add another 57 packages. Add the GNOME Display Manager (GDM), which is the graphical login manager and software that starts up the desktop session on login, and you get nearly 500 packages, as shown in the following code. If you were to do a desktop installation rather than the minimal installation, you would get around 1,000 packages. Not to keep beating on this particular drum, but more software packages means a lot more potential for vulnerable software and more work for you trying to keep it up to date.

```
[root@thornhump ~]# rpm -qa | wc -l
489
```

Installing GNOME or GDM adds so many packages because each of these software packages has a large number of dependencies. When you build the software package from source, you have the option to enable a number of features based on other software or libraries you may have installed on your system. When the maintainers of each distribution create the software packages that will be installed on your system, they determine which other software will be installed along with the one you choose. As an example, when they build GNOME, they determine which features will be enabled. This determines the software dependencies that must be installed along with GNOME. If you want more control over the dependencies that will be installed, you can build everything by hand after you have a minimal system build and a set of **development tools**, such as a compiler and a make utility (just as a starting point). These enable you to compile and install software.

Perhaps an easier way to address this problem is to use a source-based distribution like Gentoo Linux. Gentoo uses a set of scripts to automate the build process. With Gentoo, you use the **portage system** to install packages. The portage system is based on the FreeBSD ports, which is a set of third-party packages that can be installed from source. Just like the ports in FreeBSD, portage is a set of scripts that will build the software. Similar to ports, you can specify packages you don't want to install unless it's absolutely necessary.

Some software absolutely requires other software. If that is the case, you will have to build and install the dependency first. If it's optional, however, you may choose not to build the dependency, keeping the amount of software to a minimum.

FYI

Gentoo Linux is a distribution created by Daniel Robbins. The first version was released in 2001. It's named after the gentoo penguin because the gentoo is a very fast swimmer. One of the objectives of Gentoo early on was to provide a way to really speed up a system by providing users the ability to highly customize their environment. In addition to limiting build dependencies, which should create leaner executables, users can add configuration variables to create more optimized and theoretically faster executables.

Gentoo uses an **environment variable** to manage this. An environment variable is a word used to equate to something else. In math, you might use a variable like x to equate, or be equal to, some equation. A program can use an environment variable to obtain information that is relevant to the system or to a particular user. In the case of portage, the **USE** variable is a string of packages that either should or should not be included. An example of a **USE** variable for a system on which the user doesn't wish to install a GUI might include something like the following:

```
USE="-kde -qt -gnome -gtk -gdm"
```

Setting the **USE** variable like this would tell the portage system not to build any of the referenced packages unless it's absolutely necessary. In theory, this keeps the number of packages low and your system configured the way you want it. Of course, this requires you to use only a minimal number of requirements and not to just throw in dependencies because the software package happens to support a feature that relies on another piece of software.

If an optional dependency is flagged as required, you won't get the option to not install the dependency. It will just be installed automatically. As shown with Red Hat, you'll see the list of all the packages that will be installed before they actually are installed, which will allow you to make the decision whether to install everything or not.

Service Management

Service management includes both starting up services and ensuring those services all start up correctly. There is also a component of manually starting up and stopping each of these services. While there is a lot of commonality between different Linux distributions, including some of the concepts going back to AT&T Unix, there is some diversity, too.

This section covers three different system startup processes. Although two of them share features in common, the third is completely different. In addition, many utilities surround the services, including utilities used to place the service scripts correctly so they start up automatically.

SysV Init

After services are installed, they must be managed. The first thing to consider is how services get started. When a system boots up, a master process is responsible for starting everything up. When the kernel loads up, this master process starts. It figures out which runlevel should be started. A *runlevel* is a way of categorizing functionality by determining in which mode a system will operate and, as a result, which services will be available. There are seven runlevels on Unix systems, although only levels two through five have a set of services that you would start up.

Traditionally, **init** is the master process that starts all services. The init program as means of starting up the system goes back to the days of AT&T Unix. Each service on the system has a script that is responsible for setting any variables and getting the service program running. These scripts are commonly stored in the /etc/init.d/ directory. The different runlevels each have their own directories, including the runlevel number. For example, runlevel 1 might have scripts in /etc/rc1.d, while services in runlevel 2 might have scripts in /etc/rc2.d.

Different runlevels are used for different purposes. The **Linux Standard Base (LSB)** specification calls for seven runlevels. The LSB is a standard specification for what it means to be Linux. In this case, if you have an operating system that is called Linux, you can expect that it will support these seven runlevels. [Table 6-1](#) describes each of the runlevels so you can determine where you would place the services that you are installing.

Any service configured to run in a given runlevel will have two scripts in the runlevel directory: one to start the service and the other to shut it down. Both of these scripts will be symbolic links back to the real service script in /etc/init.d/, but these two links will have different names. The startup link will have a name that begins with the letter S followed by a two-digit number. The numeric value is used to determine the starting order. The second script for the service starts with the letter K and is used to kill or stop the service script. You can see examples of these names and links in the following code.

FYI

SysV is a reference to the version of AT&T Unix that was released in the 1980s. It's a shorter version of System V, although strangely, System V is the successor to System III. There were four major releases of SysV. These were referenced with an R and the release number. As an example, SysVR4 is release 4 of System V. This may also be noted as SVR4. SysVR4 was widely adopted. Many distributions of Linux take features, functionality, and behaviors from that particular release. The process used to start the system up and launch services is one such piece of functionality.

TABLE 6-1 Runlevel definitions.

RUNLEVEL PURPOSE

- | | |
|---|---|
| 0 | Shutdown —Setting the runlevel to 0 will shut the system down. |
| 1 | Single user —This is commonly used for administrative purposes. |
| 2 | Multiuser mode, no network —Users will have to log in, but no network interfaces are configured. |
| 3 | Multiuser mode with network —The system will require you to log in, and you will have network interfaces configured. |
| 4 | Not used. |
| 5 | Multiuser with network and graphical user interface —The graphical user interface is started automatically. |

6

Reboot—If you set the system to this runlevel, the system will reboot.

```
lrwxrwxrwx 1 root root 21 Jun 9 19:21 K20prelude-lml ->
  ↳ ./init.d/ prelude-lml
lrwxrwxrwx 1 root root 15 Mar 13 17:30 K20rsync -> ./init.d/rsync
lrwxrwxrwx 1 root root 15 Mar 13 17:30 K20saned -> ./init.d/saned
lrwxrwxrwx 1 root root 27 Mar 13 17:30 K20speech-dispatcher ->
  ↳ ./init.d/speech-dispatcher
lrwxrwxrwx 1 root root 32 Mar 13 17:30 K20virtualbox-guest-utils ->
  ↳ ./init.d/virtualbox-guest-utils
lrwxrwxrwx 1 root root 29 May 16 20:00 K37iptables-persistent ->
  ↳ ./init.d/iptables-persistent
lrwxrwxrwx 1 root root 16 May 3 20:48 K39auditd -> ./init.d/auditd
lrwxrwxrwx 1 root root 13 May 17 15:48 K77ntp -> ./init.d/ntp
-rw-r--r-- 1 root root 369 Mar 12 2014 README
lrwxrwxrwx 1 root root 19 Mar 13 17:30 S30killprocs ->
  ↳ ./init.d/killprocs
lrwxrwxrwx 1 root root 19 Mar 13 17:30 S70dns-clean ->
  ↳ ./init.d/dns-clean
lrwxrwxrwx 1 root root 18 Mar 13 17:30 S70pppd-dns ->
  ↳ ./init.d/pppd-dns
lrwxrwxrwx 1 root root 16 Mar 13 17:30 S90single ->
  ↳ ./init.d/single
```

Each of these three files is in the /etc/rc1.d/ directory and links to the scripts located in /etc/init.d/. This is indicated by the .. used to go up one directory level to /etc before going down to init.d/. When the scripts are run, they are run in ASCII order, so a script named S01 would start before one named S02. If two scripts have the same number value, the running order is based on the name. Uppercase letters come before lowercase letters in the ASCII table.

```
kilroy@hodgepodge /etc/init.d $ sudo service postfix restart
 * Stopping Postfix Mail Transport Agent postfix                                [ OK ]
 * Starting Postfix Mail Transport Agent postfix                               [ OK ]
```

FIGURE 6-3

Using the `service` command to start postfix.

Dependencies are handled using the name. If one service must be started for another one to work, the dependency must have a name that will be started first. Paying attention to the naming of scripts will ensure you don't have a service failure because one of the dependencies didn't start correctly.

As an example, having a mail server up and running is very important for many services because some send notifications by e-mail. As a result, having the mail server start early on in the process is important. On an Ubuntu system, the Postfix service has a lower number than most of the other services. On a recent installation of an Ubuntu system, the script in /etc/rc3.d is numbered S20, while other services, like the Apache Web server, are numbered S91. Services that are not dependencies for others will generally have high numbers, often in the 80s or 90s.

Of course, starting services manually is as easy as running either the script in /etc/init.d/ or the link in one of the runlevel directories. To start the Postfix mail server, you issue the command `/etc/init.d/postfix start`. To stop the service, you issue the command `/etc/init.d/postfix stop`. You can also check the status, reload the configuration, and restart the service, which is like issuing a stop and then a start.

There are easier ways to perform the same functions, though. Rather than typing the path to the script and then the script name, it's easier to just use the `service` command. The service utility passes commands to the service scripts without having to call the service script directly. To restart the Postfix service, you just pass the name of the service along with the `restart` command. (See [Figure 6-3](#).)

The `service` command is for more than just starting and stopping services. You can also get a list of all of the services installed and check their status. This is helpful if you want to quickly see a list of all the services and whether they are running. In the partial list that follows, you can see the output of running `service --status-all`. In the brackets, you see the status of each service. The + indicates that the service is running, the - indicates that isn't, and the ? indicates that either a status result could not be obtained or the service script just doesn't support the `status` command.



NOTE

Not all scripts support a `status` command. As a result, there is no way to determine from the service script whether the service is running or not. Some service scripts may also fail to generate a good answer for a number of reasons. This would also result in the service utility being unable to determine the status.

```
kilroy@hodgepodge /etc/init.d $ service --status-all
[ + ] acpid
[ - ] anacron
[ + ] apache2
[ + ] auditd
[ + ] avahi-daemon
[ ? ] binfmt-support
[ + ] Bluetooth
[ - ] brltty
[ - ] casper
[ ? ] console-setup
[ ? ] cpufrequtils
[ + ] cron
[ ? ] cryptdisks
[ ? ] cryptdisks-early
[ + ] cups
[ + ] cups-browsed
[ - ] dbus
[ ? ] dns-clean
[ + ] friendly-recovery
[ ? ] fwlogwatch
[ - ] grub-common
[ - ] hddtemp
[ ? ] iptables-persistent
[ ? ] irqbalance
```

```
[ + ] kerneloops
[ ? ] killprocs
[ ? ] kmod
[ - ] krb5-admin-server
[ - ] krb5-kdc
[ - ] lm-sensors
[ ? ] loadcpufreq
[ + ] mdm
[ ? ] mintsystem
[ ? ] mysql
[ ? ] networking
[ - ] nmbd
[ + ] ntp
[ ? ] ondemand
[ - ] postfix
[ ? ] pppd-dns
[ + ] prelude-correlator
[ ? ] prelude-lml
[ + ] prelude-manager
[ ? ] prl-x11
[ ? ] prltoolsd
[ - ] procps
[ - ] pulseaudio
```

When you install packages that have included services, like the prelude-lml manager, the service portion isn't necessarily installed by default. The script is in place, but the links have not been created to start the service in the right runlevels. Fortunately, there are utilities that will create all the right links to make sure the service starts. On an Ubuntu system, you would use the update-rc.d utility. As shown in the following code, you can select which runlevels you want to start and stop in, or you can just specify that you want to install in the default locations.

```
kilroy@hodgepodge /etc/init.d $ sudo update-rc.d prelude-manager
→ defaults
Adding system startup for /etc/init.d/prelude-manager ...
/etc/rc0.d/K20prelude-manager → ../init.d/prelude-manager
/etc/rc1.d/K20prelude-manager → ../init.d/prelude-manager
/etc/rc6.d/K20prelude-manager → ../init.d/prelude-manager
/etc/rc2.d/S20prelude-manager → ../init.d/prelude-manager
/etc/rc3.d/S20prelude-manager → ../init.d/prelude-manager
/etc/rc4.d/S20prelude-manager → ../init.d/prelude-manager
/etc/rc5.d/S20prelude-manager → ../init.d/prelude-manager
```

This creates start scripts in runlevels 2, 3, 4, and 5 and kill scripts in runlevels 6, 1, and 0. As an example, the first line indicating the symlinks that are created shows that a kill script is placed in runlevel 0. (A symlink is a symbolic link, sometimes called a *soft link*. It's a reference, or a shortcut, to a filename in the filesystem.) Because runlevel 0 is shut down, you want there to be only a kill function.

In runlevels 2, 3, 4, and 5, a startup link is created. The reference to /etc/rc3.d/ S20prelude-manager is an example of that. If you later decide you want to prevent the service from starting up, you just specify which

service you want to remove and tell update-rc.d to remove the service, as shown in the following code. It will then go through and delete all the links. It doesn't stop the service; it just removes the links so the service won't start up on the next reboot.

```
kilroy@hodgepodge /etc/init.d $ sudo update-rc.d -f prelude-manager
→ remove
Removing any system startup links for /etc/init.d/prelude-manager ...
/etc/rc0.d/K20prelude-manager
/etc/rc1.d/K20prelude-manager
/etc/rc2.d/S20prelude-manager
/etc/rc3.d/S20prelude-manager
/etc/rc4.d/S20prelude-manager
/etc/rc5.d/S20prelude-manager
/etc/rc6.d/K20prelude-manager
```

Each of the service scripts being manipulated is just a series of text statements commonly written in the **Bourne Again Shell (Bash)** scripting language. Bash is a command interpreter used to issue text commands. Each shell used on Unix or Linux systems includes a basic programming language to automate tasks. This is the programming language used to write these scripts. Each of the service control scripts includes functions that implement the `start`, `stop`, `status`, and other aforementioned commands that you can pass into the script using the `service` command. The scripts set environment variables, which perform functions such as storing the process identifier number in a file so it can be referenced later.

Commonly, these process IDs are stored in files in `/var/run`, where other runtime details about the process are kept. This can be useful to either restart the service or stop it. You do this by sending a signal to a process ID to get the process to shut down gracefully. While these files can be used to manipulate the service process, it's easy enough to obtain the process ID by getting a process list using the `top` or, better yet, the `ps` command. The `ps` command can be used with other commands to extract the process ID from the running system. This is more reliable than a process ID in a file because the file could have been in place across multiple service starts if they were done manually.



NOTE

The Bourne Again Shell is named after the Bourne shell, which was an early shell on Unix. Bash is considered an improvement over the original Bourne shell, with better history functionality and job control. In Bash, features from other Unix shells are integrated into the original Bourne shell.

Upstart

One of the challenges with init is that it's serial in nature. Every step of init is executed one at a time. Another challenge is that the initialization scripts don't directly support dependencies. To make sure you have one service before another, you must name the symlinks in the runlevel directories correctly.

Upstart was developed by Canonical, the company responsible for Ubuntu. Ubuntu and all Linux distributions derived from it use Upstart. Upstart is an attempt to make system and service initialization more manageable. It also uses a different paradigm for managing services. Services use a bus-style architecture where services can register to receive events that are emitted. To communicate with a service, you use a utility to emit an event to the service. This is similar to calling a script with a parameter. However, the mechanism used to get the service to respond the way you want—either by starting up, providing status, shutting down, or any other feature support by the service script—is different.

Suppose you wanted to restart the ssh server service on an Upstart system. You would use the utility initctl. As you can see here, you specify the command you want to send to the service and then the name of the service, as registered with Upstart.

```
kilroy@dallas ~ $ sudo initctl restart ssh
ssh start/running, process 6862
```

While the service-management scripts generally look a lot like those from the init style of system startup, they are stored as configuration files in the /etc/init directory and have some different capabilities. For a start, you can specify a set of dependencies. This may be a set of service scripts or it may be other system events. As an example, the following fragment is taken from the Common Unix Printing System (CUPS) service script. It relies on the filesystem being up and running, which is an indication that the operating system is likewise up and running. You can see this where it says **start on (filesystem...)**, meaning that the conditions in the parentheses must be fulfilled before the service will start. Beyond the filesystem, the CUPS service requires the avahi-daemon service to be running and then either the dbus service or the system being up in runlevels 2–5. The avahi-daemon is an implementation of Apple's Bonjour service, which provides a way of advertising various services over a local network. Because there may be external printers on the network that are managed by CUPS, this service must be running before CUPS itself is started. There is also a check to determine which runlevel you are in. Because there is no runlevel 1, the CUPS service won't start in single-user mode unless the dbus system is running. The **or** ensures that.

```
start on (filesystem
           and started avahi-daemon
           and (started dbus or runlevel [2345]))
stop on runlevel [016]
```

When the system gets to runlevel 0, 1, or 6, the CUPS service shuts down. This is based on the **stop on** statement. Some systems may rely on network interfaces being up and running. There are provisions for that within Upstart. You can check for whether a particular network interface is up before a service will start. This allows you to make sure you have very reliable services that behave in a way that you expect.

Using Upstart, init is still the master process. If you run a **pstree**, you can still see that the very top process is init. Upstart systems also retain the /etc/init.d/ directory as well as the rc directories. As a result, you can still control it using the older init style. However, it's designed to be managed by the newer, more capable functionality. Using initctl instead of the older methods will ensure that all dependencies are in place before the service runs.

Systemd

Red Hat switched to the use of **systemd** starting with Fedora Core. In Red Hat Enterprise Linux (RHEL) 7, it took over the system startup. Beginning with 15.04 (that is, the release of Ubuntu in April, 2015), Ubuntu has migrated to the use of systemd in lieu of the Upstart method previously used. In place of the init process used in both Upstart and SysVInit systems, the very first process that starts on the system is systemd.

FYI

Dbus is one of the underlying technologies that can be used with systemd. Dbus is a way of passing messages between processes. Services that want to use this essentially connect to the bus. Any message sent over the bus is sent to a specific object rather than the process as a whole. In this way, it functions as a **remote procedure call (RPC)** interface. RPC is a way of getting a function or method to run on a separate system. Typically, network file-sharing solutions use RPC to function.

Systemd, though, is a much broader effort than simply being the master process that controls all of the service stops and starts. Systemd is an attempt to provide a foundation for management of the entire system, not just simply a master process or a way to start up and manage services. The primary utility used to control a system that uses systemd is **systemctl**. With systemctl, you can send messages to processes or services as well as shut down or reboot the entire system.

Systemd offers several ways to provide better security to services than does the older init method of service management. Instead of service scripts (in the case of init) or even simple service configuration files (as in Upstart), systemd uses more feature-rich configuration files. Unlike Upstart, systemd doesn't have any scripting in the configuration files—just settings related to the service. Those settings, though, provide areas where you can harden the service. Here are some examples from a time synchronization service on an Ubuntu 15.04 system located in /etc/systemd/system/sysinit.target.wants/ systemd-timesyncd.service. These are all set in the [Exec] block that defines settings for the executable itself. Anything that can be used to provide additional information to the service executable or restrict what that executable can do would be included in the [Exec] block.

```
PrivateTmp=yes
PrivateDevices=yes
ProtectSystem=full
ProtectHome=yes
```

With the **PrivateTmp** setting, the service is provided a temporary folder to which only that service has access. In many cases, the shared temporary folder at /tmp has been a problem because of its shared nature. Files or folders in /tmp can be manipulated to cause problems for a running service. The temporary folder can also be a place to drop staging files for an attack if the service has only limited write permissions to the filesystem.

Most services have access to the shared system-level temporary folder. This makes it a place where services can do bad things. However, by restricting the service to a private folder that only it has access to for temporary information, and by jailing it there so it cannot get out, the service can only affect itself. This limits the potential danger to other services and maybe to the system as a whole.

As a rule, everyone has access to the entire range of devices on the system via /dev. The devices may have individual permissions so nothing can be done with them. At a minimum, though, they will be visible. If **PrivateDevices** is set to **yes** on a systemd system, the service only gets access to a limited set of devices in /dev. The service will no longer have access to the full range of devices. Most services don't have a reason to access devices and certainly not the complete range of services on the system. As a result, setting **PrivateDevices** to **yes** is a good way to help protect the system.

Many services also don't need to write to most places in the filesystem. As a result, it's better if services have only read-only access to most of the filesystem. **ProtectSystem** is a configuration parameter that forces specific directories to be mounted read-only for the purposes of that service. If **ProtectHome** is set to **yes**, the /usr and /boot directories are mounted as read-only for that specific service. If it's set to **full** as in the preceding example, the /etc directory is also mounted read-only. If **ProtectHome** is set as it is here, the /home and the /run/user directories appear to be empty, making them inaccessible for the purposes of the service in question.

A number of other settings are available to restrict access to the system by the services under the control of systemd. One of them restricts network access. **PrivateNetwork** can be set to **yes** to make it appear to the service that only a loopback interface is configured on the system. If you have a system that doesn't really need network access, but you have concerns about it being compromised and used as a path to connect to the Internet, you can turn this setting on to prevent Internet access or any other network access.

There are other ways of controlling services using systemd. On a Linux system, services can get a list of capabilities, which are specific permissions granted to privileged processes. Using systemd, an administrator or service owner could limit the capabilities available to a service running at a privileged level. **Capabilities** and **CapabilityBoundingSet** are two settings for which you can provide a comma-delimited list of

capabilities that you want to give the service. This will be another way of limiting what the service can gain access to.

Hardening Services

A common trope about system security is that the most secure system is one that isn't powered on. Similarly, the most secure service is one that isn't running. This is unrealistic, however. Where a service under systemd can really be restricted, there are other areas to be concerned about.

One of the first and quickest ways to harden a service—that is, to restrict what it has access to in order to protect it and the system on which it runs—is to force it to run with a limited set of permissions. You do this by forcing the service to run as a specific user who is granted permissions only on the system necessary to allow the service to function. If the service needed to read a configuration file, for example, that directory and configuration file would require read permissions for the user set on that directory and file.

The permissions and users are just the starting point, though. Many services have listeners. A listener might be a **socket**, which is a file through which a service communicates, or it may be a network port on which the service is listening. [Figure 6-4](#) shows a number of listening ports on an Ubuntu system. This list includes service names, so you can see the services that are listening. The Foreign Address column is the important one. You can see that in the case of both Postgresql and MySQL, as examples, the listeners are bound only to the **loopback interface**. Anything sent to a loopback interface comes back to the system. The loopback interface isn't accessible from anywhere outside the system. Any network communications sent out the loopback interface are returned on that interface. This is shown in the Local Address column. Each of these two services shows that the local address is localhost, which is the name given to the address 127.0.0.1, commonly bound to the loopback interface. The 127.0.0.0/8 address block is assigned to all loopback interfaces so any address in that block could be used. Typically, however, it's 127.0.0.1.

When a network service starts up, it registers itself with the operating system through a process called *binding*. This process tells the operating system which port to attach to the application. When the operating system receives a communication on that port, it knows to send the data in the communication to the listening service.

Any service that is listening on all ports, sometimes indicated by an asterisk (*) in a listing of listening ports, is exposed to any system that can reach it through any network path. This may be either through the local network or by way of the Internet. Limiting the network port to just the localhost will restrict any network communication from remote hosts. If the service is needed only to listen to connections from other services locally, as in the case of the databases mentioned earlier, then restricting the interfaces on which the service listens will be important. Perhaps it should go without saying, but it's essential that databases protect their information by any means necessary. Other services can be limited in this way as well.

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State
tcp	0	0	localhost:mysql	*:*	LISTEN
tcp	0	0	milobloom:domain	*:*	LISTEN
tcp	0	0	localhost:ipp	*:*	LISTEN
tcp	0	0	localhost:postgresql	*:*	LISTEN
tcp6	0	0	ip6-localhost:ipp	[::]:*	LISTEN
udp	0	0	*:mdns	*:*	
udp	0	0	*:49394	*:*	
udp	0	0	milobloom:domain	*:*	
udp	0	0	*:38976	*:*	
udp	0	0	*:bootpc	*:*	
udp	0	0	*:ipp	*:*	

FIGURE 6-4

Listening ports.

If you are unable to limit the interfaces on which the service listens, as in the case of a Web server like Apache for instance, there are other ways of protecting network services. One is to introduce access control lists (ACLs) into the application. Consider a mail server. With a mail server, the application may need to know which systems to allow communication with. For a Simple Mail Transport Protocol (SMTP) server tasked with sending messages to recipients as well as receiving messages for them, only trusted addresses should be allowed to send through the server. This prevents unauthorized use, which can lead to relaying or spam, sometimes called unsolicited commercial e-mail (UCE). The following line tells Postfix, an SMTP server, which addresses are allowed to send through this particular server.

```
mynetworks = 127.0.0.0/8 172.30.42.0/24 [::ffff:127.0.0.0]/104
      [::1]/128
```

The easy ones are the localhost addresses for both IPv4 and IPv6. The IPv4 address is 127.0.0.0/8, while [::ffff:127.0.0.0]/104 and [::1]/128 are IPv6 addresses referring to the localhost. In addition, the address block for the local area network (LAN) is included in the list associated with the variable `mynetworks`. This tells Postfix that anyone on the local network could send messages through this particular server. Of course, the server administrator can add as many addresses and address ranges as he or she wants to the list of trusted addresses. This is an ACL because it's a list of network addresses used to control access to the functions offered by the service.

Another means used to control access to a network service is authentication. Postfix and other SMTP servers can support this, as can Web servers. The Apache Web server supports user authentication before allowing access to the pages served up by the Web server. The following example is a set of directives in the file `.htaccess`. The `.htaccess` file is placed in a directory to which the administrator wants to control access. This is a very simple example.

```
AuthType Basic
AuthName "Authenticate"
AuthUserFile /var/www/password
AuthGroupFile /var/www/group
Require Group admins
```

In this example, the file with the users in it is in `/var/www/password`. Entries into this file are created with the utility `htpasswd`, which creates hashed passwords associated with users. Basic authentication passes the username and password in a weakly encoded fashion. To create the authentication string, the username and password are concatenated, or linked, using a colon as a delimiter. The username wubble with the password

Password123 would be put together as wubble:Password123 and the result would be encoded using **Base64**. Base64 is a way of converting non-printable characters into a string that can be transmitted using a text-based protocol like HTTP. (Base64 is used for other purposes as well.) While it's easily decoded, transmitting a username and password that has been Base64 encoded prevents it from being read in plaintext. It's not really much better than plaintext, but it will stop anyone who doesn't recognize Base64. The aforementioned username and password would be transmitted as d3ViYmxlOlBhc3N3b3JkMTIzCg== as Base64. The file /var/www/password would store a hashed version of the password, as shown in the following code.

FYI

A *hash* is a fixed-length value generated by a cryptographic function. No matter what the input is, the same length will be generated by the hash algorithm. A hash algorithm is a one-way function, meaning that you cannot feed a hash value into any program or algorithm and get the original value. Common hashing algorithms are Message Digest 5 (MD5), Secure Hash Algorithm 1 (SHA-1), and Secure Hash Algorithm 2 (SHA-2). Because there are potential issues with both MD5 and SHA-1, SHA-2 is the recommended algorithm for generating hash values. The advantages of storing password in a hash value are that the password isn't stored in plaintext, and you cannot just reverse the hash value to obtain the password.

```
wubble:$apr1$jdLo6tWZ$90KTuBjp4yxwBtf6RE5eq.
```

Digest authentication is considered better than basic authentication. With digest authentication, the server transmits a random value to the client when it indicates that authentication is required. This random value is called a **nonce**. The nonce is then combined with the username and password, and the result of that is hashed using a cryptographic hash like MD5 or SHA-2. The resulting hash is then transmitted to the server. Because the service already knows the nonce, it compares the hashed value it receives to a hash of usernames and passwords it already knows. When one matches, the server can authorize the user.

Using Mandatory Access Controls

Another means of protecting services on a Linux system is by implementing mandatory access controls. Mandatory access controls are a set of policies and permissions established at the system level. They're called *mandatory* because no user cannot change the permissions. If access control can be modified by users, as is commonly the case with the standard Linux permissions and on Windows systems, it's called *discretionary access control*. Two packages can provide mandatory access control on a Linux system: Security Enhanced Linux (SELinux) and App Armor.

Security Enhanced Linux

SELinux was originally developed by the National Security Agency (NSA), but Red Hat has completed a lot of development work on it. SELinux provides a couple of different defenses that are helpful when it comes to protecting services.

The first defense is to provide more of a lockdown on filesystem files. SELinux implements labels within the filesystem to provide more granularity with permissions on files. As an example, the following file listing shows the user, role, and type for each of the files and directories. To SELinux, all files are considered objects, so these two directories have a role object categorization. The type is based on the content of the directories. The cgi-bin directory has executable scripts in it, so that is the type assigned in the label. The html directory includes content associated with the httpd (Apache) service, so that's the type associated with the directory. The files in the directory would typically have the same label as the directory, though that isn't always the case.

```
[root@thornhump www]# ls -lz
```

```
drwxr-xr-x. root root system_u:object_r:httpd_sys_script_exec_t:s0
└── cgi-bin
drwxr-xr-x. root root system_u:object_r:httpd_sys_content_t:s0 html
```

You can set SELinux to either permissive or enforcing mode. When it's set to permissive mode, different actions are logged, but nothing is done to prevent the action or access from occurring. When SELinux is set to enforcing mode, the kernel modules created for SELinux can prevent unauthorized access. All the permissions are set through a policy that is configured by the administrator. Users cannot make changes to these permissions. Again, this is what makes the access control mandatory. The policy for SELinux is stored as a binary file, compiled from a set of configuration files in /etc/selinux. SELinux can protect more than just files. Ports also have a security context associated with them. If a process doesn't have the right context, it won't be able to bind to a port that has been configured with the same context the process has. A listing of some of the contexts and ports associated with them appears in the following code.

SELinux Port Type	Proto	Port Number
afs3_callback_port_t	tcp	7001
afs3_callback_port_t	udp	7001
afs_bos_port_t	udp	7007
afs_fs_port_t	tcp	2040
afs_fs_port_t	udp	7000, 7005
afs_ka_port_t	udp	7004
afs_pt_port_t	udp	7002
afs_vl_port_t	udp	7003
agentx_port_t	tcp	705
agentx_port_t	udp	705

These ports are known and associated with particular services. The first several types, associated with different ports, are related to the Andrew File System, a network filesystem developed at MIT. Every known service has a type associated with it. The ports will be associated with that type and the service should have an executable that is labeled with a context that includes the correct type. If the type doesn't match between the port and the executable, the executable won't be able to bind to that particular port.

AppArmor

SELinux isn't the only mandatory access control facility available for Linux. Whereas SELinux was originally developed by the NSA, AppArmor was developed specifically for a Linux distribution, Immunix. The company that developed it was eventually acquired by Novell and AppArmor was used in SUSE Linux, also owned by Novell. Today, it's used by Ubuntu and its derivatives.

To check the status of a system using AppArmor, you run `apparmor_status`. It provides a list of profiles that are in enforce mode. The following sample is from an Ubuntu system. As you can see, there are 24 profiles in enforce mode. Below that are the application profiles configured in enforce mode. Of course, the following list is just a portion of the 24 profiles configured on the system in question.

```
kilroy@milobloom:/etc/apparmor.d$ sudo apparmor_status
apparmor module is loaded.
24 profiles are loaded.
24 profiles are in enforce mode.
```

```
/sbin/dhclient
/usr/bin/evince
/usr/bin/evince-previewer
/usr/bin/evince-previewer//sanitized_helper
/usr/bin/evince-thumbnailer
```

In addition to enforce mode, AppArmor also supports complain mode. Unlike SELinux, where the entire system has to be either enforce or permissive, AppArmor can be configured on a per-application basis to either complain (log the behavior) or enforce (prevent unauthorized behavior). When an application is installed on a system, it must have an associated AppArmor profile. A sample of one of these profiles follows. This particular profile comes from the Network Time Protocol (ntp) daemon.

```
network inet dgram,
network inet6 dgram,
network inet stream,
network inet6 stream,

@{PROC}/net/if_inet6 r,
@{PROC}/*/net/if_inet6 r,
@{NTPD_DEVICE} rw,

/{,s}bin/ r,
/usr/{,s}bin/ r,
/usr/sbin/ntpd rmix,

/etc/ntp.conf r,
/etc/ntp.conf.dhcp r,
/etc/ntpd.conf r,
/etc/ntpd.conf.tmp r,
/var/lib/ntp/ntp.conf.dhcp r,
```

While this is only a section of the file, you can see that there are permissions configured for the application. The daemon is given permission to communicate with the network, which would be required for a daemon that is listening on a particular port. Beyond that is a series of files that have permissions associated with them. Most of these restrict the /usr/sbin/ntpd executable to read-only access to various configuration files. The significant difference is the binary file itself, which allows read and execute permissions (specifically, it's allowed to inherit execute permissions) and is also allowed to call memory map functions.

AppArmor provides a comprehensive way to restrict all permissions associated with any executable on the system if a profile is created for it. Restricting access to network functions in the AppArmor profile is one way to help protect a service. Restricting which portions of the filesystem can be accessed and how will also help protect the system from a service that has been compromised.

Servers Versus Desktops

There are a handful of differences between a desktop and a server. Some of them are in the kernel, including how it performs with respect to multitasking. Others are in the number of packages that are installed on a system.

A desktop system will commonly have quite a few more packages installed than a server will. A desktop will also have a different set of services installed. In the case of a server, there is typically a very limited set of services exposed to the outside world. The only services should be the primary service the server is there to provide as well.

as services that are necessary to remotely manage the system or the services. This might include SSH so an administrator can get in remotely.

Any administrative services should also exist on a separate interface and not be exposed to the outside world. The server system may have multiple network interfaces to better manage who gets access to which service and from where without having a lot of complex firewall rules to restrict access. A desktop system would commonly have only one interface.

Beyond that, there are services that are common to desktop systems. These include Samba, which is a file-sharing service used to interact with Windows systems. When it comes to file sharing, a desktop system may also use the Network File System (NFS) service to connect to file servers or even other desktop systems. It's best to restrict access using ACLs within Samba or within the NFS configuration settings to make sure only local systems can gain access to file shares.

Desktop systems may also have a number of other services associated with remote access, including remote desktop services like Virtual Network Computing (VNC) or even X. X is a windowing system developed for Unix in the 1980s, and it still provides the foundation for graphical user interfaces today. X was another service developed at MIT.

Protecting Against Development Tools

Desktop systems may have a lot of packages installed. Some of these may be development tools, which allow the user to custom build or create software.

Having development tools installed on server operating systems used to be a very significant concern. There was a time when there were a lot of different operating systems and CPU architectures in the systems that were connected to the Internet. Hewlett-Packard, Sun Microsystems, Silicon Graphics, and IBM all had variations of Unix, and these Unix variants ran on different processors. Even in the space where there was an Intel processor, there were a number of operating systems like Windows, FreeBSD, OpenBSD, and, of course, Linux. Having development tools installed on the system was considered a risk because if an attacker was able to break into a system, that person could use the tools build any attack tool he or she wanted and then deploy it.

Suppose you had a Web server, and part of the Web application was vulnerable to **command injection** attack. In other words, an attacker could send a shell command into the Web application and have the operating system execute it. Using this entry point, an attacker could have the system download the source code for an attack tool, then build the tool and use it—all from the vulnerable Web page. This might allow an attacker to open a shell remotely that he or she could use to further take advantage of the system.

This is just one scenario. If the build tools weren't installed on the target system, the attack tool could not be built. If the attacker wasn't sure which operating system you have or did not have access to the operating system and hardware architecture, he or she wouldn't know exactly which executable to put on the system. In a case where the primary CPU in use is an Intel compatible and the system is Linux, you can build any tool you want and just grab your prebuilt binary to run on the target system.

In a more homogenous environment where Linux is one of the predominant operating systems and the vast majority of Linux systems run on Intel systems, an attacker could build any Linux tool ahead of time and just put it on the target system when needed. Additionally, most Linux systems have package-management systems in which tools can be easily installed from a package repository. For example, if you want netcat, you just run the package-management tool and install it. If you have enough access to execute commands that can download files, you have access to run the uname utility to get the name of the operating system. You can see an example of that in the following. The output from the kernel version in uname includes the Linux distribution. This is an Ubuntu system.

```
Linux milobloom 3.19.0-21-generic #21-Ubuntu SMP Sun Jun 14 18:31:11
➥ UTC 2015 x86_64 x86_64 x86_64 GNU/Linux
```

If the system has a custom-built kernel, it may not include the name of the distribution. The Linux Standard Base (LSB) also specifies that a Linux distribution adhering to the Linux standard have a file that includes all the information about the Linux distribution. All you have to do to find the Linux distribution you are looking for is to look at the contents of the /etc/lsb-release file. It will include the distribution identification as well as release information. You can see the details about this Linux system in the following code.

```
kilroy@milobloom:/etc/postfix$ cat /etc/lsb-release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=15.04
DISTRIB_CODENAME=vivid
DISTRIB_DESCRIPTION="Ubuntu 15.04"
```

Once you have the distribution, you know which package-management system you are dealing with, and you can install anything you want—including development tools. This isn't to suggest that having a build system installed, including a compiler, make system, and libraries, is a good idea. The more packages you have installed, the more risk you assume, because any one of those packages could be compromised in some way. However, the risk associated with simply having development tools like compilers installed on a system isn't what it once was. At best, it's a simple hurdle for someone to overcome fairly easily either by compiling his or her own tools or by installing the tools using package-management systems. These package-management systems just did not exist earlier on other Unix systems like Solaris, HP-UX, Irix, or A/IX, making it much harder to get tools installed.

Of course, it's even easier for attackers to execute attacks from systems. Most systems have some form of scripting language installed because there are management scripts or services that rely on languages like Python, Ruby, or Perl. Even PHP, primarily used for Web programming, can have a command line interface (CLI) component, and you can write scripts using that language. Because some of these languages are incredibly powerful, they can easily be leveraged to create a variety of attack tools. Limiting these languages to systems that must have them is common in hardening a system, but determining exactly which libraries to install is a difficult task. This is where you go back to starting with a minimal installation and adding packages only as necessary.



CHAPTER SUMMARY

When it comes to hardening a system, it's best to start from the absolute minimum. This is true for both servers and desktops. Fortunately, when it comes to servers, most Linux distributions have a minimal installation target that you can use to start with the smallest number of packages possible. While a desktop installation is quite large, most distributions have a desktop target that includes just the packages necessary for a common desktop system. This may include a number of applications like a word processor and spreadsheet. Although you may not necessarily need these, they are so commonly used on desktop systems that they are considered a minimal starting point. Once you have a desktop system, you can still remove packages you don't want. Package managers like apt-get on Debian and Ubuntu will determine whether there are unnecessary packages on a system, like old dependencies that are no longer referenced, and suggest that you remove them.

After you have services installed, you can harden them by implementing authentication on them wherever possible. Network services should also be bound only to interfaces where the service is required. In some cases—for example, with a database—the service should be bound only to a loopback interface so there is no way to get to the service remotely. In cases where the service has to listen on an external interface, implementing access controls on the service will further protect it.

Linux has two separate packages to implement mandatory access controls. These controls are used to establish a policy at the system level, away from the control of individual users. Both SELinux and AppArmor implement these controls, but in different ways. SELinux allows for the labeling of files on the file system in

addition to policies, which provides for tight granularity on access controls. AppArmor has a set of policy configuration files that are implemented for each executable on the system. These specify which permissions the application is allowed, including network access and file permissions. This enables the system to retain the standard Unix permissions while still restricting access to files based on the policy settings.

Development tools are not typically needed on a server operating system. It's widely considered a good practice to remove from servers all development tools like compilers, development libraries, and make utilities, even if the risk isn't the same as it once was. It's also wise to limit the scripting languages that are installed on a system.



KEY CONCEPTS AND TERMS

Base64

Bourne Again Shell (Bash)

Command injection

Development tools

Environment variable

GNOME

Hardening

Init

Linux Standard Base (LSB)

Loopback interface

Multitasking

Nonce

Portage system

Red Hat Package Manager (RPM)

Remote procedure call (RPC)

Socket

Systemctl

Systemd

Tomcat

Upstart

X

Yellowdog Updater, Modified (yum)



CHAPTER 6 ASSESSMENT

1. AppArmor and SELinux are used to implement which of the following?

- A. Lattice-based access controls
- B. Mandatory access controls
- C. Password protection
- D. File sharing

2. A set of services that get started is determined by which of the following?

- A. Runlevel
 - B. Init
 - C. Process control
 - D. Init.d
- 3.** Service scripts in a system controlled by the init master process are kept in which directory?
- A. /proc
 - B. /etc/proc
 - C. /init
 - D. /etc/init.d
- 4.** A database server should typically be restricted to listening on which port?
- A. eth0
 - B. All ports
 - C. Loopback
 - D. Looparound
- 5.** SELinux was initially developed as a set of extensions by which of the following?
- A. The NSA
 - B. The CIA
 - C. The Electronic Freedom Foundation
 - D. The GNU Foundation
- 6.** Systemctl is used to perform what system function?
- A. Install packages
 - B. Manage services
 - C. Manage passwords
 - D. Configure the Web server
- 7.** What might services implement to provide better protection from remote attacks?
- A. Access control lists
 - B. Systemctl
 - C. Initd
 - D. Firewall-lib
- 8.** Which utility would you use to add or remove packages on a Debian or Ubuntu system?
- A. Yum
 - B. Apt-cache
 - C. Apt-get
 - D. Bin
- 9.** Which of the following is the lowest level of authentication used by the Apache Web server?
- A. Digest
 - B. NTLM
 - C. Base19
 - D. Basic
- 10.** Additional packages that are necessary for a requested package to function are called which of the following?
- A. Inheritance
 - B. Terminal
 - C. Dependencies
 - D. Package management

CHAPTER

7 Networks, Firewalls, and More

WHEN YOU HAVE A NETWORK SERVICE INSTALLED, you allow other systems to connect to you. If you use common services like Web servers, SSH servers, and file-sharing servers, you are guaranteed to have specific ports open. When you connect to a Web server, for example, there is an expectation that it will be listening on port 80. You can certainly have your Web server listening on a different port, but that makes it harder for users to connect to you. The **obscurity** you introduce by moving to a different port is likely to have a minimal impact on whether an attacker finds your service or not. Obscurity is the hiding of information. Sometimes this means placing it in nonstandard locations.

Some services offer access control lists (ACLs) that can restrict which IP addresses can connect to the service. Other services may use the TCP Wrapper program that handles the same sort of access controls. These features operate within an application. To catch network packets as early as possible, you want to use iptables, which is a kernel-based firewall. The iptables rules live in the kernel space, but managing them requires programs that users can run. One of these programs is also called iptables.

You can do more to enhance system security with mandatory access controls. Two major Linux options in this area are **Security Enhanced Linux (SELinux)** and **Application Armor (AppArmor)**. Both of these are implementations of mandatory access controls and use kernel drivers. You can't run both on the same system at the same time because they would interfere with one another.

Chapter 7 Topics

This chapter covers the following topics and concepts:

- How to identify different services and their TCP/IP ports
- How to configure obscurity on open ports
- How to protect some services with TCP Wrapper
- What some appropriate packet-filtering firewall commands are
- What alternate attack vectors are
- Which issues relate to wireless networks
- How to configure SELinux for a system
- How to set up AppArmor profiles
- What best practices are for networks, firewalls, and TCP/IP communications

Chapter 7 Goals

When you complete this chapter, you will be able to:

- Manage services on different TCP/IP ports
- Protect different services with TCP Wrapper and firewalls
- Understand the risks associated with alternative network connections
- Use SELinux and AppArmor to enhance security

Services on Every TCP/IP Port

Because there are more than 65,000 port numbers associated with Transmission Control Protocol/Internet Protocol (TCP/IP) services, accounting for different network services can get confusing. To reduce confusion, the **Internet Assigned Numbers Authority (IANA)** registers and manages well-known port numbers and protocols. The list of well-known port numbers includes hundreds of services that have been registered to specific port numbers.

The association between port numbers and services is documented in the /etc/services file. Other Linux configuration options refer to this file. For example, when the `iptables` command is used to configure a firewall on a Red Hat system, the associated rules are stored in the /etc/sysconfig/iptables file. When the iptables service is made active, however, and translated to a list of rules, configured iptables rules use the information in /etc/services to substitute protocol acronyms for port numbers. For example, since that file specifies ssh for port 22, the iptables rules would show ssh in place of port 22.

TABLE 7-1 A sample of well-known port-numbers.

PORt NUMBER	DESCRIPTION
21	File Transfer Protocol (FTP) connections (port 20 is the standard for FTP data transfer)
22	Secure Shell (SSH)
23	Telnet
25	Simple Mail Transfer Protocol (SMTP)
80	Hypertext Transfer Protocol (HTTP)
110	Post Office Protocol version 3 (POP3)
143	Internet Message Access Protocol (versions 2 and 4)
443	Secure HTTP (HTTPS)

Protocols and Numbers in /etc/services

Take a few moments to browse through the /etc/services file. If you have some experience with networks, you should recognize some of these port numbers. In fact, if you've passed the exams associated with certain Linux certifications, you've had to memorize more than a few well-known port numbers. If you know standard port numbers, you can create appropriate firewalls that accommodate active services more quickly.

IANA classifies port numbers in three categories:

- **Well known**—The well-known TCP/IP port numbers range from 0 to 1023. As specified by IANA, these port numbers are limited for use by system (or root) processes or by programs executed by privileged users.
- **Registered**—Registered ports have been assigned to various services. While it is best to limit the use of these ports to privileged users, it's not always required, such as for various chat services.
- **Dynamic or private**—IANA does not register services in this range of ports, leaving them free for clients who choose their ports dynamically or for privately configured protocols.

If you're using a protocol analyzer to detect the port numbers associated with network packets, be careful. IANA ports are associated with servers. The port numbers associated with clients frequently vary. Client port numbers for TCP services typically range from 1024 to 4096. Some well-known ports are listed in [Table 7-1](#).

Protection by the Protocol and Number

If you're creating a firewall, pay attention to the standard ports and protocols listed in /etc/services. Some services work with both TCP and User Datagram Protocol (UDP) packets. To allow access to such services, you need to know how to create firewall rules that support access to both types of packets on the target port number(s).

Of course, if you use nonstandard port numbers, you must document everything carefully. Administrators who work with your networks need that information to know how to make authorized connections.

Obscurity and the Open Port Problem

It isn't as easy as it looks to configure nonstandard ports. With the available Linux tools, anyone can easily identify the use of a nonstandard port. Obscurity goes beyond ports into the information that is made publicly available about your networks. Obscurity is widely considered to be a weak form of security. However, when used in conjunction with other layers of security, obscurity can make it slightly harder for adversaries to find targets.

Obscure Ports

The great majority of Web sites use the standard TCP/IP port 80 to serve unencrypted Web pages. A few webmasters configure nonstandard port numbers. Alternative ports that are frequently used for Web sites include 81, 82, 8080, and 8090. For most services, to configure a nonstandard port, you need to explicitly cite the desired port in the configuration file. User clients also need to know how to access those obscure ports. The connection from a client could be simple; for example, the elinks www.example.org:81 command uses the noted console Web browser to connect to the noted uniform resource locator (URL) on port 81.



NOTE

If you set up an obscure port for a Web site, provide instructions on how to access that Web site, as described in this section.

The connection to a client may require a switch. For example, the following command connects to an SSH server on port 2424:

```
# ssh -p 2424 michael@host.example.org
```

Opening Obscure Open Ports

The work required to set up obscure open ports may be for naught without corresponding changes to the firewall. An appropriately configured firewall accepts packets that are directed at specific ports (and more). Packets with other destinations and characteristics are rejected.

In other words, when configuring obscure ports for a service, configuring different ports for clients and servers is not enough. You also need to configure an open port in the firewall. If associated traffic uses both TCP and UDP packets, you'll need to configure that obscure port for both types of packets. You can use the port-scanning software **nmap** to identify open ports on a networked system. Nmap will also detect ports that have been filtered by a firewall. If nmap doesn't detect an open port, it means that port isn't listening.

Because you can use nmap and other tools to easily identify open ports on networked systems, and the services running on those ports can also be identified, moving services around doesn't always make a lot of sense.

Obscurity by Other Means

The use of nonstandard open ports is just one way to obscure local systems and networks. Services can be obscured by their login messages. Services can also be obscured by their error messages. The following directive in the main configuration file for the Secure Shell (SSH) server (/etc/ssh/sshd_config) points to the /etc/issue.net file as the login message for users:

Banner /etc/issue.net

Once configured, you can add appropriate legal warnings or other messages to warn off an attacker or adversary. The same file is associated with login messages to Telnet, should you choose to take the risks associated with that protocol, which transmits sensitive information in cleartext. (For that reason and others, the use of Telnet is not recommended.)

It is widely considered to be a good practice to limit the amount of information services provide. Many services offer a banner that allows clients that are connecting the reassurance that they have connected to the right service. As an example, this is the banner provided by an SMTP server:

```
hodgepodge audit # telnet localhost 25
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 hodgepodge ESMTP Postfix (Ubuntu)
```

You can see that this is a Postfix SMTP server running on an Ubuntu system. You can alter the banner to not advertise that it's a Postfix server. This may slow down an attacker. If you don't do this, the attacker can determine vulnerabilities specifically associated with the Postfix mail server.

More importantly, you can protect yourself by limiting the amount of information your Web server provides. Many Web servers offer up not only their name and version but also the versions of various modules that are installed and functional within the Web server. The following is a fragment of a header from a connection to an Apache Web server:

```
Server: Apache/2.4.7 (Ubuntu)
```

This is clearly an Apache Web server running on an Ubuntu system. More damning, though, is the version number. In some cases, you may be able to get the version number of the PHP module that may be installed on the system. You may also get information about a variety of other modules that have been installed. The best approach—and yes, there is a bit of obscurity involved here—is to reduce the amount of information provided by telling your Web server to not provide anything but the name of the product. In this case, that would be Apache.

Protect with TCP Wrapper

The TCP Wrapper system was originally developed to protect the services controlled through the internet super server and extended internet super server services. It's been extended statically and dynamically to additional services. You'll see how such services can be protected through the /etc/hosts.allow and /etc/hosts.deny configuration files.

What Services Are TCP Wrapped?

As mentioned, there are several groups of services that may be protected by TCP Wrapper. If you're looking for candidate services, examine the following configuration files and commands:

- **Regular internet super server**—Any services configured in the /etc/inetd.conf file can be protected by TCP Wrapper, as long as the TCP daemon (/usr/sbin/tcpd) is included in the server column of the associated configuration directive.
- **Extended internet super server**—Any services configured in the /etc/xinetd.conf file can be protected by TCP Wrapper. Normally, the /etc/xinetd.conf file points to individual services in the /etc/xinetd.d/ directory.
- **Services dynamically linked to libwrap.so.0**—Run the ldd command on the full path to a target daemon, such as the SSH service at /usr/sbin/sshd. If you see a libwrap file in the output, there's a dynamic link between the noted service and TCP Wrapper.

- **Services statically linked to libwrap.so.0**—Run the `strings` command on the full path to a target daemon, such as the Portmap service at `/sbin/portmap`. If you see a `libwrap` or `/etc/hosts.allow` file in the output, there's a static link between the noted service and TCP Wrapper.

All such services can be protected with appropriate directives in the `/etc/hosts.allow` and `/etc/hosts.deny` configuration files.

Configure TCP Wrapper Protection

Any services that have the protection of TCP Wrapper are filtered through rules configured in the `/etc/hosts.allow` and `/etc/hosts.deny` files. Rules in these files may be configured by Internet Protocol (IP) address, domain, or user. Because rules in these files may conflict, they are considered in the following order:

- **/etc/hosts.allow**—Rules in this file are checked first. If the service is explicitly allowed, control is returned to the service. Any rules in `/etc/hosts.deny` are ignored.
- **/etc/hosts.deny**—Any service that makes it beyond `/etc/hosts.allow` looks at rules in this file. Any rule in this file that explicitly denies access to a service is considered. These rules may send a message to the client.
- **If no rule applies**—If no rule is configured in either file, access is granted.

Directives in both configuration files may be set up with the following format:

```
daemons : clients [ : command ]
```

One way to set up maximum protection with TCP Wrapper is to include the following rule in the `/etc/hosts.deny` file. It denies access to all daemons from all clients:

```
ALL : ALL
```

You can then configure other rules in `/etc/hosts.allow` to explicitly allow access to desired services. First, confirm that the SSH service is associated with TCP Wrapper with the following command:

```
$ ldd /usr/sbin/sshd | grep libwrap
```

Once the status of SSH and TCP Wrapper is confirmed, you can allow access to SSH with an appropriate rule in the `/etc/hosts.allow` file. The following rule would allow access to all hosts in the noted IP address network to the Secure Shell service:

```
sshd : 10.22.33.
```

Exceptions can be made with the help of the EXCEPT directive, as illustrated here:

```
sshd : 10.22.33. EXCEPT 10.22.33.44
```

Similar rules can be made by domain names:

```
sshd : .example.org EXCEPT temp1.example.org
```

Alternatively, access can be limited by username:

```
sshd : michael@10.22.33.44
```

Access can also be limited by groupname, where the @ symbol labels `admin` as a group and serves as an identifier of an associated host:

```
sshd : @admin@10.22.33.44
```

You can go farther with the `/etc/hosts.deny` file, as it can be used to send messages to users or administrators. For example, Stanford University recommends the use of the following directive, which should be entered all on one line:

```
ALL:ALL : spawn echo "%d\: %u@%h" |  
↳ /bin/mailx -s "TCPAlert for %H"  
↳ as-admin@example.com : banners /etc/banners
```

The noted directive specifies the daemon (%d), the client username (%u), and the client hostname (%h), and sends it as an e-mail message to the noted `admin@example.com` e-mail address. The noted banner file (`/etc/banners`) can be used to send appropriate welcome or warning messages based on the service being controlled. Stanford University, in its documentation (https://www.stanford.edu/dept/as/ia/security/policies_standards/AS_standards/libwrap_access_control_standard_1.0.html), suggests that its administrators configure the following appropriate welcome message for their users:

Access to this system is limited to authorized users for University business purposes only. Unauthorized access to or use of this system is prohibited and may subject you to civil and criminal prosecution. Use of this system may be monitored for the purpose of maintaining system security and unauthorized access or usage.

Stanford University also suggests that its administrators set up the following message for users who are denied access to TCP Wrapper services. The hostname of the server is substituted for %h in the message.

You have attempted to connect to a restricted access server which you have not been authorized to connect to. Unauthorized access to this system is an actionable offense and will be prosecuted to the fullest extent of the law. It appears that you have attempted to connect from host %h and that host does not have permission/authorization to access this server. Your actions have been recorded and are being reported to the system administrators of this system.

You will now be disconnected!

Of course, the wording should be revised per the policies and characteristics of your organization.

Packet-Filtering Firewalls

When Linux is protected with a packet-filtering firewall, it is protected with a set of rules defined by the `iptables` command. This command can be used to check various parts of a network packet using patterns. If the pattern is matched, you can configure that `iptables` command to accept, reject, deny, or even forward that packet. Different sets of rules can be created for any host in a demilitarized zone (DMZ) as well as private networks behind that DMZ.

Before exploring the `iptables` commands that filter packets, be aware that the `iptables` command is frequently used to masquerade the addresses of a private IP network as a second IP address, typically a public IP address on the Internet.

In addition, any `iptables` rules that forward a packet won't work unless forwarding options in the `/proc` directory, typically in the `/proc/sys/net/ipv4/ip_forward` file. Such options can be configured on a more permanent basis in the `/etc/sysctl.conf` configuration file.

While the options described here apply to Internet Protocol version 4 (IPv4) networking, similar commands and options apply to Internet Protocol version 6 (IPv6) networking. In most cases, the only differences are the addresses along with the applicable command (`ip6tables` instead of `iptables`).

Packet filtering relies on information embedded in the header of each packet. Based on that information, you can create `iptables`-based rules that can redirect the packet. That information can include network cards, IP addresses, port numbers, and more.

This section serves as a brief introduction to the `iptables` command. A good firewall can include dozens of commands, which can identify different packets and/or source/ destination addresses. The `iptables` command can also be configured to specify desired actions to take on those packets. Several excellent books are available that can help you customize `iptables` firewalls in quite a bit of detail.

Basic Firewall Commands

In a typical firewall configuration file, you'll find chains of **iptables** commands.

The rules defined by each **iptables** command in a chain are considered one by one, in order. The basic format of the command is as follows:

```
iptables -t table option direction packet_pattern -j action
```

Now let's examine each of these options in turn.

The **iptables** Table Option

The first option associated with **iptables** is the **-t** option, also known as the table switch. Two basic options for this switch are available (the default is **filter**):

- **filter**—Specifies rules for filtering packets.
- **nat**—Short for network address translation, the **iptables -t nat** command can be used to configure masquerading.

For example, the following command configures masquerading on the first Ethernet device, eth0. This presumes some other network device such as eth1 is connected to and configured on some internal private IP address network:

```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Normally, this is followed by two other commands that allow forwarding between the noted Ethernet network cards:

```
iptables -A FORWARD -i eth0 -o eth1 -m state  
↳ --state RELATED,ESTABLISHED -j ACCEPT
```

```
iptables -A FORWARD -i eth1 -o eth0 -j ACCEPT
```

These two commands provide an introduction to the other **-t** switch, for packet filtering. As the **-t filter** option is the default, it does not need to be included in the command. The first command takes packets from the external interface (eth0) and checks the state of their modules (**-m state**). For those packets where connections have been established or are related to an existing connection, the **iptables** rule jumps (**-j**) to accept the packet. The second command accepts and forwards all packets from network card eth1 destined for outside networks through network card eth0.

Of course, **iptables** rules can be more complex.

Options and Directions for the **iptables** Command

In this section, you'll examine six basic options for the **iptables** command. In most cases, **iptables** options can be specified as letters or more expressive word switches, as follows:

- **-A (--append)**—Appends a rule to the end of a chain
- **-I (--insert) n**—Inserts a rule in a chain of rules, as rule number *n*
- **-R (--replace) n**—Replaces a rule in a chain of rules, as rule number *n*
- **-D (--delete) n**—Deletes a rule in a chain of rules, as rule number *n*
- **-L (--list)**—Lists existing **iptables** rules
- **-F (--flush)**—Deletes (flushes) all configured rules in the specified chain

The last two commands are frequently executed on their own. For example, the **iptables -L** command lists all currently active rules defined in associated configuration files. If there is a problem, you may choose to temporarily disable the current firewall with the **iptables -F** command. As long as any firewall configuration files still exist, you should be able to reactivate the current firewall with the following command:

```
# iptables-restore /path/to/iptables_configuration_file
```

When appropriate, these options should be coupled with a direction based on the source and destination of the packet. The three available directions are as follows:

- **INPUT**—Applies the iptables rule to incoming packets.
- **OUTPUT**—Applies the iptables rule to outgoing packets.
- **FORWARD**—Applies the iptables rule to incoming packets that are being sent (forwarded) to other systems.

Packets and Patterns for the `iptables` Command

Once the option and direction for the `iptables` command is determined, the next step is the pattern match for the packet. In other words, the `iptables` command can be used to check the packet for characteristics such as the IP address. For example, the following options are based on source and destination IP addresses:

- **-s *ip_address***—The rule is applied to packets that come from the noted source address.
- **-d *ip_address***—The rule is applied to packets that are directed to the noted destination address.

Other characteristics may be associated with the protocol and TCP/IP port defined in the packet. For example, the **-p** switch can specify the packet type at the application layer of the TCP/IP protocol suite. The three packet-type options at this level are TCP, UDP, and the Internet Control Message Protocol (ICMP). The TCP and UDP protocols are usually associated with a port number.

Actions for the `iptables` Command

The previous parts of the `iptables` command help define a packet. Once defined, you need to tell the `iptables` command what to do with the packet. The four options that follow are associated with the **-j** switch:

- **-j DROP**—The packet is dropped. No response is sent to the source of the network packet.
- **-j REJECT**—Despite the name of the option, the packet is dropped. However, an error message is sent to the system that is the source of the network packet.
- **-j LOG**—The packet is logged. Other **-j** switches may specify actions that may follow.
- **-j ACCEPT**—The packet is accepted, and is sent to the destination specified in the header of the packet.

Sample `iptables` Commands

With the `iptables` options shown, you should be able to understand several different kinds of commands. As an example, [Figure 7-1](#) shows an excerpt of an `iptables` configuration file. These options are added to the `iptables` command when the firewall/`iptables` service starts. Depending on the complexity of your system or the network your firewall is protecting, the configuration file for your firewalls may include hundreds of `iptables` commands. Such complexity can lead to errors. In many cases, fewer `iptables` rules leads to systems that are better protected because there is less risk for error.

```

:Firewall-INPUT - [0:0]
-A INPUT -j Firewall-INPUT
-A FORWARD -j Firewall-INPUT
-A Firewall-INPUT -i lo -j ACCEPT
-A Firewall-INPUT -p icmp --icmp-type any -j ACCEPT
-A Firewall-INPUT -p 50 -j ACCEPT
-A Firewall-INPUT -p 51 -j ACCEPT
-A Firewall-INPUT -p udp --dport 5353 -d 224.0.0.251 -j ACCEPT
-A Firewall-INPUT -p udp -m udp --dport 631 -j ACCEPT
-A Firewall-INPUT -p tcp -m tcp --dport 631 -j ACCEPT
-A Firewall-INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A Firewall-INPUT -m state --state NEW -m tcp -p tcp --dport 21 -j ACCEPT
-A Firewall-INPUT -m state --state NEW -m tcp -p tcp --dport 2049 -j ACCEPT
-A Firewall-INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
-A Firewall-INPUT -m state --state NEW -m tcp -p tcp --dport 137 -j ACCEPT
-A Firewall-INPUT -m state --state NEW -m tcp -p tcp --dport 138 -j ACCEPT
-A Firewall-INPUT -m state --state NEW -m tcp -p tcp --dport 139 -j ACCEPT
-A Firewall-INPUT -m state --state NEW -m tcp -p tcp --dport 445 -j ACCEPT
-A Firewall-INPUT -m state --state NEW -m tcp -p tcp --dport 80 -j ACCEPT
-A Firewall-INPUT -m state --state NEW -m tcp -p tcp --dport 81 -j ACCEPT
-A Firewall-INPUT -m state --state NEW -m tcp -p tcp --dport 82 -j ACCEPT
-A Firewall-INPUT -m state --state NEW -m tcp -p tcp --dport 902 -j ACCEPT
-A Firewall-INPUT -j REJECT --reject-with-icmp-host-prohibited
COMMIT

```

FIGURE 7-1

Firewall rules from /etc/sysconfig/iptables.

The first two commands specify that INPUT and FORWARD packets are to be accepted and read by the next rule, **Firewall-INPUT**.

Next, the following command accepts all input from the loopback adapter, as specified by the standard loopback device, **lo**:

```
iptables -A Firewall-INPUT -i lo -j ACCEPT
```

The command that follows accepts all ICMP messages. The most common ICMP message is from a **ping** command:

```
iptables -A Firewall-INPUT -p icmp --icmp-type any -j ACCEPT
```

The following directives are associated with ports 50 and 51. They relate to the packets that support Internet Protocol Security (IPSec) communication for authenticating and encrypting each network packet. If you don't plan to set up a virtual private network (VPN) or a similar encapsulated connection, consider omitting these directives:

```
iptables -A Firewall-INPUT -p 50 -j ACCEPT
```

```
iptables -A Firewall-INPUT -p 51 -j ACCEPT
```

The next directive cites port 5353, which is associated with the **multicast Domain Name Service (mDNS) protocol**. This protocol provides DNS functionality for systems configured with zero configuration networking. This is also known as automatic private IP addressing (APIPA) on Microsoft systems and Bonjour on Apple systems. If you don't need mDNS, consider omitting this directive.

```
iptables -A Firewall-INPUT -p udp
    ↵--dport 5353 -d 224.0.0.251 -j ACCEPT
```

The directives that follow are based on a local Common Unix Printing System (CUPS) server. The standard CUPS server communicates with TCP and UDP packets on port 631. If you don't need to set up printers on or printing from this system, consider omitting these directives.

```
iptables -A Firewall-INPUT -p udp
    ↵-m udp --dport 631 -j ACCEPT

iptables -A Firewall-INPUT -p tcp
    ↵-m tcp --dport 631 -j ACCEPT
```

The directive that follows continues established connections:

```
iptables -A Firewall-INPUT -m state
    ↵--state ESTABLISHED,RELATED -j ACCEPT
```

In most firewall configurations, the next set of directives applies only if services on the noted ports are active. To see the associated ports, review the /etc/services file. The final directive sends a REJECT message in response to communication attempts on all other ports.

Many bastion hosts require relatively few `iptables` commands. For example, a bastion host with a Web server that also supports SSH-based remote administration could work with just the following rules:

```
iptables -A INPUT -j Firewall-INPUT
iptables -A FORWARD -j Firewall-INPUT
iptables -A Firewall-INPUT -i lo -j ACCEPT
iptables -A Firewall-INPUT -p icmp --icmp-type any -j ACCEPT
iptables -A Firewall-INPUT -m state
    ↵--state ESTABLISHED,RELATED -j ACCEPT
iptables -A Firewall-INPUT -m state
    ↵--state NEW -m tcp -p tcp --dport 22 -j ACCEPT
iptables -A Firewall-INPUT -m state
    ↵--state NEW -m tcp -p tcp --dport 80 -j ACCEPT
iptables -A Firewall-INPUT -j REJECT
    ↵--reject-with icmp-host-prohibited
```

If your systems support Internet Protocol version 6 (IPv6) networking, be sure to make similar changes to the associated configuration file. On Red Hat systems, that file is /etc/sysconfig/ip6tables.

From this point, the firewall becomes much more complex. Additional changes, discussed next, regulate different kinds of network packets in detail.

Additional `iptables` Rules

The `iptables` rules listed at the end of the previous section might be too simple. Several additional rules of interest may prevent **denial of service (DoS)** attacks, prevent attempts to connect from private IP address blocks, and slow attempts to attack SSH services.

Denial of service rules. Three rules that can prevent or at least slow down DoS attacks are described in the “Linux Packet Filtering HOWTO,” available from <http://www.iptables.org/documentation/HOWTO/packet-filtering-HOWTO.html>. While the references there are to the older Linux 2.4 kernel, the description associated with the `iptables` command still applies today.

The following command limits accepted TCP packets to one per second:

```
# iptables -A Firewall-INPUT -p tcp
  ↵--syn -m limit --limit 1/s -j ACCEPT
```

Incidentally, this command is no longer necessary. You can get equivalent protection from a kernel setting. Just activate the boolean option in the /proc/sys/net/ipv4/tcp_syncookies file. (In other words, change the value of the file from 0 to 1.)

Next, the following command limits connection attempts from port scanners:

```
# iptables -A Firewall-INPUT -p tcp
  ↵--tcp-flags SYN,ACK,FIN,RST RST -m limit --limit 1/s -j ACCEPT
```

Note how access is limited. In other words, this iptables rule accepts one packet with TCP flags per second.

Change default rules. The noted changes to default `iptables` commands may be for naught unless the default rules for packets that are input and forwarded are changed. On a Red Hat system, examine the /etc/sysconfig/iptables file. The default as shown here for those packets is permissive:

```
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
```

These options should be changed to the following:

```
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
```

Restrict ping and related messages. The U.S. National Security Agency has some specific recommendations on how to manage packets associated with the `ping` command, along with related packets associated with the ICMP protocol. Their first recommendation is to remove the following rule from the minimal firewall configuration shown earlier:

```
iptables -A Firewall-INPUT -p icmp
  ↵--icmp-type any -j ACCEPT
```

This rule accepts all ICMP packets, exposing the local system to so-called *ping storms* of packets that deny service to others. The following alternatives accept legitimate responses when local users run the `ping` command:

```
iptables -A Firewall-INPUT -p icmp
  ↵--icmp-type echo-reply -j ACCEPT
iptables -A Firewall-INPUT -p icmp
  ↵--icmp-type destination-unreachable -j ACCEPT
iptables -A Firewall-INPUT -p icmp
  ↵--icmp-type time-exceeded -j ACCEPT
```

The responses are as implied by the options attached to the `--icmp-type` switch: for example, `echo-reply` accepts ICMP messages that respond to a `ping` command. If you want to allow others to ping your system, with limits, the following command rule limits responses to `ping` commands to one per second.

```
iptables -A Firewall-INPUT -p icmp
  ↵--icmp-type echo-request -m limit --limit 1/s -j ACCEPT
```

If you don't want local systems to respond to `ping` command requests, substitute the following command rule:

```
iptables -A Firewall -p icmp
  ↵--icmp-type echo-request -j DROP
```

Block information from suspicious IP addresses. When communicating on the Internet, if you identify a packet with a source IP address that should be on a private network, be suspicious. Such addresses may be an attempt by a malicious user to hide the identity of his or her system. To that end, secure firewalls will drop packets with a source IP address associated with a private network or any other address that should not be assigned to a regular system. For IPv4 addresses, such rules should include the following:

```
iptables -A Firewall-INPUT -i eth0
  ↳ -s 10.0.0.0/8 -j LOG --log-prefix "Dropped private class A address"

iptables -A Firewall-INPUT -i eth0
  ↳ -s 172.16.0.0/12 -j LOG --log-prefix "Dropped private
    ↳ class B address"

iptables -A Firewall-INPUT -i eth0
  ↳ -s 192.168.0.0/16 -j LOG --log-prefix "Dropped private
    ↳ class C address"

iptables -A Firewall-INPUT -i eth0 -s
  ↳ 224.0.0.0/4 -j LOG --log-prefix "Dropped multicast address"

iptables -A Firewall-INPUT -i eth0 -s
  ↳ 240.0.0.0/5 -j LOG --log-prefix "Dropped class E address"

iptables -A Firewall-INPUT -i eth0
  ↳ -d 127.0.0.0/8 -j LOG --log-prefix
    ↳ "Dropped attempt to connect to the loopback address"
```

If you've set up IPv6 addresses, corresponding rules should be added to the corresponding firewall configuration file.

Slow attacks on SSH services. With port scanners and a little work, a malicious user should find it easy to identify not only open ports, but also the traffic expected on those ports. Once that's done, that person can try some typical hostnames and dictionaries of standard passwords. But dictionaries are large. A malicious user who tries to break into a system with a dictionary attack can be slowed down with two `iptables` commands similar to the following:

```
iptables -A Firewall-INPUT -i eth0 -p tcp -m tcp
  ↳ --dport 22 -m state --state NEW -j SSH_RULE

iptables -A SSH_RULE -i eth0 -p tcp -m tcp --dport 22
  ↳ -m state --state NEW -m recent --name SSH --seconds 60
  ↳ --hitcount 3 --rttl --name SSH -j DROP
```

The first command specifies that the firewall jumps (- j) to the `SSH_RULE` firewall rule. The second command limits the number of login attempts to three. After three failures, the external user has to wait 60 seconds to try again. That should discourage even users who are just trying a few thousand dictionary words with a typical username.

The `iptables` Service

On Red Hat systems, there is an `iptables` service in the `/etc/init.d/` directory normally started during the boot process. It reads options from the `/etc/sysconfig/iptables` configuration file. On Ubuntu systems, when an `iptables` firewall is configured, an additional package, `iptables-persistent`, will be required to have a service script installed that will load the firewall rules at boot time.

The `iptables` package also installs the `iptables-save` and `iptables-restore` commands. These programs will save the rules in memory to a file or read from a file and load them into memory. This would then make them the functional rules on the system. Once you have a way to store rules to disk and then reload them, you have the ability to install rules and make them persistent across boots. This assumes that there is a script that will both store the existing rules as well as load up the file of rules into memory.

Examples of Firewall-Management Tools

If all this seems confusing, GUI-based tools can be helpful. The basic firewall described earlier was created with the Red Hat Security Level Configuration tool, which is shown in [Figure 7-2](#). It allows easy configuration of a firewall. The standard options shown in the Trusted Services window are based on external access to the noted local servers. The port numbers shown in the Other Ports area at the bottom of the screen are the port numbers of additional traffic allowed into the system.

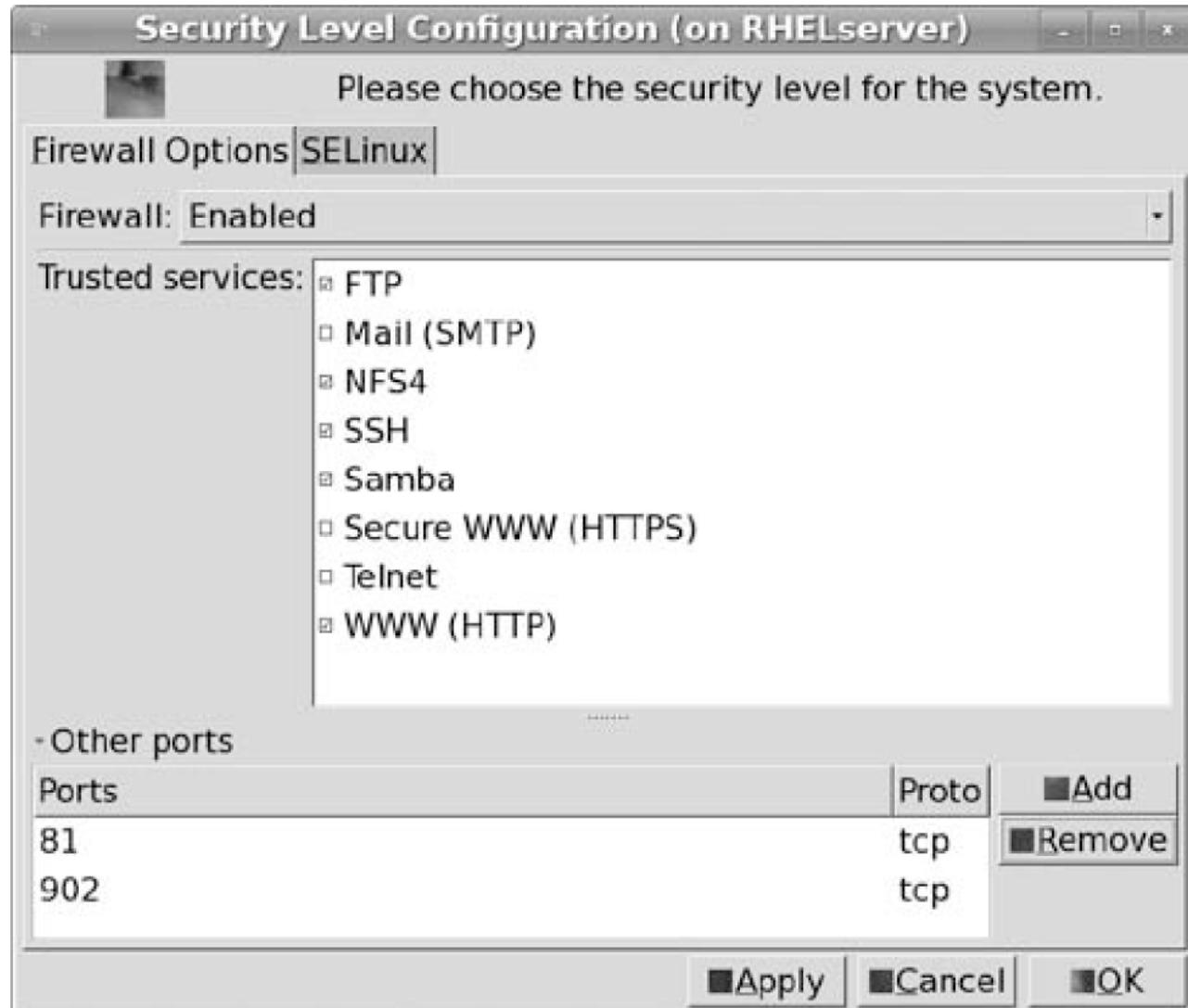
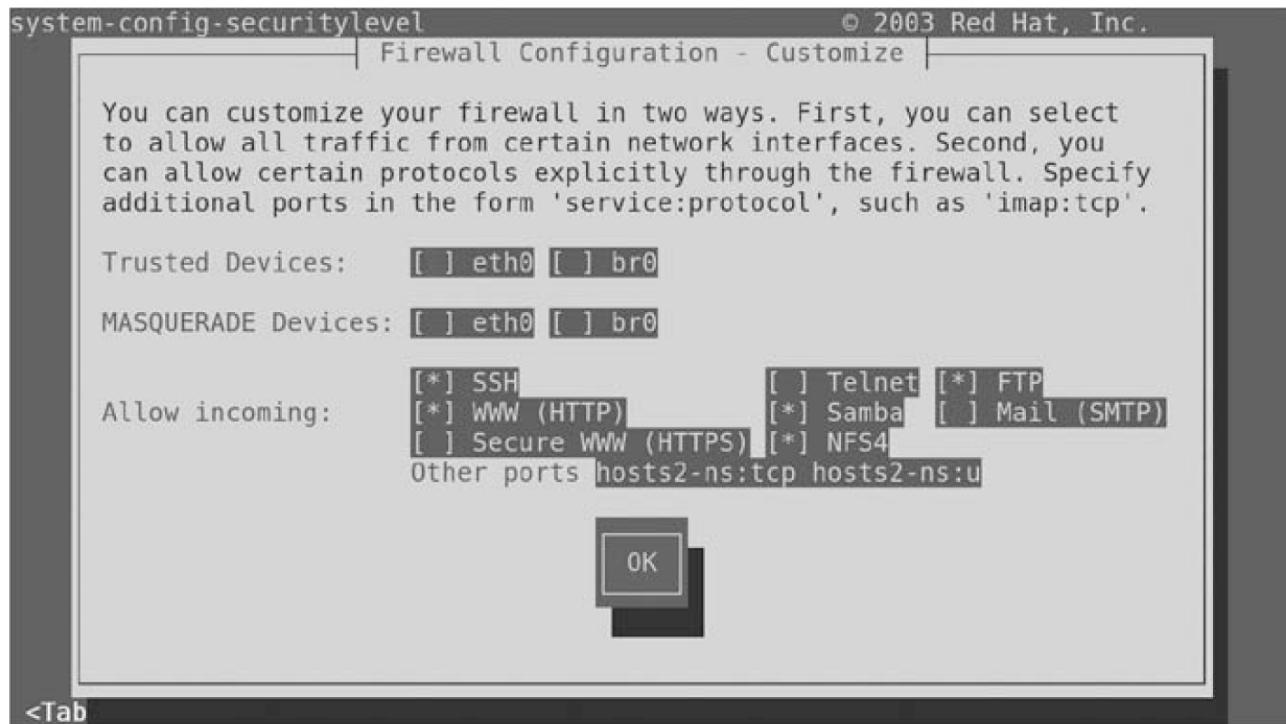


FIGURE 7-2

The GUI Security Level Configuration tool.

**FIGURE 7-3**

The console-based Security Level Configuration tool.

Differences between GUI tools can be instructive. For example, Figure 7-3 shows the console-based version of the same tool. This low-level graphical tool includes more features than its GUI cousin, as it supports masquerading as well as the ability to exempt traffic from a specific network card. And yes, it can be run on a terminal, without a GUI. Both the text-based interface and the graphical interface are still available in newer versions of Red Hat, but they won't work if firewalld is the firewall software instead of iptables. With newer versions of Red Hat, firewalld is the default firewall used. You can still use iptables if you have disabled firewalld. In that case, you can use either of these programs to configure your firewall.

Now for some contrast, examine the GNOME uncomplicated firewall (Gufw) firewall- configuration tool, which you can start in an Ubuntu GUI with the `gufw` command. As shown in Figure 7-4, it includes options similar to those available for the Red Hat Security Level Configuration tool.

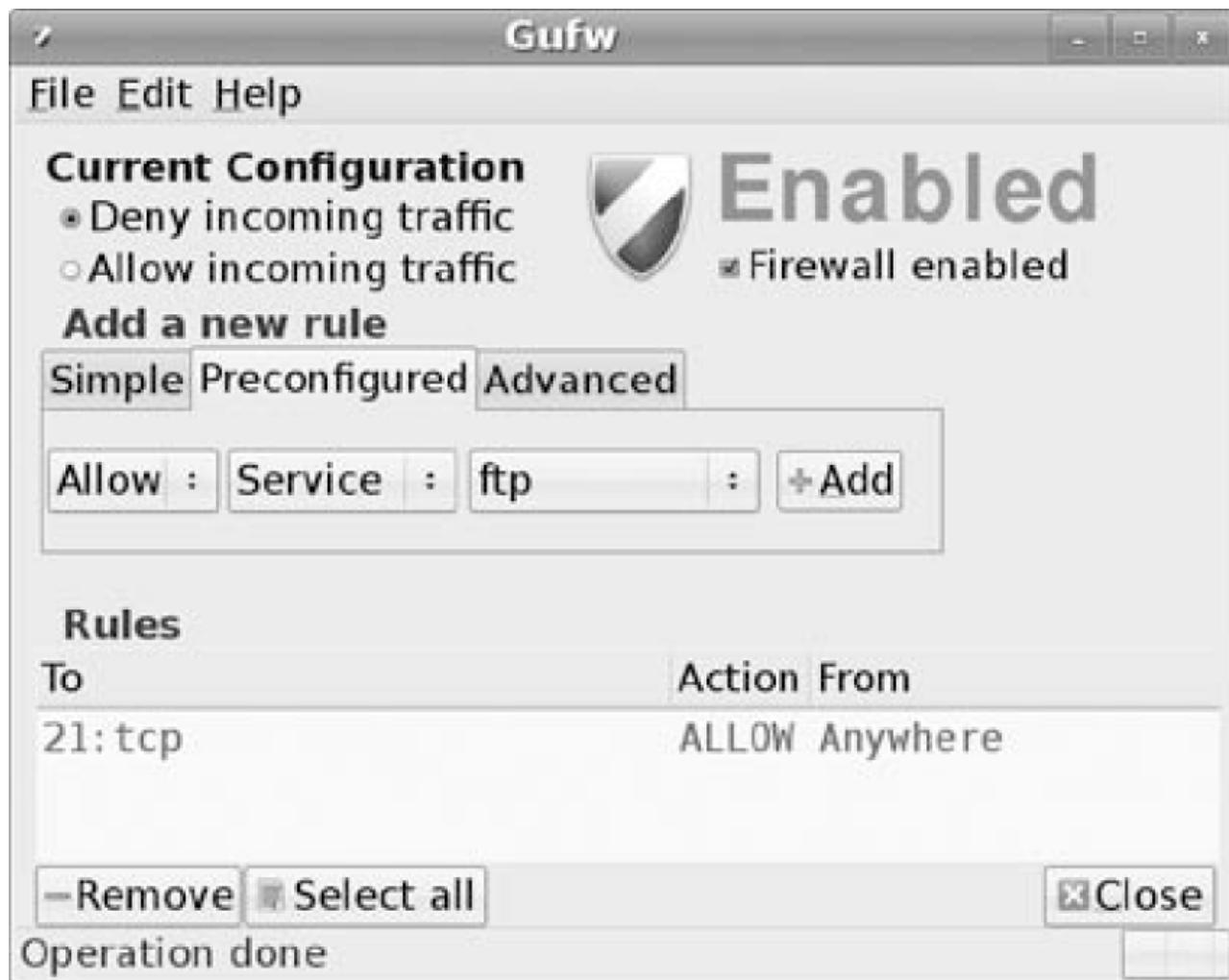


FIGURE 7-4
The GNOME uncomplicated firewall (Gufw) tool.

If you believe firewall commands are confusing, try some of these tools. More important, examine the effect of a change through these tools on the associated iptables configuration files. Save an iptables configuration file. Make a change with a configuration tool. Review the changes in the iptables configuration file. The changes should help you understand which `iptables` command can help secure different systems.

Firewalld

Red Hat no longer uses iptables in the latest releases of its operating systems. Instead, Red Hat uses a system called **firewalld**. Firewalld is the next generation of Linux firewall. With iptables, you get a system where you can quickly and easily add firewall rules using the `iptables` command, but these rules aren't persistent without additional service support. You also get a fairly complex way to configure rules, and that additional complexity can lead to mistakes. If you don't get the command-line switches to the `iptables` command correct, you don't get the results you want in your firewall. The `iptables-save` and `iptables-restore` commands are used to save and restore the firewall rules but they are stored as a set of command-line rules to be applied with the `iptables` command. These can be challenging to read. Using `iptables` is also complicated when you have mobile devices like laptops. You may want a different set of rules when you have your laptop at McDonald's or Starbucks than when you have your laptop at home.

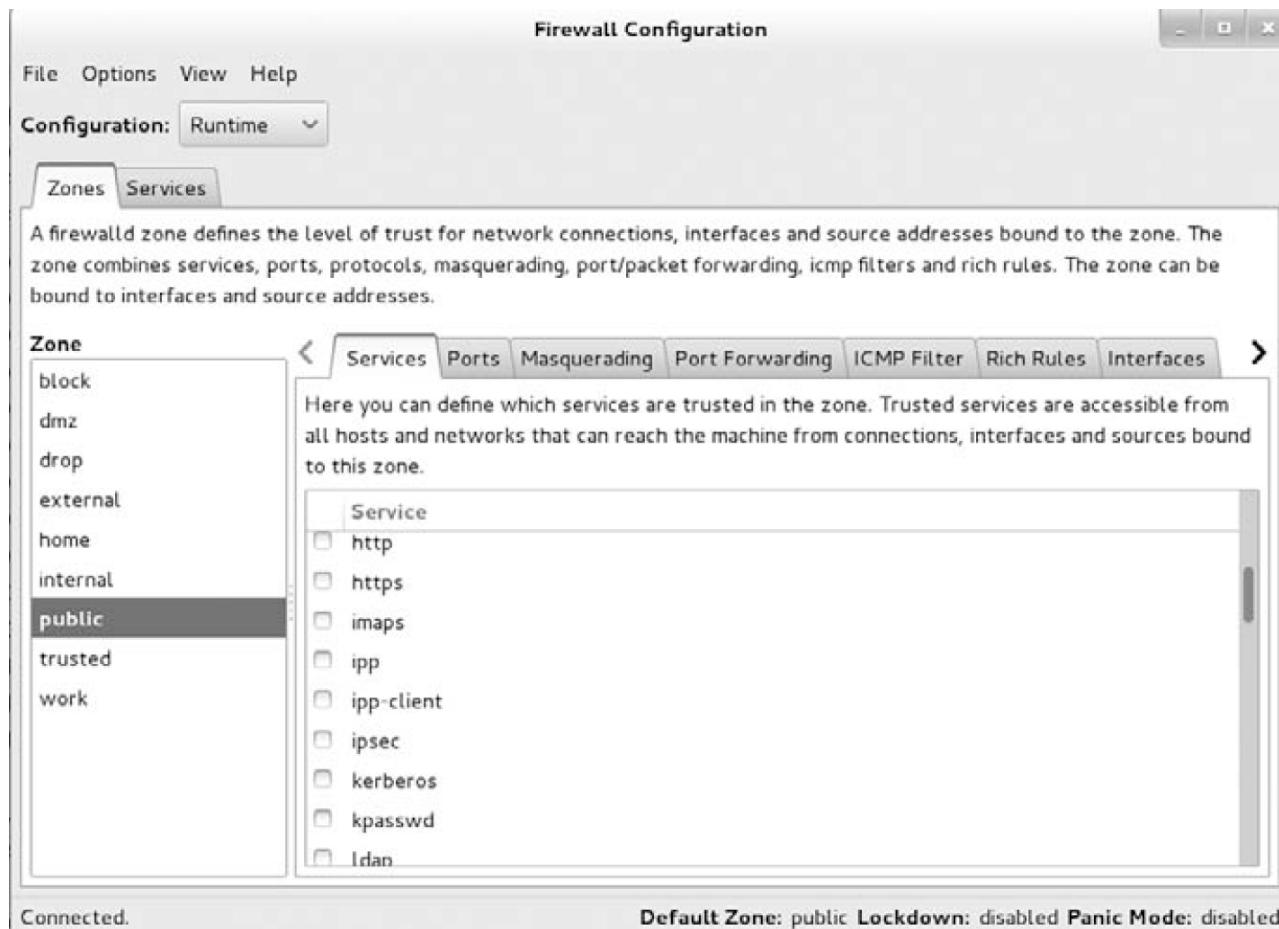
Much like Windows, firewalld implements zones as a way of quickly implementing rules based on where you are. In the simplest terms, a zone is a set of rules that can be implemented based on where you are located. The zones are

given names that are easy to understand. As an example, there is a zone named public. You would apply this zone to any interface that is connected to a network in a public location, like an open Wi-Fi network at a coffee shop. According to the Security Guide from Red Hat at https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Security_Guide/sec-Using_Firewalls.html, the following are default zones that come already configured with firewalld:

- **drop**—Any incoming network packets are dropped. There is no reply. Only outgoing network connections are possible.
- **block**—Any incoming network connections are rejected with an **icmp-host-prohibited** message for IPv4 and **icmp6-adm-prohibited** message for IPv6. Only network connections initiated from within the system are possible.
- **public**—For use in public areas. You do not trust the other computers on the network not to harm your computer. Only selected incoming connections are accepted.
- **external**—For use on external networks with masquerading enabled especially for routers. You do not trust the other computers on the network not to harm your computer. Only selected incoming connections are accepted.
- **dmz**—For computers in your demilitarized zone that are publicly accessible with limited access to your internal network. Only selected incoming connections are accepted.
- **work**—For use in work areas. You mostly trust the other computers on networks not to harm your computer. Only selected incoming connections are accepted.
- **home**—For use in home areas. You mostly trust the other computers on networks not to harm your computer. Only selected incoming connections are accepted.
- **internal**—For use on internal networks. You mostly trust the other computers on the networks not to harm your computer. Only selected incoming connections are accepted.
- **trusted**—All network connections are accepted.

Using firewalld, you specify services and the ports that are associated with those services. Once you have services configured, you can then enable or disable those services in each of the zones. Another advantage is the Firewall Configuration dialog box, shown in [Figure 7-5](#). It gives you the ability to apply any changes to the runtime configuration or the persistent configuration. Any runtime configuration change would happen to the live system as soon as the change was applied. Persistent changes would be written out to disk.

Of course, firewalld also has all of the capabilities of iptables in addition to the zones. You can use firewalld for network address translation (NAT), which can be done using the Masquerading tab. You can also add rules for ports that may not be associated with defined services. Using the Rich Rules feature, you can add logging and auditing rules as well as rate-limiting rules.

**FIGURE 7-5**

The Firewall Configuration dialog box.

The zone configurations are stored in XML files in /etc/firewalld/zones. The **public** zone on a default installation looks like the following. You can see that there are two services that are configured in that zone. These two services are enabled in the **public** zone, meaning that both ssh and DHCP client connections for IPv6 will be allowed through the firewall. One of the advantages of XML, although it does create large configuration files, is the fact that these files are self-documenting. You can see the name of the zone and its description along with the list of services.

```
<?xml version="1.0" encoding="utf-8"?>
<zone>
    <short>Public</short>
    <description>For use in public areas. You do not trust the other
    computers on networks to not harm your computer. Only selected incoming
    connections are accepted.</description>
    <service name="dhcpcv6-client"/>
    <service name="ssh"/>
</zone>
```

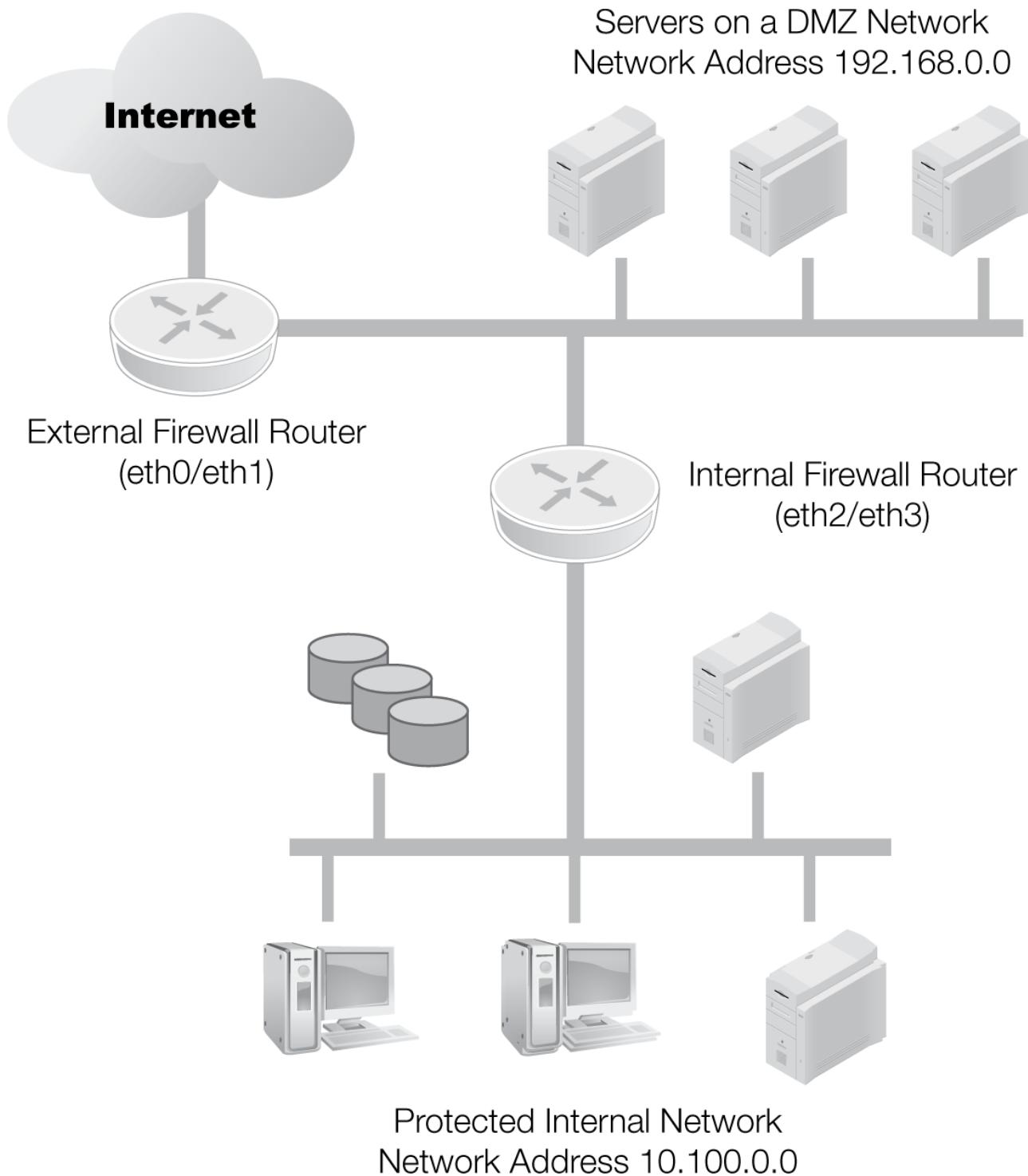
A Firewall for the Demilitarized Zone

If you have servers that provide services such as Web sites and FTP-based file sharing, consider configuring them in a DMZ. A DMZ network is separate from but fits between the Internet and an otherwise wholly private network, as shown in [Figure 7-6](#).

The networks described in [Figure 7-6](#) require additional explanation. [Figure 7-6](#) depicts three distinct networks. The wide area network (WAN) may be a corporate network or the Internet. The DMZ is an internal network with one private IP network address. The protected internal network has a distinct and separate private IP network address. To translate between networks, both routers are configured to masquerade IP addresses. The network addresses shown are just two possible examples. Traffic between the WAN (Internet) and the protected internal network is filtered through two different firewall routers.

Each firewall router has two network devices. In [Figure 7-6](#), the first Ethernet device (eth0) on the external firewall router is connected to the WAN (Internet). The second Ethernet device (eth1) on that same router is connected to the DMZ network. All traffic directed to and from the DMZ and the protected internal network is filtered through a firewall on this router.

Each server on the DMZ network is used to serve data to users on the Internet. Each server has a single function, such as serving Web pages, sharing files via the FTP service, or sharing authoritative name service information via a caching Domain Name System (DNS) name server. The external firewall router should stop network traffic that would otherwise harm internal systems. In addition, that external firewall router should filter requests for each service and redirect appropriate requests to the right server.

**FIGURE 7-6**

A DMZ belongs between the Internet and an internal network.

Now look at the internal firewall router. The first Ethernet device (eth2) on that router is connected to the DMZ network. The second Ethernet device (eth3) on that router is connected to the protected internal network. The internal firewall router should also stop the same network traffic that would otherwise harm internal systems. It should be designed as if all of the systems are untrusted. Assume they have been compromised or are otherwise under the control of someone who may want to gain unauthorized access to the internal network.

The only network traffic that should be directed from the external firewall router to the internal network is information requested by those internal network clients. That traffic should be filtered to stop inappropriate traffic such as DoS, rapid attempts at repeated logins, and so on.

In many cases, the functionality of both routers is incorporated into one system with three network cards. That system would include the functionality of both firewalls. If configured on a Linux system, that system should be configured with packet forwarding for both IPv4 and IPv6 traffic in the /etc/sysctl.conf file and /proc/ settings described earlier in this chapter.

Two examples of commands that would enable forwarding to a Web server on the DMZ are as follows. The first command routes all incoming TCP packets associated with port 80 to the system on IP address 192.168.100.50. It assumes the second Ethernet card (eth1) is the network card between the DMZ and the Internet.

```
iptables -A PREROUTING -t nat -i eth1
    ↪ -p tcp --dport 80 -j DNAT --to 192.168.100.50:80
```

The second rule specifically accepts TCP packets sent to port 80 into the DMZ network.

```
iptables -A Firewall-INPUT -p tcp
    ↪ -m state --state NEW --dport 80 -i eth1 -j ACCEPT
```

A Firewall for the Internal Network

Assuming a DMZ is configured, you'll want to set up a separate firewall between the DMZ network and the internal network. The first two command rules for such a firewall are shown here. The network card between the internal network and the DMZ is the first Ethernet card (eth0); the network card between the DMZ and the Internet is eth1. The first command rule is as follows:

```
iptables -A FORWARD -i eth1 -o eth0 -m state
    ↪ --state ESTABLISHED,RELATED -j ACCEPT
```

It's followed by the following rule, which allows all traffic from the protected internal network out to the Internet. Of course, if organizational policies dictate, you may want to limit such traffic.

```
iptables -A FORWARD -i eth0 -o eth1 -j ACCEPT
```

To allow systems on the internal network to communicate with the Internet, you need a rule that masquerades the IP addresses of the internal network directly to the network card connected to the Internet:

```
iptables -t nat -A POSTROUTING -o eth1
    ↪ -j SNAT --to public_IP_address
```

The `public_IP_address` is the IPv4 address between the DMZ and the Internet.

Alternate Attack Vectors

Today, most attention is focused on regular Internet connections, as if high-speed network connections are the only way to get onto the Internet. Older networks may still have legacy connections such as telephone modems and even serial ports.

Security extends beyond regular network connections. For example, the electromagnetic emissions of a computer monitor can be used by malicious users to identify critical information. Wireless networks also involve a special type of electromagnetic emission.

In addition, any malicious user who breaks into a local system may have access to nonstandard services. Some services can be run with administrative permissions and should be regulated carefully.

Attacks Through Nonstandard Connections

Most home users connect through cable modems, Digital Subscriber Line (DSL) adapters, and similar network connections over the public Internet. But not all users have access to such high-speed lines. Many servers still support connections over serial ports. Some systems can even track the electromagnetic output of devices such as monitors to decipher user input.

The Telephone Modem and Related Devices

This section covers those devices that communicate over the older system known as the **public switched telephone network (PSTN)**. A **telephone modem** is a device that modulates data to the sine waves associated with voice transmission. A second telephone modem on the target network demodulates the sine waves back to digital data. Those features make up the terms modulator-demodulator, or modem for short.

Many users still have access to telephone modems. Because there are still many areas without affordable high-speed Internet access, telephone modems are still, in many cases, the only way a user can get onto the Internet. Nevertheless, telephone modems are a frequently forgotten part of the network infrastructure.

Because Linux is fairly easy to administer from the command-line interface, the 56Kbps transmission speed associated with a telephone modem is sufficient for administrative connections. It's also fast enough for older users, who are accustomed to text-based tools for e-mail, Web browsing, and more.



WARNING

Beware: Malicious users have access to tools such as war dialers, which can check phone numbers quickly for the sound associated with a telephone modem. You can almost count on malicious users finding any modem connection on your network with security weaknesses.

But connections over the PSTN go beyond telephone modems. Related connections include **Integrated Services Digital Network (ISDN)** devices.

While the connections for such devices vary on older networks, logins from such devices to Linux systems are generally governed by **Remote Authentication Dial In User Service (RADIUS)**. Authentication via RADIUS can be configured through a local shadow password suite, LDAP servers, or even a preconfigured group of RADIUS usernames and passwords.

RADIUS supports both the **Password Authentication Protocol (PAP)** and the **Challenge-Handshake Authentication Protocol (CHAP)**, both protocols for validating users before allowing access. CHAP is a bit more secure, as it can send a challenge to a client system at any time.

Serial Ports

Serial ports are included in this section because communication can be enabled over such ports with a so-called *null modem*. Such a connection is commonly used to enable a connection to a text console, which is sufficient to administer Linux systems. In other words, any user who has physical access to the system can connect a laptop terminal to that serial port and log in directly to that system, bypassing security measures such as firewalls that may be present on that system or network.

At the very least, that user can then identify the operating system, potentially force a reboot, and possibly break into an unprotected account on the target machine.

To that end, you need to consider different ways to lock out the serial port. You could add a physical lock or perhaps seal the port with clay. If it's connected to a removable internal card, you could remove that card. If the port is controlled within the Basic Input/ Output System (BIOS) or Unified Extensible Firmware Interface (UEFI) menus, you could disable that port.

Tracking Other Electromagnetic Output

Electrical signals, when unshielded, send information over more than just wires. Just as electricity is produced by turning a magnet inside a coil of electrical wires, magnetic force can be produced by electricity running through a wire. This phenomenon is an effect of electromagnetic radiation.

One method used by some attackers to take advantage of electromagnetic output is known as **Van Eck phreaking**, first proven by Dutch researcher Wim van Eck. He showed it's possible to take the radiation from modern computer displays and recover the associated image. Van Eck phreaking works with both cathode ray tube (CRT) and liquid crystal display (LCD) monitors. Similar attacks have proven successful against the electromagnetic radiation generated by keyboards, printers, and other devices.

Attacks on Scheduling Services

While some services provide services directly to users, either remote or local, other services exist to manage the system. When you are running and managing an operating system, you want to have a number of tasks scheduled. When you schedule tasks, you need a service running constantly that will take care of running those tasks for you. Linux comes with two: cron and **at**. The next sections explain those services.

FYI

The North Atlantic Treaty Organization (NATO) has conducted a series of studies of electromagnetic output from computers. The study is code-named TEMPEST. The results were documented in National Security Telecommunications and Information Systems Security Advisory Memoranda (NSTISSAM), which have been partially declassified. The associated standards may be cost-prohibitive for most organizations.

The cron Service

The cron service is a scheduler for jobs to be run on a regular basis. In general, cron jobs can be configured to run hourly, daily, weekly, or monthly. Administrative cron jobs are configured in the /etc/crontab configuration file. Users can run their own cron jobs if their accounts are allowed or not denied in the /etc/cron.allow and /etc/cron.deny files. The contents of these files are simple: They contain a list of users, one user per line.

If neither of these files exists, access to the cron service varies by distribution. If you're working on a Red Hat system, access is limited to users with administrative privileges. If you're working on an Ubuntu system, all users are able to access the cron service.

If only the /etc/cron.deny file exists, all users not listed in that file are allowed to create their own cron jobs. If both files exist, then only those users listed in /etc/cron.allow are permitted to set up their own cron jobs, and the contents of the /etc/cron.deny file are ignored. In other words, if /etc/cron.allow exists, it doesn't matter whether the /etc/cron.deny file exists.

To limit the damage associated with a compromised account, it can be helpful to add the usernames of at least service user accounts to /etc/cron.deny. If a malicious user compromises one of those accounts, that person at least won't be able to set up administrative jobs with those accounts. Of course, such accounts should never be included in the /etc/cron.allow file.

The at Service

The at service is a scheduler for jobs to be run on a one-time basis. Such jobs can be configured to run at any time in the future. Access to the at service depends on the status of the /etc/at.allow and /etc/at.deny files. Normally, the /etc/at.deny file exists when the at package is installed.

If an /etc/at.allow file exists, only users listed in it may submit jobs using the at service. If the /etc/at.allow file does not exist, the service will allow access to any user who is not listed in the /etc/at.deny file. If neither file exists, access is limited to the root administrative user.

To limit the damage associated with a compromised account, it can be helpful to add the usernames of at least service user accounts to /etc/at.deny. If a malicious user compromises one of those accounts, that person at least won't be able to set up administrative jobs with those accounts. Of course, such accounts should never be included in the /etc/at.allow file.

While it would be safest to limit access to the root administrative account, that's not always practical. It's a terrific convenience for users to be able to run jobs on systems at night, during periods when the computer is not heavily used.

Wireless-Network Issues

Wireless networking is nearly ubiquitous in public areas. It's not difficult to find free wireless access points. Many of these access points are not secure. In fact, many do not even require passwords. There are a lot of factors that go into hardening your wireless networks to protect against unauthorized access and guard the integrity of the data being transmitted over those networks.

The risks of wireless hardware extend to most off-the-shelf systems. Most systems include active wireless network cards by default. When analyzing the state of your wireless security, don't forget any Bluetooth devices that might be installed.

While there are problems with wireless security, it's important to have at least an acceptable use policy for the variety of available wireless hardware. Whether a user is connecting to a network from a laptop, a mobile phone, a netbook, a tablet, or some other networkable device, an acceptable use policy can help users understand the risks. The following are areas that are commonly covered in an acceptable use policy:

- Access policy
- Authentication policy
- Accountability policy
- Availability
- System and network maintenance policy
- Acquisition guidelines
- Violations reporting
- Audit policy

Linux and Wireless Hardware

Many open source security experts suggest that you use wireless access points (WAPs) with customizable firmware. Two major Linux-based firmware solutions are **wireless receiver-transmitter (WRT) software**, known as DD-WRT and OpenWrt. Both firmware solutions are available on a number of commercial off-the-shelf wireless routers.

Finding native Linux support for wireless hardware can be a challenge. The availability of a Linux wireless driver often depends on the chipset. Some manufacturers change chipsets while retaining the same part number for their wireless devices.

While there are ways to use Microsoft drivers on Linux systems based on software wrappers, such options don't enable the full capability of the devices and may even subject such devices to malicious user break-ins based on troubled Microsoft libraries. When fully featured, Linux wireless cards can be configured in several different modes. The details are not directly related to security.

Encrypting Wireless Networks

Wireless networks have three protocols available for encryption. This has been an iterative process. As wireless networking has become more widespread, encryption capabilities have become stronger.

technical TIP

The encryption of WEP included vulnerabilities within the initialization vector that helped to generate the key. As a result, WEP is fairly easy to crack, meaning that communications encrypted with WEP can be decrypted without much difficulty. There are open source tools that can be used to perform the cracking.

Encryption algorithms such as WPA and WPA2 may be cracked if they use weak pre-shared keys based on dictionary words. For this reason, you should always use a long, strong pre-shared key constructed in a manner similar to a password. More information on this process is available from <http://www.aircrack-ng.org/> and

<http://www.airtightnetworks.com/home/resources/knowledge-center/wpa-wpa2-tkip-attack.html>. Even with secure wireless protocols, networks are still susceptible to other attacks such as the spoofing of hardware addresses and honeypots.

- **Wired Equivalent Privacy (WEP)**—WEP uses stream ciphers and checksums for confidentiality and integrity. It works with open-system and shared-key authentication. While it works with keys of up to 256 bits, WEP packets can be intercepted and decrypted. Standard 40-bit WEP keys can be decrypted with tools available from normal Linux repositories in a matter of minutes.
- **Wi-Fi Protected Access (WPA)**—WPA uses the Temporal Key Integrity Protocol (TKIP), which uses a unique encryption key on every packet.
- **Wi-Fi Protected Access, version 2 (WPA2)**—WPA2 uses the Counter Mode with Cipher Block Chaining Message Authentication Code Protocol (CCMP), with the Advanced Encryption Standard (AES) algorithm.

In addition, you can add a little obscurity by hiding the name of the WAP, also known as the extended service set identifier (ESSID), as well as by changing the password frequently.

In addition, you can use a **wireless intrusion detection system (WIDS)**. A WIDS is software, such as aircrack-ng, that can help detect attacks on a wireless network. While the aircrack-ng package can be used to recover lost WEP and WPA keys, it can also monitor wireless networks for intrusions. When combined with the kismet package, it can be used to detect a range of attacks from floods to packet injections.

Bluetooth Connections

When considering wireless connections, don't forget Bluetooth devices. Not all such devices support encryption. Bluetooth devices are configured in files in the /etc/bluetooth/ directory. Individual devices may be configured in the hcid.conf file in that directory. If the Bluetooth device can support encryption, you can enable it in the hcid.conf file. Add the following directive to the stanza associated with your Bluetooth device:

```
encrypt enable;
```

Security Enhanced Linux

SELinux was developed by the U.S. National Security Agency (NSA) as a mandatory access control system. It provides an additional layer of protection using Linux security modules included in the kernel. The NSA helped Red Hat integrate SELinux into Red Hat Enterprise Linux. Because the NSA released the source code for SELinux under open source licenses, the same software is available for other Linux distributions.



NOTE

Because SELinux is primarily used on Red Hat systems, the focus of this section is on Red Hat-based configuration files. Nevertheless, it's possible to configure SELinux on other AppArmor-focused distributions, including Ubuntu and SUSE Linux. Just remember: SELinux and AppArmor can't be used on the same system.

While there are a number of detailed SELinux configuration files, the basic configuration is stored in the /etc/sysconfig/ directory, in the selinux file. It's actually linked to the /etc/selinux/config file. The two directives in this file are simple:

- **SELINUX**—May be set to `enforcing`, `permissive`, or `disabled`. The `enforcing` and `disabled` modes are self-explanatory. In contrast, `permissive` mode logs all policy violations in the messages and audit/audit.log files in the /var/log/ directory.
- **SELINUXTYPE**—May be set to `targeted` or `strict`. The default targeted daemons include the following services: Dynamic Host Configuration Protocol (DHCP), Apache, the Berkeley Internet Name Domain (BIND) version of the Domain Name Service (DNS), the Name Service Cache Daemon (NCSD), the Network Time Protocol (NTP) service, the Portmap service, the Simple Network Management Protocol (SNMP) service, the Squid Web Proxy server, and the system logging service.

Detailed configuration options are listed in the `/selinux/` directory. If you've changed SELINUX to or from `disabled`, you may need to reboot the system to make sure all related files are properly labeled. Especially if SELINUX is being enabled, this process can add a number of minutes to the boot process.

SELinux access permissions are represented by bitmaps known as **access vectors**. An access vector is an access permission represented by a bitmap. The way they're stored is known as an **access vector cache (AVC)**.

FYI

SELinux is an intimidating subject for many users. The complexity associated with SELinux configuration is one reason many administrators use AppArmor. One of the problems is that it's time consuming to configure SELinux to work with custom configurations. If you're having problems with SELinux and a specific service, you may want to temporarily disable SELinux protection for that service. This is not ideal, but at least SELinux protection will still be available for other targeted services, as defined by the `SELINUXTYPE` directive in the `/etc/sysconfig/selinux` configuration file.

On the other hand, Novell no longer directly employs AppArmor developers. Nevertheless, AppArmor development continues, courtesy of developers associated with the openSUSE project (<http://en.opensuse.org/AppArmor>).

The Power of SELinux

On one level, SELinux is conceptually like a good firewall. All actions are first denied. SELinux contexts are then created for users, applications, and services to work normally. These contexts are associated with just the configuration files and program libraries needed by authorized users to make desired applications and services work.

SELinux can be used to assign access and transition rights to every user, application, process, and file. Interactions between these entities are governed by SELinux policies. If any of these entities don't belong, then neither access nor transition rights are granted. For example, if the service user named apache somehow gets full root administrative access, that user still won't be allowed to modify a different service such as an FTP server.

Basic SELinux Configuration

The simplest way to find the current status of SELinux on the local system is with the **`sestatus`** command. It confirms the configuration shown in the `/etc/sysconfig/selinux` file. If you want to find the current SELinux status of files in the local directory, try the `ls -Z` command. As an example, look at the SELinux characteristics of the default FTP server configuration directory:

```
$ ls -Z /var/ftp/
drwxr-xr-x root root system_u:object_r:public_content_t pub
```

The output shows the SELinux characteristics of the `pub/` subdirectory in the `/var/ftp/` directory. If you'd rather configure a different directory for the FTP server, SELinux contexts on that directory must be the same. In other words, that directory must be associated with the system user (`system_u`), system objects (`object_r`), on a directory type associated with public information (`public_content_t`).

One way to change another directory to conform is by running some **`chcon`** commands. Specifically, you would want to change the SELinux context associated with that user, system, and directory. For example, to change the user and type contexts on a `/srv/ftp/` directory, you'd run the following command:

```
# chcon -u system_u -t public_content_t /srv/ftp/
```

A number of other commands can help you configure SELinux from the command-line interface. As you customize SELinux for your systems, there will be many cases where you modify SELinux policies to enable normal operation of regular services on the target system.

Configuration from the Command Line

On Red Hat systems, most of the important command-line SELinux tools are part of the policycoreutils, setools, and libselinux-utils packages. That package includes a couple of configuration files: sestatus.conf and selinux/restorecond.conf in the /etc/ directory. These files include lists of critical files and services to be watched for changes. You may want to add more files and services to these lists. The commands from the three aforementioned packages are described briefly in [Table 7-2](#).

TABLE 7-2 Basic SELinux control commands.

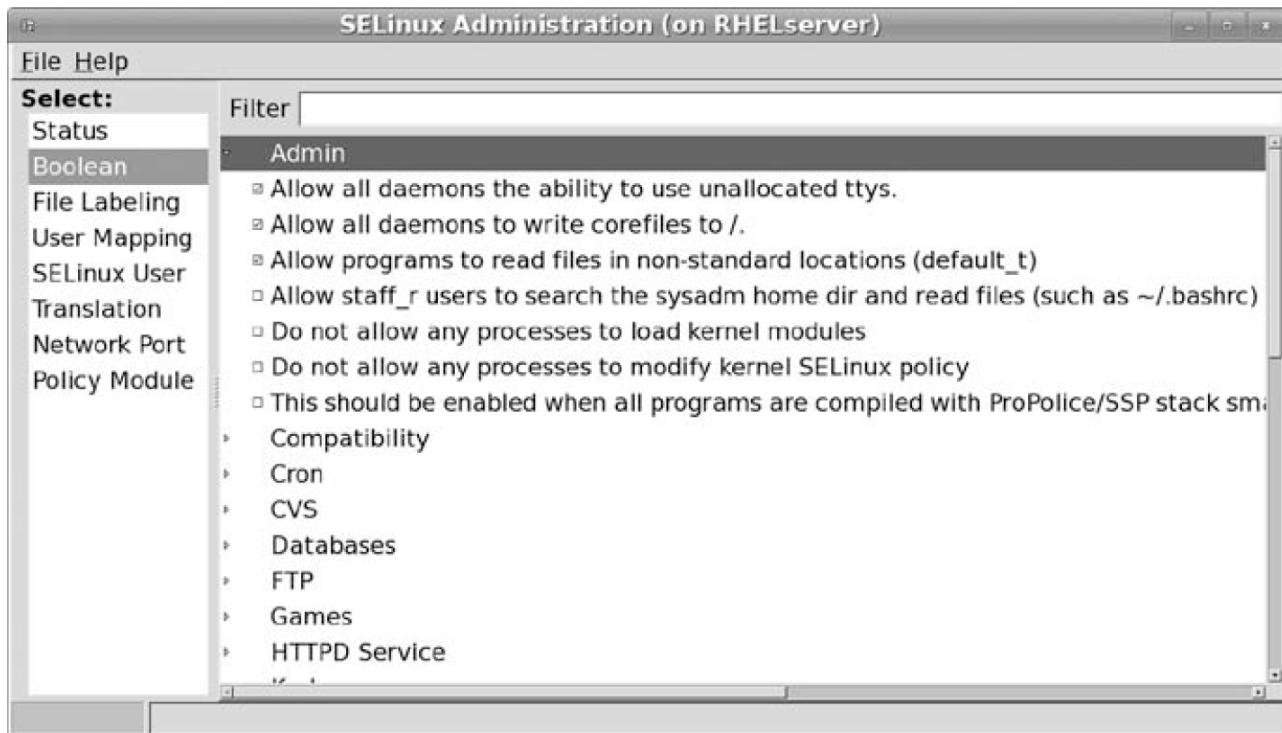
COMMAND	DESCRIPTION
<code>audit2allow</code>	Reviews cited logs and returns options that would allow actions that violate current SELinux policies. To see how it works, run the <code>audit2allow < /var/log/audit/audit.log</code> command.
<code>audit2why</code>	Translates SELinux audit messages into more understandable reasons for an SELinux violation. To see how it works, run the <code>audit2why < /var/log/audit/audit.log</code> command.
<code>avcstat</code>	Displays AVC statistics related to SELinux activity.
<code>chcat</code>	Specifies categories for SELinux security. Configured in /etc/selinux/targeted/setrans.conf. To see current categories, run the <code>chcat -L</code> command.
<code>fixfiles</code>	Relabels specified files. Do not apply to a whole system except by rebooting to avoid instability. For example, the <code>fixfiles -R packagename restore</code> command restores the original contexts of files from the noted <code>packagename</code> .
<code>genhomedircon</code>	Generates contexts from home directories of valid regular users. For example, the <code>genhomedircon</code> command by itself uses the file_contexts configuration to create a file_contexts.homedir configuration in the /etc/selinux/targeted/context/files/ directory.
<code>getenforce</code>	Returns the current mode of SELinux: <code>enforcing</code> , <code>permissive</code> , or <code>disabled</code> .
<code>getsebool</code>	Finds the value of the noted SELinux boolean. For the complete list, run the <code>getsebool -a</code> command. Values are stored in the /selinux/booleans/ directory.
<code>load_policy</code>	Loads the currently configured policy into the kernel. Activates configured changes.
<code>matchpathcon</code>	Identifies the default SELinux security context for a specified directory.
<code>open_init_pty</code>	Opens and runs a program in a pseudo-terminal.
<code>restorecon</code>	Restores the original SELinux file security contexts. When used with <code>-r</code> , the command works recursively.
<code>restorecond</code>	Specifies the daemon associated with the original SELinux contexts on a file. When run by itself, it focuses on files listed in the /etc/selinux/restorecond.conf file.
<code>run_init</code>	Runs an <code>init</code> script in SELinux contexts. Based on the /etc/selinux/targeted/context/initrc_context file.
<code>secon</code>	Reviews the SELinux contexts of a file, program, or user input.
<code>sechecker</code>	Checks SELinux policies of profiles and modules. For a current list, run the <code>sechecker -l</code> command.
<code>sediff</code>	Analyzes the difference between two SELinux text policy files.
<code>seinfo</code>	Returns statistics from SELinux policy files. For examples, look for the <code>policy.kern</code> and <code>policy.21</code> files in an /selinux/ subdirectory.

<code>sesearch</code>	Queries policies by type, including audit, type, role, and more.
<code>semanage</code>	Manages SELinux policies for logins, users, ports, interfaces, contexts, and translations.
<code>semodule</code>	Loads and manages SELinux policy modules. Lists of binary modules are stored in the /etc/selinux/targeted/modules/previous/modules/ directory.
<code>semodule_deps</code>	Shows required dependencies for different module packages.
<code>semodule_expand</code>	Expands a base module package to a binary policy file. Used internally by the SELinux management library.
<code>semodule_link</code>	Links modules together. Primarily used by developers.
<code>semodule_package</code>	Creates an SELinux policy package file.
<code>sestatus</code>	Returns the current top-level SELinux run status.
<code>setenforce</code>	Modifies the current mode of SELinux to enforcing or permissive.
<code>setsebool</code>	Sets a boolean value for an SELinux option.
<code>togglesebool</code>	Changes the value of an SELinux boolean, as listed in the /selinux/booleans/ directory.

The SELinux Administration Tool

If you’re relatively new to SELinux, the GUI-based SELinux Administration tool can be helpful. Take a look at [Figure 7-7](#). It illustrates the boolean nature of SELinux settings, along with options for administrative users. It’s also an excellent way to illustrate the configuration options available. To start this tool in a GUI, run the `system-config-selinux` command.

You’ll come back to this tool shortly as an organized way to review what can be configured with SELinux.

**FIGURE 7-7**

The SELinux Administration tool.

The SELinux Troubleshooter

In a graphical desktop environment, SELinux, when active, provides information when something violates configured SELinux rules. You could review the associated log files (`/var/log/messages` and `/var/log/audit/audit.log`), but many of these files include messages that are cryptic to the casual Linux administrator. To start the SELinux Troubleshooter, you'll need to run the following command from a GUI console:

```
# sealert -b
```

If you're just learning SELinux, the SELinux Troubleshooter is one of the few excellent reasons to use the GUI. While it's a front-end to the `audit2allow` and `audit2why` commands, it can help you isolate individual SELinux problems along with their solutions. As shown in [Figure 7-8](#), it even suggests the actions to take if you do not believe the error is a security breach.

SELinux Boolean Settings

Detailed configuration of SELinux is made possible by the boolean files in the `/selinux/booleans/` directory. As suggested by the name, the directory is full of files with just one binary number. In other words, the setting associated with the file name is either on or off (1 or 0). The `/selinux/booleans/` directory, unfortunately, is not a complete list for what's available, as default booleans aren't included in that directory. [Table 7-3](#) is based on the boolean options in the SELinux Administration tool shown in [Figure 7-7](#).

**FIGURE 7-8**

The SELinux Troubleshooter.

Be aware, [Table 7-3](#) does not list every individual boolean. It also does not explain the details behind a boolean, as those details would require another book. Because some of the detailed booleans also address features not covered in this book, they may require additional research on your part. Many of those details are not included in the glossary, as they are not by themselves central to the security features of Linux. Fortunately, most of the descriptions of individual booleans in the SELinux Administration tool are excellent.

Where the categories are associated with specific daemons, there is usually a boolean option to disable SELinux protection of that daemon. If you're having trouble getting a service to work with SELinux, you may consider activating that option temporarily until you can create the features and policies to get SELinux working with that daemon.

For more information, see the latest Red Hat Enterprise Linux Deployment Guide, available online from <http://www.redhat.com/docs/manuals/enterprise/>. The titles in the following sections correspond to the boolean subcategories in the SELinux Administration tool. These titles are subject to change and expansion as SELinux development continues.

TABLE 7-3 Boolean options in the SELinux Administration tool.

COMMAND	DESCRIPTION
Admin	SELinux administrative booleans can allow system users to use unallocated terminals, write to the top-level root directory, read files from directories not specially configured, support administrative user access to the home directories of other administrative users, prevent the loading of kernel modules, prevent processes from modifying SELinux policy, and enable buffer-overflow protection from the ProPolice protector.
Compatibility	SELinux compatibility booleans are related to administrative privileges with respect to tracing or debugging problems. You may consider keeping this ability disabled on bastion hosts, as this ability would also help a black-hat hacker develop a more dangerous Trojan on your systems.
Cron	SELinux cron daemon booleans can help regulate what cron jobs can do with respect to relabeling SELinux file contexts. It also affects the extra rules associated with the fcron service, which is sort of like the regular cron daemon with extra features.
CVS	

	If you use the concurrent versions system (CVS) for development, you may consider allowing it to read shadow passwords. On the other hand, such a capability could lead to a security breach if a malicious user were to gain access to the CVS daemon.
Databases	This section supports SELinux features associated with the MySQL and PostgreSQL database daemons.
FTP	SELinux FTP daemon booleans can help regulate features of installed FTP services. Because this is based on a Red Hat configuration, it has been tested with the Very Secure File Transfer Protocol (vsftpd) service. Different settings in this section support uploads to directories configured with the public_content_rw_t security context along with access to the Common Internet File System (CIFS) and Network File System (NFS) servers for file transfers.
Games	This section supports SELinux features associated with game services.
HTTPD service	There are nearly 20 SELinux booleans available to help regulate access to the hypertext transfer protocol daemon (HTTPD). These booleans assume you're working with the Apache Web server. These booleans are associated with several features from pluggable authentication modules (PAMs) to scripts to interactivity with CIFS, NFS, and FTP services.
Kerberos	The SELinux Kerberos booleans are simple. One allows access to Kerberos authentication from other services. It's something to consider if you want to enable the more secure features of NFS version 4. Other Kerberos booleans are associated with the Kerberos administrative and key control daemons.
Memory protection	The memory protection booleans are related to executable files. This is one area where maximum protection is appropriate, as unconfined executable files can easily lead to attacks that overload a system.
Mount	The booleans in this section relate to the automounter service. For most secure systems, the automounter is not required. Automounter features can be configured on other services.
Name service	The name service is a reference to DNS and the related NCSD daemon for cached network searches. SELinux protection for the master zone file could help prevent attacks on master DNS servers.
NFS	NFS is a reference to the network file service. SELinux boolean features can enable or prevent Kerberos-based authentication through the general security services daemon (GSSD).
NIS	NIS is a reference to the unsecure Network Information Service (NIS). Default SELinux booleans disable access to NIS.
Polyinstantiation	Polyinstantiation supports different views for different users.
Pppd	Short for the point-to-point-protocol daemon, pppd supports communication primarily over telephone modems.
Printing	Printer-related SELinux booleans are focused on CUPS. Options allow you to disable SELinux protection for various CUPS-related daemons. Alternatively, it allows you to substitute the Line Print Daemon (LPD) for CUPS.
rsync	The rsync service is frequently used for backups, as it can synchronize the data on two systems. The power of rsync comes from how it only transfers changed data over a network connection. Data transferred over rsync can be encrypted with the help of the SSH service.
Samba	Samba is the Linux implementation of Microsoft file and printer sharing. SELinux booleans in this area allow you to enable major Samba features. These features include the ability to act as a

	domain controller, read and write to shared directories, use NFS directories, share user home directories, and more.
SASL Authentication Server	The Simple Authentication and Security Layer (SASL) server is another method to support network authentication through the files of the shadow password suite. SELinux disables such access by default.
SELinux service protection	The SELinux service protection category allows you to disable SELinux protection for a wide variety of applications and services, from the Amanda backup service to Xen for virtual machines.
SpamAssassin	If you want SpamAssassin software to have access to the latest databases of unwanted e-mail, you'll want to allow the associated daemon to have network access.
Squid	Squid is the standard Linux Web proxy server. As with SpamAssassin, you'll want to allow the associated daemon to have network access.
SSH	SELinux policies related to SSH can enable administrative and host-based access.
Universal SSL tunnel	The Secure Sockets Layer (SSL) is one way to create a secure tunnel for communication over a public network such as the Internet. While it's frequently run as part of the extended internet super server, an SELinux boolean allows it to be run more securely as a separate daemon.
User privs	Short for user privileges, the SELinux booleans in this section can support access by regular users to commands needed by some power users. These booleans support access from the <code>cdrecord</code> command to shared network directories, access to the <code>ping</code> command, and more.
Web applications	Short for the point-to-point-protocol daemon, <code>pppd</code> supports communication primarily over telephone modems.
X Server	The SELinux booleans in this category support multiple users on the same X Server. This is possible because users can log into the same X Server remotely using the X Display Manager control protocol (XDMCP).
Zebra	SELinux supports the ability of the zebra daemon to write to routing tables.

Setting Up AppArmor Profiles

AppArmor is the major open source alternative to SELinux. Both provide mandatory access control for files, directories, and users. Some administrators believe that AppArmor policies are easier to create and configure. However, the protection afforded by AppArmor depends on the policies you create for the services and critical files on the local system.

If you've installed AppArmor and SELinux packages on the same system, be aware that these mandatory access control systems are not compatible. Because AppArmor drivers are integrated into current Linux kernels, you can disable AppArmor with the following entry on the kernel command line in the bootloader:

```
apparmor=0
```

Basic AppArmor Configuration

AppArmor has four basic modes: `enforce`, `complain`, `audit`, and `unconfined`. These modes are roughly parallel to the SELinux `enforcing`, `permissive`, and `disabled` modes. In other words, if you're just learning AppArmor, you could set up `complain` or `audit` mode and learn how AppArmor works from the log messages. In fact, AppArmor normally uses the same log files as SELinux: messages and audit/audit.log in the `/var/log/` directory.

Because AppArmor is the primary mandatory access control option on the Ubuntu distribution, the configuration described in this chapter reflects what is seen on that distribution. Nevertheless, the primary developers for AppArmor are sponsored by Novell, the company behind SUSE Linux. Because Ubuntu does not currently have a GUI tool to administer AppArmor, you'll take a brief look at SUSE's AppArmor administrative tool, which is part of its yet another setup tool (YaST) package.

For now, run the **apparmor_status** command from an administrative account. This command lists the configured profiles of various commands and services. If the AppArmor module is loaded, you'll see that confirmed in the output. You'll also see a list of loaded AppArmor profiles. This list will include the full path to executable files and processes with AppArmor profiles. These profiles are further subdivided by those set to **enforce** and **complain** modes.

AppArmor Configuration Files

The standard configuration files for AppArmor are stored in the `/etc/apparmor/` directory. If you don't see several configuration files in this directory, you'll need to install the `apparmor`, `apparmor-utils`, and `apparmor-notify` packages. AppArmor profiles for specific commands and services are stored in the `/etc/apparmor.d/` directory. Those profiles will be described shortly.

Despite the name, the primary AppArmor configuration file is `logprof.conf`, in the `/etc/apparmor/` directory.

`logprof.conf`

The `logprof.conf` file includes a number of basic AppArmor settings. Not all of them are active. For example, the file on the Ubuntu 10.04 release refers to a release that's a more than a year older (Intrepid Ibex). But with that in mind, a couple of the stanzas in this file include valuable information.

The [settings] stanza. Directives in the `[settings]` stanza include the directories with AppArmor configuration files. If you're looking for additional preconfigured AppArmor profiles, it points to the `/usr/share/doc/apparmor-profiles/extras/` directory. It configures logs in standard locations: `/var/log/messages`, `/var/log/syslog`, and files in the `/var/log/audit/` directory.

The [qualifiers] stanza. Directives in the `[qualifiers]` stanza include shells, which should not have AppArmor profiles. This stanza includes basic commands such as `mount` that don't work if confined with AppArmor rules. It lists basic shell commands that should not be profiled, such as `awk`, `ls`, and `chmod`.

`notify.conf`

The standard directive in this file limits access to the `aa-notify` command to members of the admin group. While the configuration file specifies `apparmor-notify`, the actual command is `aa-notify`, which is linked to the `apparmor_notify` command.

`severity.db`

The `severity.db` file includes a substantial number of key files, with severity levels between 0 and 10. For example, look at this excerpt for severity levels related to files in the `/etc/init.d/` directory:

```
/etc/init.d/* 1 10 0
```

The numbers relate to read, write, and execute access. As the scripts in this directory are well known, it's not a big deal if they're read or executed. However, if someone overwrites one of these scripts, you as a security administrator need to know about it.

`subdomain.conf`

The code in the `subdomain.conf` file is information only for now. However, it provides insight into possible future directions for AppArmor.

AppArmor Profiles

Most AppArmor profiles can be found in the /etc/apparmor.d/ directory. The default profiles in this directory cover a number of standard services and servers, including the Apache Web server, the `ping` command, the Samba file server, and more.

Take a look at the `sbin.dhclient3` file. It's the AppArmor profile for the `/sbin/dhclient3` command. Most AppArmor profiles have `include` directives to incorporate the contents of other files. For example, the following directive, if it were active (the `#` is a comment character), would include the contents of the `abstractions/base` configuration profile:

```
#include <abstractions/base>
```

The `capability` options that follow relate to the functionality of the command. Take a look at the following four `capability` options:

```
capability net_bind_service,  
capability net_raw,  
capability sys_module,  
capability dac_override,
```

These options associate the `sbin.dhclient3` profile with the noted capabilities: to acquire DNS information from a Berkeley Internet Name Domain (BIND) server; to access raw network packets; to work with system modules; and to override discretionary access controls.

The following options relate to network information, to be expected in packets in raw form. Most of the files that follow are associated with different access modes.

AppArmor Access Modes

Access modes are features associated not only with the executable file, but with related configuration, logging, lock files, and whatever else may be related to the command profile itself. The access modes that you'll see in a configuration profile are described in [Table 7-4](#). Be aware: You can configure only one mode from `ux`, `Ux`, `px`, `Px`, and `ix` on a single executable file.

Sample AppArmor Profiles

AppArmor is a work in progress. You may be interested in profiles beyond the standards shown in the /etc/apparmor.d/ directory. An extensive list is available in the /usr/share/doc/apparmor-profiles/extras/ directory. If you want to install one of the services associated with these profiles, you may choose to copy one of these extra profiles into the /etc/apparmor.d/ directory. Once copied, you can activate the profile with the `/etc/init.d/apparmor reload` command.

AppArmor Configuration and Management Commands

There are several AppArmor commands. The `apparmor_status` command, as described earlier, lists configured profiles of commands and services, along with their `enforce` or `complain` modes. Additional AppArmor commands start with `aa-` and reside mostly in the /usr/sbin/ directory. The purpose of each command is described in [Table 7-5](#).

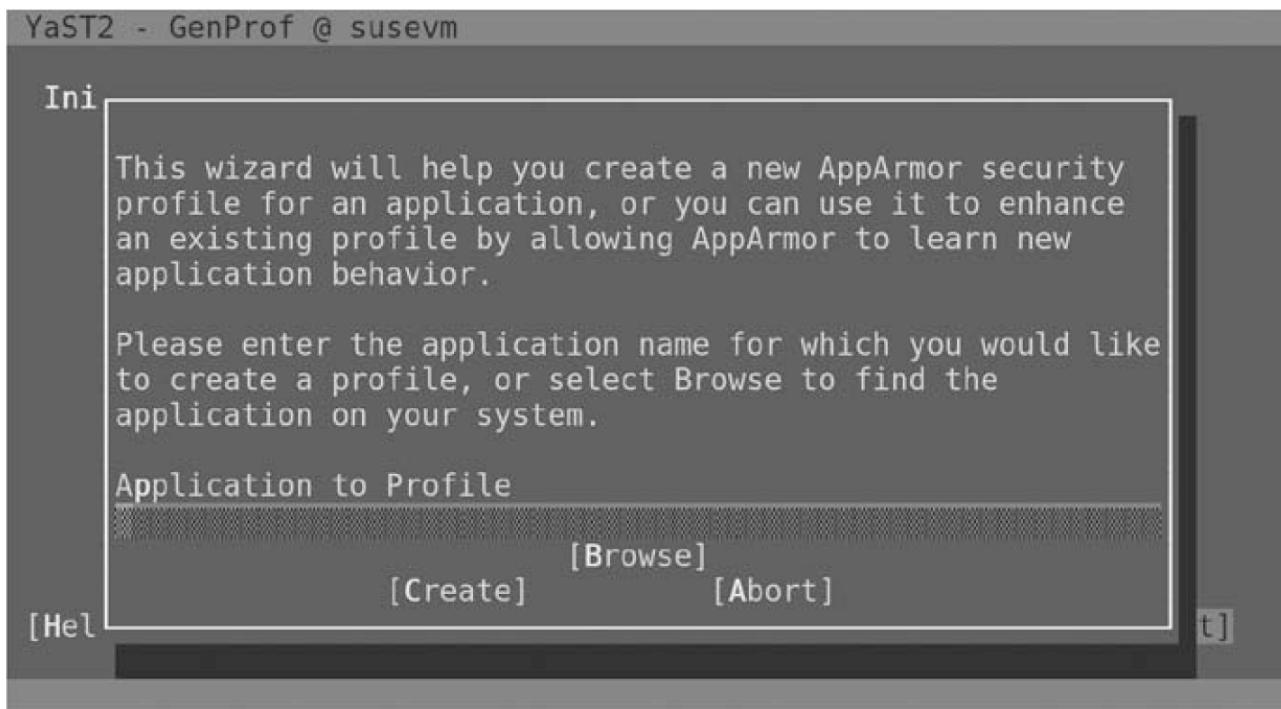
TABLE 7-4 AppArmor access modes.

MODE DESCRIPTION
r Read
w Write (cannot use with append)

a	Append (cannot use with write)
ux	Execute, in unconfined mode
Ux	Execute, in unconfined mode; supports unconfined child processes
px	Profile execute mode; works only if related files/commands also have AppArmor profiles
Px	Profile execute mode; works only if related files/commands also have AppArmor profiles, supports unconfined child processes
ix	Inherit execute mode; inherits the current profile
m	Maps to an executable file
l	Allows links to other files
k	Supports file locks

TABLE 7-5 AppArmor commands.

COMMAND	DESCRIPTION
aa-audit	Requires an existing security profile for a command or service. If that profile exists, the aa-audit command changes the profile to audit mode.
aa-autodep	Creates a sample security profile for a command or service.
aa-complain	Requires an existing security profile for a command or service. If that profile exists, the aa-complain command changes the profile to complain mode.
aa-enforce	Requires an existing security profile for a command or service. If that profile exists, the aa-enforce command changes the profile to enforce mode.
aa-genprof	Reviews the /etc/apparmor.d/ directory for an existing profile for a command or service. If the profile does not exist, it's created using the aa-autodep command.
aa-logprof	Reviews existing logs for suggested profile updates.
aa-notify	Lists denial log messages.
aa-status	Reviews current profiles and processes.
aa-unconfined	Lists networked processes not controlled by AppArmor.

**FIGURE 7-9**

An AppArmor configuration tool.

An AppArmor Configuration Tool

If you feel the need for a graphical tool to configure AppArmor, it's available on SUSE Linux. While it's not one of the distributions covered in this book, SUSE Linux is an excellent distribution with enterprise-quality support, courtesy of Novell. [Figure 7-9](#) illustrates a step in SUSE's AppArmor configuration wizard, available through its YaST package. Those of you familiar with SUSE will recognize the console version of the tool, which is a low-level graphics version of that can be used remotely without a GUI.

Best Practices: Networks, Firewalls, and TCP/IP Communications

When securing a system, you need to know about various TCP/IP ports and how services communicate over these ports, as defined in `/etc/services`.

Obscurity may be one option to help secure a network. If you configure an obscure nonstandard port for a service, you need to make sure that both clients and servers work with that port. Furthermore, you need to make sure any firewalls allow communication through the nonstandard port. Obscurity works in other ways. In login messages, you can throw off an attacker with the right kind of message. In error messages, you can configure some services to provide as little information as possible.

While the TCP Wrapper system was originally designed to protect services associated with the internet super servers, it can also protect services statically and dynamically linked to the `libwrap.so.0` libraries. Access to such services can be regulated with options in the `/etc/hosts.allow` and `/etc/hosts.deny` files. Access can be regulated by host, by IP address, and even by username.

The Linux packet-filtering firewall command is `iptables`. Different `iptables` rules can be created to protect a DMZ, to forward communications to bastion hosts, and to protect an internal network. The `iptables` command can also be used to masquerade IP addresses, supporting communications between different IP address networks. While the default rules associated with a bastion host are simple, you should do more to block communication from suspicious IP addresses, slow or prevent potential attacks, and more.

When considering network security, don't forget the alternate attack vectors. Many networks still support connections through telephone modems and serial ports. Some attackers can even find what's on your screen by tracking electromagnetic emissions. While these services aren't directly related to networks, don't forget to protect administrative services such as cron and at.

Wireless communications are another problem. Even if corporate policies prohibit wireless communication, a lot of new hardware comes with vulnerable active wireless network devices. In any case, a prohibition on wireless communications is not realistic in the current world. You should have some sort of organizational policy. When addressing wireless security, don't forget the Bluetooth connection.

Security beyond the network includes mandatory access control systems such as SELinux and AppArmor. SELinux was developed by the NSA and integrated into Red Hat Enterprise Linux to help secure that distribution. Because it is open source, SELinux can be installed on other distributions. SELinux enables access control of a wide variety of services. It's primarily configured in boolean settings in the /selinux/BOOLEANS/ directory. If you're just learning SELinux, examine the SELinux Administration tool to see what can be done with this system. In addition, the SELinux Troubleshooter may suggest solutions when you run into errors.

For mandatory access control systems, AppArmor is the major open source alternative to SELinux. Standard AppArmor profiles are stored in the /etc/apparmor.d/ directory. AppArmor profiles include a variety of access modes for the service or command in question. Such profiles also regulate how related commands and services are used.



CHAPTER SUMMARY

Network security starts with an understanding of available TCP/IP ports. Obscurity in TCP/IP communication may discourage some malicious users. Access to services controlled by the internet super servers along with those services related to TCP Wrapper libraries may be regulated by the /etc/hosts.allow and /etc/hosts.deny files. Access through various TCP/IP ports can be regulated by iptables as a packet filtering firewall.

When planning network security, it is important to consider all means of communications with a network, including telephone modems, wireless devices, Bluetooth adapters, and more. When working with security in general, consider implementing a mandatory access control system. The two major Linux options in this area are SELinux and AppArmor. Both systems can help confine compromised parts of a system. The mandatory access control can help prevent one compromised system from affecting another system.



KEY CONCEPTS AND TERMS

Access vectors

Access vector cache (AVC)

apparmor_status

Application Armor (AppArmor)

at

Challenge-Handshake Authentication Protocol (CHAP)

chcon

Denial of service (DoS) attack

Firewalld

Integrated Services Digital Network (ISDN)

Internet Assigned Numbers Authority (IANA)

Multicast Domain Name Service (mDNS) protocol

nmap

Obscurity

Password Authentication Protocol (PAP)

Public switched telephone network (PSTN)

Remote Authentication Dial In User Service (RADIUS)

secon

Security Enhanced Linux (SELinux)

sestatus

Telephone modem

Van Eck phreaking

Wireless intrusion detection system (WIDS)

Wireless receiver-transmitter (WRT) software



CHAPTER 7 ASSESSMENT

1. Well-known TCP/IP ports range from _____ to _____.
2. The nmap command checks for open ports on a remote system.
 - A. True
 - B. False
3. Which of the following configuration files is considered first with respect to TCP Wrapper security?
 - A. /etc/inetd.conf
 - B. /etc/xinetd.conf
 - C. /etc/hosts.allow
 - D. /etc/hosts.deny
4. Which of the following library files is associated with TCP Wrapper?
 - A. /etc/libwrap.so.0
 - B. /lib/libwrap.so.0
 - C. /usr/lib/libwrap.so.0
 - D. /var/lib/libwrap.so.0
5. Which of the following **iptables** command switches adds a rule to the middle of a chain?
 - A. -A
 - B. -I
 - C. -L
 - D. -C
6. Which of the following actions is not used with the -j switch for the **iptables** command?
 - A. DROP
 - B. REJECT
 - C. LOG
 - D. FORWARD
7. The **iptables** command switch associated with a destination port is _____.
8. The PSDN network is associated with regular telephone modems.
 - A. True
 - B. False
9. Which of these files must exist for regular users to access the at daemon?
 - A. /etc/at

- B. /etc/at.deny
 - C. /etc/at.conf
 - D. /etc/at/at.deny
10. Which of the following commands lists the SELinux characteristics of a file?
- A. *ls filename*
 - B. *ls -SE filename*
 - C. *ls -l filename*
 - D. *ls -Z filename*
11. Which of the following commands can be used to customize the SELinux characteristics of a file?
- A. **fixfiles**
 - B. **chcon**
 - C. **restorecon**
 - D. **secon**
12. To start the SELinux Troubleshooter in a GUI, run the following command: _____.
13. Which of the following directories include active AppArmor profiles?
- A. /etc/apparmor/
 - B. /etc/apparmor.d/
 - C. /usr/share/doc/apparmor-profiles/extras/
 - D. /usr/share/doc/apparmor-profiles/

CHAPTER

8 Networked Filesystems and Remote Access

T

HIS CHAPTER IS FOCUSED ON FILE-SHARING SYSTEMS. Because file-sharing servers are typically administered remotely, you also need to understand the ins and outs of remote access systems.

The Network File System (NFS) service can be mounted as if it were a local system. The latest version of NFS can be secured with more than just host-level restrictions. Careful configuration can help keep the Very Secure File Transfer Protocol (vsftp) server living up to its name. Samba can be used to securely manage domain-based networks. Even though Microsoft no longer supports some of the networks it has developed, Samba does.

In general, you should configure as few services as possible on any single system. In the context of this chapter, that means you might configure NFS and the Secure Shell (SSH) on system A, the vsftp and SSH on system B, Samba and SSH on system C, and so on. SSH on these systems facilitates remote administration.

Because SSH is a common element, you need to know how to configure that service securely. But there will be users who prefer Telnet. For those users, there are options. Furthermore, there are options for those users who still work with other cleartext protocols.

Chapter 8 Topics

This chapter covers the following topics and concepts:

- What basic principles for systems with shared networking services are
- How to secure NFS as if it were local
- How to keep vsftp very secure
- How Linux can be configured to act like a more secure Windows server
- How SSH can be kept secure
- What basic principles of encryption on networks are
- How to help users who must use Telnet
- How to secure modem connections
- How to move away from cleartext access
- What best practices are for networked filesystems and remote access

Chapter 8 Goals

When you complete this chapter, you will be able to:

- Secure shared networked filesystems
- Maximize the security associated with SSH
- Encrypt communications
- Provide secure alternatives to cleartext systems

Basic Principles for Systems with Shared Networking Services

Running fewer services on a system requires a reduced amount of resources. Fewer running services also means there are not as many ways for attackers or malicious users to compromise or misuse the resources on the system. Finally, fewer running services means a significantly reduced risk to you and your organization.

The most secure systems start from a minimal installation with an absolute minimum of services. If you administer services remotely, it might be wise to install and configure SSH on that baseline system. Only when you're ready to set up a system with a shared network service should that service be installed.

While it's always best to reduce the software installed on any given system, there are some software packages that are essential no matter what. One of those is **Kerberos**, which is used for authentication across multiple systems. You will also need to use a Network Time Protocol (NTP) service to make sure all of your systems are synchronized to a single time source. This is essential for Kerberos because it is very time sensitive, but it is also useful when you are looking at logs across multiple systems.

After NTP and Kerberos are set up, the next step is to set up services that work with the **generic security services application program interface (GSSAPI)**. The details depend on the service. For example, NFS requires special Kerberos configuration options; Telnet requires additional packages. Details are discussed later in this chapter.

FYI

Kerberos was developed at the Massachusetts Institute of Technology (MIT) as part of Project Athena, a distributed computing environment still in use at MIT. Other important software components that are now part of Linux began as part of Project Athena, including the X Window System and the Lightweight Directory Access Protocol (LDAP). Kerberos is named after the Greek mythological three-headed guard dog, sometimes known as Cerberus.

Configure an NTP Server

This section summarizes those steps required to get an NTP server up and running on a system. The installation process is not very difficult on most Linux distributions. Much of the work to install and configure is done automatically. For both Red Hat and Ubuntu systems, the relevant package name is `ntp`. This chapter covers setting up your system as an NTP client and using NTP servers on the Internet.

Once installed, NTP services are easy to configure. The default versions of NTP installed for both Red Hat and Ubuntu systems include a `server` directive with the URL of an associated NTP server. Examples include the following:

```
server 0.rhel.pool.ntp.org
server ntp.ubuntu.com
```

If the objective is to keep systems on a local area network (LAN) in sync with each other, it may be best to configure an NTP server on a local network. If the objective is to keep systems on remote networks in sync, it may make more sense to configure connections to the same remote NTP servers, or perhaps two NTP servers equally distant from the target networks.

Some Stratum 2 servers listed on <http://psp2.ntp.org/bin/view/Servers/WebHome> may be available for public use. The rules are created by the owners of each of these servers. Administrators of public NTP servers frequently limit access to their systems, as overloaded NTP servers are less accurate. (There are different levels of NTP servers. The lower the number, the closer you are to the atomic clock managed by the National Institute of Standards and Technology. A Stratum 2 server is a second-tier NTP server.)

Install and Configure a Kerberos Server

In this section, you'll review basic information required to install a Kerberos server. Because the actual steps, configuration files, and package names vary by distribution, this section provides only general information. Be aware that there are often multiple ways to perform the same actions with various Kerberos commands.

Kerberos goes beyond basic authentication using a concept known as a **key distribution center (KDC)** to verify the credentials of a user before allowing access to various network services. The KDC is used to distribute cryptographic keys to avoid some of the challenges associated with key exchange. When verified, the KDC issues a **ticket-granting ticket (TGT)**, a sort of time-limited super-ticket that supports access to other systems without additional authentication. The TGT is a pass that assures other servers that you have been authenticated.

The Kerberos server that grants TGTs is known as a **ticket-granting server (TGS)**, which works hand in hand with the KDC. The configuration that follows sets up both functions on a single system.

Basic Kerberos Configuration

If you use Kerberos for network authentication, it requires the installation of Kerberos software on both the client and the server. Otherwise, services such as Telnet that normally send usernames and passwords in cleartext will not be protected. This section assumes that separate systems have been set up as Kerberos servers and clients. With that in mind, once proper packages are installed on the server, you'll need to first modify the /etc/krb5.conf file.

When reviewing this file, note the subtle differences between the **Kerberos realm** and the associated uniform resource identifiers (URIs). For example, if the Kerberos realm on the local network is **EXAMPLE.ORG**, the domain is **example.org**. The Kerberos realm is typically the name of the domain for the LAN or enterprise network, in uppercase letters. It's an administrative concept that collects systems together for management and authentication purposes. Every system would belong to a specific Kerberos realm.



NOTE

The actual domains and Kerberos realms will vary. The Internet Assigned Numbers Authority (IANA) has assigned the [example.com](#), [example.net](#), and [example.org](#) domains for documentation purposes. However, these names are not to be assigned or used on the Internet. This system extends to the Kerberos realms **EXAMPLE.COM**, **EXAMPLE.NET**, and **EXAMPLE.ORG**.

Any changes made to /etc/krb5.conf should be matched with changes either to the Domain Name System (DNS) database or the /etc/hosts files for each applicable system. Critical directives in /etc/krb5.conf include the following. For the purpose of this section, the local network is configured on the [example.org](#) domain. Note the port numbers associated with the Kerberos key distribution center (**kdc**) and the administrative server (**admin_server**).

```
[libdefaults]
    default_realm = EXAMPLE.ORG
[realms]
    EXAMPLE.ORG = {
        kdc = RHELserver.example.org:88
        admin_server = RHELserver.example.org:749
        default_domain = example.org }
[domain_realm]
    .example.org = EXAMPLE.ORG
    example.org = EXAMPLE.ORG
```

After these changes are made, run the following command to create the Kerberos database. The directory with the command varies. If it is not in your PATH, look in directories such as /usr/sbin/ and /usr/kerberos/sbin/.

```
# kdb5_util create
```

You'll be prompted with information that should match the default realm, along with the master password for the Kerberos database. If you forget that database password, you'll have to delete the files in the /var/lib/krb5kdc/ or /var/kerberos/krb5kdc/ directories (depending on distribution) before re-creating the database by rerunning this command.

When the basic database is available in the appropriate directory, it's time to set up keytab files, which are pairs of Kerberos principals and clients. (A **Kerberos principal** is an identity associated with Kerberos tickets. It includes the user, the Kerberos administrator, and the realm.) Once created, Kerberos authentication between the two systems works without a password. To start the process, run the **ktutil** command, which starts a **ktutil:** prompt. Most commands that you can run at that prompt are shown in the **ktutil** man page.

At the prompt, you can enter user-account information along with encryption keys. For example, the following **addent** commands at the **ktutil** prompt adds `michael@EXAMPLE.ORG` as a Kerberos principal with the **-p** switch, a unique identity to which a Kerberos server can assign tickets. The **-k 1** specifies key version number 1. The **-e rc4-hmac** specifies an encryption scheme, created by Ron Rivest. The **rc4** stands for Ron's code, version 4. (*Ron* is a reference to Ron Rivest, one of the developers of the RSA algorithm for public key cryptography.) It's associated with the hash-based message authentication code (**hmac**).

```
ktutil: addent -password -p admin@EXAMPLE.ORG -k 1 -e rc4-hmac
ktutil: addent -password -p michael@EXAMPLE.ORG -k 1 -e rc4-hmac
```

In a similar fashion, you can set up different forms of encryption. The following commands set up key version 1 for the Advanced Encryption Standard with 256 bits (**aes256**), with message processing using ciphertext stealing.

```
ktutil: addent -password -p admin@EXAMPLE.ORG
  ↘ -k 1 -e aes256-cts
ktutil: addent -password -p michael@EXAMPLE.ORG
  ↘ -k 1 -e aes256-cts
```

The keys can then be written to a local keytab file. The following commands write it to the `michael.keytab` file in the local directory.

```
ktutil: wkt michael keytab
```

After keytab files have been created for all desired users, the next step is to merge keytab files into the /etc/krb5.keytab file. First, the following commands read keytab files for users `michael` and `donna` into local memory:

```
ktutil: read_kt michael.keytab
ktutil: read_kt donna.keytab
```

Then the following command writes those keytab files to the `krb5.keytab` file in the local directory.

```
ktutil: write_kt krb5.keytab
```

To implement it for use with Kerberos, you need to copy it to the /etc/ directory. Now that users have been set up, the next step is to set up keys to connect the local Kerberos server system with desired clients. You can create such keys with the **kadmin.local** command, which opens a **kadmin.local:** prompt. The following command sets a random key for a host with the noted fully qualified domain name (FQDN).

```
kadmin.local: addprinc -randkey host/ubuntuserver.example.org
```

Do not be concerned about any warning message. Just beware: Policies can be created as defined in the man page for the `kadmin` command. If the command is successful, you'll see the following message:

```
Principal "host/ubuntuserver.example.org@EXAMPLE.ORG" created.
```

Now you're just about ready to set up the system with the Kerberos key server to authenticate users for other systems described throughout the rest of this chapter. All you need to do is start or reload the applicable service scripts in the `/etc/init.d/` directory with names like `krb5kdc`, `kadmin`, and `krb5-admin-server`. The actual script names will vary by distribution.

Additional Kerberos Configuration Options

The previous section covered just the basic configuration options required to get Kerberos working. But to make it work for desired services, more is required in key configuration files, namely `/etc/krb5.conf`. The `[libdefaults]` stanza includes the `default_realm`, which is set to the capitalized version of the local domain name—in that case, `EXAMPLE.ORG`. Additional options of interest in that stanza include the following:

```
kdc_timesync = 1
forwardable = true
proxiable = true
```

The `kdc_timesync` option keeps the Kerberos server running if there's a temporary problem with the connection to the NTP server. The next two options can help cache and forward Kerberos keys if there's a problem with one of multiple Kerberos servers.

One more problem: Kerberos version 4 is known to have security issues. Nevertheless, older releases of Kerberos version 5 software enable access using version 4 tickets, courtesy of the `krb524init` command. That command is no longer included in the Kerberos servers released with Ubuntu Lucid Lynx and Red Hat Enterprise Linux (RHEL) 6. On older systems, to avoid this potential security issue, you should make sure the following directives are set to false:

```
krb4_convert = false
krb4_get_tickets = false
```

Securing NFS as If It Were Local

Versions of NFS prior to version 4 (NFSv4) may have been more susceptible to unauthorized users gaining access to systems. Before NFSv4, there was no user-based authentication. As such, a user with a system on the local network could connect to NFS directories shared with that network. If that NFS share included a user home directory, all the user needed to do was to set up a client system with the appropriate username and ID number on the local client system. He could then get full access to the other user's home directory files through the remote share.

That has changed with the help of Kerberos, which can at least authenticate client systems. So a malicious user with a portable system won't be able to connect to your NFS shares. Of course, this assumes that the malicious user hasn't found another way to break into a client that already has an NFS-based Kerberos ticket.

Configure NFS Kerberos Tickets

Proceed to the Kerberos server. At the `kadmin.local:` prompt, you'll create Kerberos tickets for NFS clients. (A **Kerberos ticket** is the proof on one system that verifies the identity of a second system. If coupled with

appropriate configuration options in the /etc/exports file, Kerberos tickets can be used to limit access to authorized clients.

NFS is frequently used to share the /home/ directory from a central server. Because that directory contains files for all regular users, each user needs to be able to access that directory from any connected and authorized client system. To that end, you can create a Kerberos ticket for a system using the following steps. (On some distributions, the `kadmin.local` command works in place of the `kadmin` command.)

1. Access the `kadmin`: prompt with administrative privileges with the `kadmin` command. For the Kerberos administrative key created earlier, you'd access the `kadmin`: prompt with the following command:
`# kadmin michael/admin@EXAMPLE.ORG`
2. Add a random NFS key for the client system with the following command. If your domain and realms are different, change the command accordingly. `kadmin: addprinc -randkey nfs/ubuntuh.example.org@EXAMPLE.ORG`
3. Enter the next command to add this ticket to the keytab for the local server: `kadmin: xst nfs/ubuntuh.example.org@EXAMPLE.ORG`

You'll need to repeat steps 1 and 3 on the client system. When this is complete, you can configure a shared directory on the NFS server with Kerberos options.

Configure NFS Shares for Kerberos

Shared NFS directories are configured in the /etc/exports file. For regular NFS shares, the standard configuration includes domain names or IP addresses along with mount options. For example, the following directive is a typical regular NFS share directive:

```
/backups    192.168.0.0/255.255.255.0(rw,sync)
```

This is a straightforward share of the /backups/ directory with the systems on the noted IP network address and subnet mask. The `rw` and `sync` mean that the share is read-write and reads and writes are done synchronously. Other options are shown in the man pages for the `mount` command and /etc/exports file.

Generally, NFS directories should rarely if ever be exported with the `no_root_squash` option, as that would allow root administrative access to the shared directory. That root user could even be just a local user on a system that has an IP address in the address range associated with the network address and subnet mask.

In contrast, if the packages for NFSv4 are installed, you can set up a share similar to the following, which authenticates shares to clients specifically assigned by the Kerberos server:

```
/backups    gss/krb5i(sync,rw)
```

Once configured on the NFSv4 server, users on Kerberos-authorized client systems will be able to connect to the share. Such users would either need root administrative privileges or require appropriate configuration changes to the /etc/fstab file. These changes should include the `krb5i` option.

Keeping vsftp Very Secure

FTP is not generally considered to be a good option for file transfer because of the lack of encryption. However, it's still used because there are so many legacy processes and applications that use FTP. While many FTP server options are available, vsftpd is the server used by Red Hat, SUSE, Debian, and even the developers of the Linux kernel.

To review, the standard vsftpd configuration file is `vsftpd.conf`. The directory varies by distribution. For example, it's in the /etc/ directory for Ubuntu and the /etc/vsftpd/ directory for Red Hat. For a full list of available vsftpd directives, see the Web site associated with vsftpd developers: <https://security.appspot.com/vsftpd.html>.

Configuration Options for vsftp

For many users, a vsftp server can be a terrific convenience. The FTP protocol is still perhaps the most efficient way to upload and download large files. But such a service comes with risks. To that end, you may want to regulate and even isolate the directories where vsftp stores files for anonymous uploads and downloads. The developers behind different Linux distributions configure that anonymous directory in different locations, such as /var/ftp/, /srv/ftp/, and /home/ftp/. The following directive sets up a specific directory for anonymous access to the vsftp server:

```
anon_root = /secure
```

Unless otherwise configured, vsftp is actually run as an internet super server service. It's safer to run vsftp as its own service, with its own control script in the /etc/init.d/ directory. Both Ubuntu and Red Hat versions of vsftp make this possible with the following directive:

```
listen = yes
```

However, if the vsftp server is to work on an Internet Protocol version 6 (IPv6) network, you could configure the following alternative directive. Just be aware that vsftp won't work with both versions of the **listen** directive:

```
listen_ipv6 = yes
```

Frequently, administrators like you are responsible for transmitting corporate or organizational policies regarding the use of services like vsftp. If the following directive is enabled, vsftp looks for a hidden .message file in each accessible directory unless overridden by a **message_file** directive.

```
dirmessage_enable = yes
```

Given the popularity of many anonymous FTP services, logging information about uploads and downloads may be useful. It's activated with the **xferlog_enable** directive shown next. While the default log file is /var/log/vsftpd.log, it can be changed with the **xferlog_file** directive. If you do change that file, be sure to change the corresponding file in the /etc/logrotate.d/ directory. Otherwise, the size of this file could easily overwhelm many systems.

```
xferlog_enable = yes
```

One option for uploads is to change the ownership of such files to a specific user, such as nobody or another user configured with minimal privileges. That can reduce the risk of an uploaded script or binary file that otherwise affects the security of the FTP server. Of course, this assumes user nobody is properly configured in the shadow password suite (or other user database) with appropriate minimal privileges.

```
chown_uploads = yes
```

```
chown_username = nobody
```

If you want to specifically set a **nonprivileged user**, the following directive can help. (A nonprivileged user is an account with standard end-user operating system permissions. This type of user does not have administrative permissions that would be found with a superuser, root, or administrative account.)

```
nopriv_user = nobody
```

A couple of additional options can work sort of like screensavers. The following options end the connection if no commands or additional file transfer bits are detected in the given number of seconds. Some administrators may prefer to reduce the value of the **idle_session_timeout** directive.

```
idle_session_timeout = 600
```

```
data_connection_timeout = 120
```

Generally, secure FTP servers do not allow connections based on the American standard code for information interchange (ASCII), as that can lead to denial of service (DoS) attacks.

Additional vsftpd Configuration Files

Two other configuration files are directly used for the vsftpd service: `ftpusers` and `user_list`. The list of users in these files is based on the default set of service users. In other words, it includes users with user IDs (UIDs) below 100, 500, or 1000, depending on the distribution. If a malicious user compromises one of these accounts, he or she may be able to obtain limited administrative privileges (or more) through them. These files can help prevent malicious users from using these privileges at least on the vsftpd server.

If more services are installed on the local system, you may see additional service users in the files of the shadow password suite. In that case, you should add those users to the appropriate files. Red Hat systems specifically cite `/etc/vsftpd/ftpusers` in the associated pluggable authentication module (PAM) configuration file, `/etc/pam.d/vsftpd`.



TIP

Security Enhanced Linux (SELinux) provides additional protection for vsftpd. Directories shared using FTP servers should be labeled with the `public_content_t` type. If you want to configure a directory for uploads, you'll need to configure that directory with the `public_content_rw_t` type. The same options work for directories associated with rsync servers.

Linux as a More Secure Windows Server

This section focuses on the global settings associated with Samba, including how it can be set up as a **Primary Domain Controller (PDC)**, or its cousin, the **Backup Domain Controller (BDC)**. (A PDC is a master server on a Microsoft Windows NT domain that controls and can grant access to a number of computer resources based on the usernames and passwords in its database. A BDC is a backup for a PDC on a Microsoft Windows NT domain.) With the right options, the PDC database of usernames and passwords can be maintained on a Linux server.

Samba is the Linux implementation of Microsoft's Server Message Block protocol. As this evolved into the Common Internet File System (CIFS), Samba has kept pace. Because case generally does not matter at the Microsoft command line, the value given to most Samba directives can be uppercase or lowercase. For example, `security = USER` works as well as `security = user`.



NOTE

Because PDCs are based on the Microsoft Windows NT 4 Server operating system, Microsoft hasn't supported PDCs on networks in years. However, PDCs on Linux servers configured with the Samba file server are still supported. As long as Windows XP clients are supported, Microsoft should support those systems as PDC clients. However, because this is a book on Linux security, no claims are made with respect to Microsoft security support.

Administrators who believe in Linux are likely to accept the notion that a Linux system, when substituted for a Windows system, is more secure. Some security professionals may disagree with that assessment. However, Linux systems maintained as PDCs are currently more secure, as Samba developers monitor and maintain such services while Windows developers no longer do so.

Samba Global Options

This section focuses on directives in the main Samba configuration file, `smb.conf`, in the `/etc/samba/` directory. This section focuses primarily on the directives in the `[global]` section of this file. When reading this file, be aware there are two different comment characters. Any line that starts with a pound sign (#) or a semicolon (;) is a comment in this file.

Any Linux system configured with Samba can be set up as a server on a Microsoft network. While this section is based on the Samba configuration file from Red Hat Enterprise Linux, the same directives can be used on any Linux system where Samba is installed. The following subsections are based on the sections from the sample version of that file.

One thing that may appear different about many Samba directives is that they're multiple words. Don't be fooled. A multiple-word directive such as `server string` and `add user script` is a single directive.



NOTE

Samba directives include many synonyms. For example, `hosts allow` is the same as `allow hosts`. Samba directives also accommodate common alternative spellings, such as `browsable` and `browseable`. If you see one directive in documentation and a similar directive in a configuration file, look a little deeper. The directives might actually mean the same thing.

Network-Related Options

Microsoft networks began with workgroups of computers where directories were shared solely based on passwords. As such, Samba used the `workgroup` directive to specify the name of the network of the noted group of computers. When the latest Microsoft networks evolved into domains, Samba retained the same `workgroup` directive to specify the name of the domain. So depending on the type of network being configured, the value assigned to `workgroup` can be the name of either a workgroup or a domain.

As shown here, the `workgroup` directive is normally coupled with the `server string` and `netbios name` directives. The example shown returns **Samba Server Version** followed by the version number of the Samba server, as specified by the `%v`. It's shown as a comment in a Microsoft My Network Places window or in the output of a Microsoft `net view` or a Samba `smbclient -L` command.

```
workgroup = bigdomain
server string = Samba Server Version %v
netbios name = trivialinfo
```

If obscurity is part of your security strategy, consider changing the values of `server string` and `netbios name`. The values shown there can throw off the casual intruder. The `netbios name` directive is associated with the **Network Basic Input/Output System (NetBIOS)** name of a system. A NetBIOS name is commonly assigned on Microsoft-style networks. It is associated with the Session Layer of the **Open Systems Interconnection (OSI) Reference Model** of networking. (The OSI model is similar to the TCP/IP protocol suite, but with seven layers instead of four.) If the `netbios name` value is not assigned, the system hostname is used.

The `interfaces` and `hosts allow` directives can limit access to the Samba server in two ways. The examples shown here limit the interfaces and addresses to which Samba listens. The `hosts allow` directive can be revised to limit access to a domain or even individual hostnames.

```
interfaces = lo eth0 192.168.12.2/24 192.168.13.2/24
hosts allow = 127. 192.168.12. 192.168.13.
```

Logging Options

It may be helpful to set up logs by different machines. The following directives would set up log files by NetBIOS name or IP address. The `max log size` directive sets a limit on log files, in kilobytes.

```
log file = /var/log/samba/%m.log
max log size = 50
```

Standalone Server Options

This section is closely related to the next section. Some of the options and directives are mutually exclusive. For example, while you should activate only one `security` directive in this configuration file, examples of `security` directives are listed in both sections. The two `security` options described in this section are `user` and `share`.

Security-conscious professionals should almost never set `security = share`, as that allows access to shared directories for users without accounts. All they need is the password for the shared directory. So for a standalone server, a PDC, or a BDC, you want to set up the following directives:

```
security = user
passdb backend = smbpasswd
```

The `smbpasswd` and `tdbsam` options allow administrators to set up or change user passwords with the `smbpasswd` command. Be aware: The Samba user password database is separate from the Linux password database, and is stored in files in the `/etc/samba/` directory. In the configuration shown, it's stored in the `smbpasswd` file. If set to `tdbsam`, user-account information is stored in the `passdb.tdb` file.

Alternatively, if you want to use an LDAP database, set `passdb backend = ldapsam`. You may consider including a number of other related directives shown in [Table 8-1](#). The table lists just some of the most important LDAP directives. For example, options associated with logging are not included in the list and may be found in the Samba documentation described earlier. Some of the listed directives may be used to map Linux UIDs and group IDs (GIDs) to other authentication databases.

If the local system is configured as the one Microsoft server on the local network, the authentication database will be stored on the local system. Depending on how `smb.conf` is configured, that will be in files either in the `/etc/samba/` or the `/etc/samba/private/` directory.

TABLE 8-1 LDAP-related Samba directives.

LDAP DIRECTIVE	DESCRIPTION
<code>client ldap sasl wrapping</code>	This allows LDAP traffic to be encrypted using the simple authentication and security layer (SASL). It's associated with the following Microsoft registry key: <code>HKLM\System\CurrentControlSet\Services\NTDS\Parameters\LDAPServerIntegrity</code>
<code>idmap backend</code>	This supports a database association between Microsoft security identifiers (SIDs) and Linux UIDs and GIDs.
<code>idmap gid</code>	This specifies a range of GIDs to be mapped to Microsoft SIDs. It's not limited to LDAP authentication databases.
<code>idmap uid</code>	This specifies a range of UIDs to be mapped to Microsoft SIDs. It's not limited to LDAP authentication databases.
<code>ldap admin dn</code>	This specifies the distinguished name to contact the LDAP server.
<code>ldap group suffix</code>	This sets the suffix to be used for LDAP groups, such as <code>ou=Groups</code> .

ldap machine suffix	This defines the suffix to be used for LDAP systems, such as ou=Computers .
ldap password sync	This defines whether the Microsoft LDAP database is to be updated when a Samba password is updated.
ldap suffix	This sets the base entry for the object, such as dc=example, dc=org .
ldap user suffix	This sets the suffix to be used for LDAP users, such as ou=People .

FYI

One trivial bit is the terms associated with user authentication on Linux and Microsoft systems. Users on Linux systems log in to a computer. Users on Microsoft systems log on to a computer. Because Samba is the Linux implementation of Microsoft networking, the Samba configuration file configures Microsoft-style user logons. Thus, the comments in the Samba server configuration file specify “login” and the associated directives use **logon**.

If you are configuring a PDC, pay attention to the [netlogon] and [profiles] stanzas in the “Shares Definitions” section of the **smb.conf** configuration file. These stanzas determine how logon information is shared by user.

Domain Members Options

This section includes many of the same directives as the “Standalone Server Options” section. In this [global] part of the file, directives should be used only once. Three options are noted in this section for the **security** directive:

- **domain**—This assumes the local system has been added to an NT domain. It refers authentication requests to another server.
- **server**—This works if the local system has not been added to an NT domain, if a local **smbpasswd** database is available. It is vulnerable to man-in-the-middle attacks.
- **ads**—This sets the local system as a domain member in an Active Directory domain.

Because these options require access to some other system for authentication, they won’t work without the **password server** directive to point to the FQDN or IP address of that server. If you’ve set up a Kerberos server with an Active Directory, use the **realm** directive to specify the name of the Kerberos realm.

Domain Controller Options

Local Samba servers can be configured to send and receive login information from Microsoft systems. To that end a Samba-enabled Linux system can be configured as if it were a Microsoft client. Login scripts can be configured for machines and users with the **logon script** directive:

```
logon script = %m.bat
logon script = %u.bat
```

Profiles can be enabled with the **logon path** directive. For example, the following directive specifies a shared profile/ directory on some server named **authentication**. The %U means user profile information is stored in a subdirectory named for the user.

```
logon path = \\authentication\profile\%U
```

Browser Control Options

On a Microsoft network, the browser collects the domain or workgroup name of every system on the network. But only one system can be a browser at a time on a LAN. The browser control options include directives that drive which system is assigned as the **master browser** on a Microsoft network. (A master browser is a system assigned to maintain a database of NetBIOS names and their services, such as domain or workgroup membership.) The terms assume that different systems put themselves up for election as the master browser. The relevant directives are as follows:

- **domain master**—This determines whether the local system forces an election for the domain.
- **local master**—This determines whether the system puts itself up for election.
- **os level**—This sets the advertising level for browsing elections. It may be set between 0 and 255.
- **preferred master**—This determines whether the local system forces an election for the local network.

Name Resolution

Name resolution on Microsoft networks is based on the status of **Windows Internet Name Service (WINS) servers** and DNS servers. A WINS server contains a database of NetBIOS names and IP addresses. If the NetBIOS name does not exist, the WINS server defaults to the hostname of the system. The basic directives in this section are straightforward:

- **wins support**—This enables a WINS server on the local system.
- **wins server**—This configures a pointer to a running WINS server.
- **wins proxy**—This redirects WINS requests to a system specified by **wins server**.
- **dns proxy**—This supports NetBIOS name resolution via DNS.

Printing Options

By default, a Samba server shares printers configured using the Common Unix Printing System (CUPS) over the Windows network. The following options load CUPS printers and configure raw data for processing by Microsoft printer drivers:

```
load printers = yes
cups options = raw
```

Samba as a Primary Domain Controller

If you want to configure a Windows NT-style PDC, pay attention to the **security** directive described earlier. It works with the [netlogon] and [profiles] stanzas in the “Share Definitions” section of the smb.conf configuration file. The [netlogon] stanza is for storage of user-specific Microsoft logon scripts. The **path** directive determines the directory where those scripts are stored. It’s associated with the username, based on the **logon script** directive described earlier.

```
path = /var/lib/samba/netlogon
```

The **guest ok** access can be set to **no** to prevent users from changing others’ scripts. The **writable** and **share modes** options shown also limit access to the administrative user.

```
guest ok = no
writable = no
share modes = no
```

The [profiles] stanza is used to allow users to log on from different workstations. The profiles in the noted **path** directory allow users with Microsoft clients to have the same look and feel on each workstation. Profiles may be stored in username-specific subdirectories, as defined by the **path**.

```
path = /home/samba/profiles/%U
```

The next three directives allow users to change profiles and write those changes. To set up standard profiles, you may choose to set `read only = yes`.

```
read only = no
create mask = 0600
directory mask = 0700
```

The last two directives limit access to authorized users:

```
guest ok = no
browseable = no
```

To generate a Samba configuration for a server that will act as an Active Directory server, you can run the `samba-tool`. It will step you through all of the settings by asking questions. When you are finished, you will have an `smb.conf` file that will have Samba acting as a domain controller on your network. This does require that you have Kerberos installed and configured, of course. The `samba-tool domain provision` command will create the configuration file for you.

Making Sure SSH Stays Secure

In this section, you'll see how to best secure SSH clients and servers. One of the benefits of SSH is encryption. However, until an SSH connection is made, transmissions are still sent in cleartext. So a malicious user may be able to determine that an SSH connection is being established and eavesdrop on any communications that take place before the user initiates the SSH connection.

In addition to using SSH to configure interactive username/password connections, you may also automate the SSH connection process through the use of public/private key pairs.

The Secure Shell Server

The standard SSH server configuration file is `sshd_config`, in the `/etc/ssh/` directory. It works very well on its own. On most distributions, all you need to do is install the server with a package name like `openssh-server` and make sure it's running. You should be able to connect from a remote system with commands described in the next section on the SSH client.

It's important to modify the default SSH server configuration file to promote security, however. Directives listed here are based on the default versions of `sshd_config` installed for Red Hat and Ubuntu systems. First, the standard SSH port number is 22, as confirmed here:

Port 22

Because SSH version 1 has known security weaknesses, most administrators encourage the use of SSH version 2. To ensure that no one attempts to exploit a vulnerability with SSH version 1, you will need to be explicit about using only version 2 of the protocol. Therefore, most `sshd_config` files limit access to SSH version 2 with the following directive:

Protocol 2

Assuming you accept the use of SSH version 2, other directives associated with SSH version 1 are ignored (and as such will not be covered here). The `ListenAddress` directive can limit the networks configured for SSH. For example, the following directive looks for and listens to network cards with the noted IP address:

```
ListenAddress 192.168.10.1
```

The following `HostKey` directives specify Digital Signature Algorithm (DSA) and Rivest Shamir Adelman (RSA) host key files that can be used to help verify the integrity of the host server. SSH won't work unless permissions on these files are limited to the root administrative user:

```
HostKey /etc/ssh/ssh_host_dsa_key
HostKey /etc/ssh/ssh_host_rsa_key
```

However, there are companion public key files, `ssh_host_dsa_key.pub` and `ssh_host_rsa_key.pub`, that can and should be readable by all users. The next important directive, `SyslogFacility`, determines where logging information is stored. It's normally set to `AUTH` or `AUTHPRIV`. The log file for `AUTH` or `AUTHPRIV` messages is normally defined in the `/etc/syslog.conf` file.

The authentication directives that follow are mostly self-explanatory. `LoginGraceTime` disconnects if a login hasn't happened in the noted time. While the default is `yes`, `PermitRootLogin` should almost always be set to `no` to minimize the risk of a malicious user decrypting an administrative password sent over a network. `StrictModes` keeps users from setting world-writable files over an SSH connection. `MaxAuthTries` limits the number of login attempts per connection.

```
LoginGraceTime 2m
PermitRootLogin no
StrictModes yes
MaxAuthTries 6
```

The additional authentication options shown here support the use of private/public key authentication with passphrases as described later in this chapter. `AuthorizedKeysFile` specifies the location of authorized SSH keys in each user's home directory:

```
PubkeyAuthentication yes
AuthorizedKeysFile .ssh/authorized_keys
```

Generally, you want to retain user-based authentication. To that end, the following host-based authentication option is disabled by default and should remain that way:

```
HostbasedAuthentication no
IgnoreRhosts yes
```

However, you need to retain the following default option to avoid malicious users who substitute their systems for known systems:

```
IgnoreUserKnownHosts no
```

After you've set up passphrases, you can change this to `no` to disable cleartext tunneled passwords:

```
PasswordAuthentication yes
```

Of course, when passwords are used, they should not be empty:

```
PermitEmptyPasswords no
```

If you need SSH connections over a telephone modem, you may consider activating this option. However, there are interaction problems with PAMs.

```
ChallengeResponseAuthentication no
```

If you've set up a Kerberos server, you can change some of these default Kerberos options for Kerberos or regular passwords. Methods to do so include Kerberos tickets or **Andrew Filesystem (AFS)** tokens. (AFS is a distributed network filesystem.)

```
KerberosAuthentication no
KerberosOrLocalPasswd yes
KerberosTicketCleanup yes
KerberosGetAFSToken no
```

Related to Kerberos tickets is the generic security services application programming interface (GSSAPI), as described earlier in this chapter:

```
GSSAPIAuthentication no
GSSAPIAuthentication yes
GSSAPICleanupCredentials yes
GSSAPICleanupCredentials yes
```

If you need GUI-based tools, the next directive can be especially valuable. **X11** is the name of a protocol associated with the Linux X Window system. It allows you to start and use GUI tools remotely:



TIP

Malicious users frequently try to break into SSH servers. One open source effort to block known malicious users is available from <http://denyhosts.sourceforge.net/>. It was developed by Phil Schwartz. SSH is a TCP Wrapper service. The DenyHosts software takes advantage of this by adding address information from repeat offenders to the local /etc/hosts.deny file.

```
X11Forwarding yes
```

If login banners are used to transmit corporate or organizational policies for remote users, you'll want to activate the following directive and enter those policies in the /etc/issue.net file:

```
Banner /etc/issue.net
```

One of the values of the SSH server configuration file is its ability to encrypt FTP connections. The following directive supports connections to user home directories with the **sftp** command:

```
Subsystem sftp /usr/libexec/openssh/sftp-server
```

The Secure Shell Client

Several SSH client commands are made available when the SSH client package is installed. These commands include **ssh** for client connections, **scp** for secure copying, and **sftp** for encrypted connections to FTP servers.

Anonymous FTP connections are not allowed with the **sftp** command.

SSH Logins

The most straightforward way to log into a remote system is with the **ssh user@hostname** command. If no username is specified, the **ssh** command assumes the current user account on the local system also applies on the remote system. You can substitute the FQDN or IP address for the hostname.

If you want to log in remotely with X11 forwarding, use **ssh's -X switch**.

Secure SSH Copying

The `scp` command can copy a file over an SSH connection. If you run a command like the following, expect to be prompted for the noted user's password or passphrase on the remote system. After those are verified, the `copy` command is processed, copying the noted localfile to the `/backups/` directory on the remote system. The `scp` command is a much better option than using FTP because `scp` is encrypted. No credentials are passed in the clear over the network.

```
scp localfile michael@remote.example.com:/backups/
```

Create a Secure Shell Passphrase

Passphrases are more secure than passwords. They use private/public key pairs with some large number of bits, typically 1,024 or more. The key pairs are associated with a passphrase. The example in this section sets up an SSH passphrase for logins from a client to a remote server. It starts with the `ssh-keygen` command, which leads to the messages that follow. Unless you're creating a key for the root administrative user (which is not recommended), these commands should be run from a regular user account:

```
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/michael/.ssh/id_rsa):
```

The noted file is the location of the RSA private key. Most users will accept the default location, at which point the next message prompts them for a passphrase. The best passphrases are complete sentences, with uppercase and lowercase characters, numbers, and punctuation. The following messages omit the actual key fingerprint. The passphrase isn't shown on the screen to minimize the risk posed by shoulder surfers. (A shoulder surfer is someone who looks over the shoulder of the person at the keyboard typing. This vantage point may allow the observer to acquire usernames and passwords.)

```
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/michael/.ssh/id_rsa.
Your public key has been saved in /home/michael/.ssh/id_rsa.pub.
The key fingerprint is:
```

By default, the encryption key is based on the RSA algorithm, which can be configured at 768 bits or more. Keys with large numbers of bits, such as 4,096, may take some time to create. While the same command can create DSA keys with the `-t dsa` switch, only 1,024-bit DSA keys are available. You should now see the noted `id_rsa` and `id_rsa.pub` files in your home directory, in the `.ssh/` subdirectory.

The next step is to copy the `id_rsa.pub` public key to the remote server. While you could use a USB key for this purpose, you can also use the `ssh-copy-id` command to transmit the public key:

```
$ ssh-copy-id -i .ssh/id_rsa.pub LucidServer.example.org
```

If this is the first time an SSH connection is being made to the noted Lucid Server system, you'll see messages similar to those shown here:

```
The authenticity of host 'lucidservermin.example.org
(192.168.100.27)' can't be established. RSA key fingerprint
is [some deleted hexadecimal number]
```

Assuming the IP address is what you expect, proceed with the connection by typing **yes** at the displayed prompt:

```
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'lucidserver.example.org'
(RSA) to the list of known hosts.
```

If you never want to transmit even an encrypted password over a network, the following message should give you pause. If you prefer, stop here and get out that USB key.

```
michael@lucidservermin.example.org's password:
```

If you proceed with a working password, the contents of the noted id_rsa.pub file are appended to the end of the .ssh/authorized_keys file on the remote system. If you don't want to transmit over the network, you could copy the id_rsa.pub file to a USB key, connect the USB key to the remote system, and run the following command on the remote system (in this case, the [lucidserver.example.org](#) system):

```
$ cat id_rsa.pub >> .ssh/authorized_keys
```

If successful, you should now be able to use the passphrase to connect to the remote system. You should be prompted with the following:

```
Enter passphrase for key '/home/michael/.ssh/id_rsa':
```



FIGURE 8-1

A request for a passphrase.

Alternatively, if X11 communication is enabled over the network, you may be prompted with a window similar to that shown in [Figure 8-1](#).

Basic Principles of Encryption on Networks

You've just read about one method of encrypted communication, using SSH. Other methods are available. One that is commonly used is **Internet Protocol Security**. IPSec is a set of extensions to IP that were developed as part

of IPv6 but can be implemented along with IPv4 to allow encryption between two hosts. It works at both ends of a connection, tunneling communications through protocols 50 and 51. In tunneling mode, it is a form of virtual private networking.

Another method is based on the Secure Sockets Layer (SSL) protocol. For most systems, SSL has been superseded by Transport Layer Security (TLS). In most cases, it uses a different port to enable encrypted communication for common services. Uses of SSL and TLS will be discussed in more detail, starting with the discussion of e-mail clients later in this chapter.

The focus of this section is IPSec connections. An IPSec connection is a method for connecting remote hosts and networks using a secure tunnel. IPSec can connect different private networks over the Internet. It relies on encapsulation to hide communications over public networks.

Different port and protocol numbers are used for IPSec connections. Assuming a firewall is configured on each network, you'll have to make sure these ports and protocols are open on the respective networks:

- **Encapsulating Security Payload (ESP) protocol**—The **Encapsulating Security Payload (ESP) protocol** uses protocol 50. ESP traffic can be allowed through an iptables firewall with a `-p 50 -j ACCEPT` switch.
- **Authentication Header (AH) protocol**—The **Authentication Header (AH) protocol** uses protocol 51. AH traffic can be allowed through an iptables firewall with a `-p 51 -j ACCEPT` switch.
- **Internet Key Exchange (IKE) protocol**—The **Internet Key Exchange (IKE) protocol** uses the User Datagram Protocol (UDP) over port 500. IKE traffic can be allowed through an iptables firewall with a `-p udp --dport 500 -j ACCEPT` switch.
- **Network Address Translation (NAT) traversal protocol**—The **Network Address Translation (NAT) traversal protocol** uses the Transmission Control Protocol (TCP) and UDP over port 4500. NAT traffic can be allowed through an iptables firewall with `-p tcp --dport 4500 -j ACCEPT` and `-p udp --dport 4500 -j ACCEPT` switches.

IPSec encryption depends on the IKE service, sometimes known as *raccoon*. The IKE service is configured in files in the /etc/raccoon/ directory. On Ubuntu systems, IPSec is configured in the /etc/ipsec-tools.conf file. Alternatively, if you use Red Hat's Network Configuration tool, you'll see appropriate configuration directives in dedicated files in the /etc/sysconfig/network-scripts/ directory.

The Red Hat configuration is described first in each of the following subsections, as it includes just those parameters that need to be configured. In either case, if the connection is successful, you should be able to run a command like `tcpdump` on both ends to detect traffic traveling with the noted port numbers.

Host-to-Host IPSec on Red Hat

On Red Hat systems, IPSec host-to-host connections are configured in two files in the /etc/sysconfig/network-scripts/ directory. One is ifcfg-host, where *host* is the name of the host-to-host device. The configuration is simple, including just four directives:

```
DST=192.168.122.127
TYPE=IPSEC
ONBOOT=yes
IKE_METHOD=PSK
```

The local host is assumed to be one end of the connection. the DST is the destination. It's an IPSec connection activated during the boot process, and the IKE method uses a pre-shared key (PSK). There should be a matching file on the remote system. The difference is in the IP address associated with the DST directive.

Both systems also require authentication keys. The file will be stored in the same /etc/sysconfig/network-scripts/ directory, with a name of *keys-host*. The file should be set with read and write permissions for just the root administrative user.

Host-to-Host IPSec on Ubuntu

On Ubuntu systems, an IPSec host-to-host connection begins in the /etc/ipsec-tools.conf file. The script shown here is based on the **setkey -f** command, on a host with IP address 192.168.0.100. This file creates a security policy database (SPD) for a host connection from IP address 192.168.0.100 to 10.0.0.100. It flushes any previous regular and SPD policies:

```
#!/usr/sbin/setkey -f
flush;
spdflush;
```

Then it adds SPD policies for traffic to and from the noted IP addresses. The **esp** and **ah** directives enable traffic over protocols 50 and 51 as described earlier in this section.

```
spdadd 192.168.0.100 10.0.0.100 any -P out ipsec
    esp/transport//require
    ah/transport//require;
spdadd 10.0.0.100 192.168.0.100 any -P in ipsec
    esp/transport//require
    ah/transport//require;
```

A matching /etc/ipsec-tools.conf file should be created on the remote system, with IP address 10.0.0.100. The IP address should be reversed in the remote version of this file. Both files should be set as read-write only by the root administrative user.

Now you'll create two files on the local system—in this case, the one with an IP address of 192.168.0.100. Both files will be in the /etc/racoon/ directory. The first file is racoon.conf. One sample customized for this connection is shown in [Figure 8-2](#).

Now you'll create a psk.txt file in the /etc/racoon/ directory. The default version includes some options. You'll need to include the IP address of the remote system:

```
10.0.0.100 0ab35s1j349gas
```

You'll want the same files on the remote 10.0.0.100 system, with the IP addresses switched in each file. Once these changes are made to both systems, you should be able to run the following command on both systems:

```
# setkey -f /etc/ipsec-tools.conf
```

```

path pre_shared_key "/etc/racoon/psk.txt";
path certificate "/etc/racoon/certs";

remote 10.0.0.100 {
    exchange_mode main,aggressive;
    proposal {
        encryption_algorithm 3des;
        hash_algorithm sha1;
        authentication_method pre_shared_key;
        dh_group modp1024;
    }
    generate_policy off;
}

sainfo anonymous {
    pfs_group 2;
    lifetime time 1 hour
    encryption_algorithm 3des;
    authentication_algorithm hmac_md5;
    compression_algorithm deflate;
}

```

FIGURE 8-2

An /etc/racoon/racoon.conf file for host-to-host IPSec connections.

If successful, you should be able to apply the `tcpdump` command on both systems. Specifically, the following `tcpdump` command should list all traffic going to the remote 10.0.0.100 system over the network device eth0. The `-n` specifies numeric addresses and the `-i` specifies the network device:

```
# tcpdump -n -i eth0 host 10.0.0.100
```



NOTE

A slightly different method for host-to-host IPSec connections is available from the Ubuntu community documentation at <https://help.ubuntu.com/community/IPSecHowTo>.

Network-to-Network IPSec on Red Hat

The basic configuration for network-to-network IPSec connections on Red Hat systems is similar to the previously described IPSec host-to-host connections. The two applicable files are configured in the /etc/sysconfig/network-scripts/ directory. One is `ifcfg-net`, where `net` is the name of the network-to-network device. The configuration is a bit more complex, as it includes a few more directives. But several should already be familiar:

```

ONBOOT=yes
IKE_METHOD=PSK
DSTGW=192.168.122.1
SRCGW=192.168.0.1
DSTNET=192.168.122.0/24
SRCNET=192.168.0.0/24
DST=192.168.122.27
TYPE=IPSEC

```

The “new” directives include the destination network gateway address (**DSTGW**), the source network gateway address (**SRCGW**), the destination network address (**DSTNET**), and the source network address (**SRCNET**). The network address format is in Classless Inter-Domain Routing (CIDR) notation, which should already be familiar to anyone who has previous Linux command-line experience.

The second applicable file is for the authentication keys. While the name of the file should be something like **keys-net**, the format should be identical to the previously described **keys-host** file for host-to-host connections. Of course, the actual key should be different.

Network-to-Network IPSec on Ubuntu

A similar configuration for Ubuntu-based network-to-network IPSec connections is available from the default version of the **racoon-tool.conf** file in the **/etc/racoon/** directory.

Helping Users Who Must Use Telnet

Just as the customer is always right, the user is always right. One might think that all users with more computer experience would be more understanding of security concerns. But old habits die hard. Users who have used Telnet for remote connections may always want to use Telnet for remote connections. There are cases where either the hardware or legacy application in use requires Telnet. Some much older and more fragile hardware can’t keep up with the demands of encryption, so it can’t support SSH.

Unfortunately, it’s far too easy to read usernames and passwords sent using cleartext protocols such as Telnet. You’ll read more about cleartext access later. For now, it’s important to consider the more secure alternatives for remote connections to regular Telnet.

Persuade Users to Convert to SSH

Ideally, you’ll be able to persuade local users to connect to the SSH service described earlier in this chapter using passphrases. You might even turn off Telnet servers and present SSH as the only available alternative. SSH is the most secure option available for remote access. One possibility is to present the steps associated with the two different remote access options, side by side. One example is shown in [Figure 8-3](#), where a login to a regular Telnet server is compared to a login to an SSH server using a passphrase. If you can convince users that the SSH connection is no more difficult than the Telnet connection—and is more secure—you will have done an excellent job selling the most secure remote access option available.

Logging In Via Telnet

```
[michael@RHELserver ~]$ telnet 192.168.122.27
Trying 192.168.122.27...
Connected to LucidServerMin (192.168.122.27).
Escape character is '^]'.
Ubuntu lucid (development branch)
LucidServerMin login: michael
Password:
Last login: Thu Apr  1 09:40:13 PDT 2010 from rhelserver.example.org on
pts/0
Linux LucidServerMin 2.6.32-14-generic-pae #20-Ubuntu SMP Sat Feb 20 07
:07:46 UTC 2010 i686

For official documentation, please visit:
 * http://help.ubuntu.com/
michael@LucidServerMin:~$ Connection closed by foreign host.
```

Logging In Via SSH

```
[michael@RHELserver ~]$ ssh 192.168.122.27
Enter passphrase for key '/home/michael/.ssh/id_rsa':
Linux LucidServerMin 2.6.32-14-generic-pae #20-Ubuntu SMP Sat Feb 20 07
:07:46 UTC 2010 i686

For official documentation, please visit:
 * http://help.ubuntu.com/
Last login: Thu Apr  1 09:41:33 2010 from RHELserver.example.org
michael@LucidServerMin:~$
```

FIGURE 8-3

A comparison of Telnet and SSH remote access methods.

Install More Secure Telnet Servers and Clients

Unfortunately, not all users are easily convinced. You may need to consider alternatives—specifically, more secure versions of Telnet. Packages associated with SSL and Kerberos are available. Unfortunately, if the user is not using SSL or Kerberos-based Telnet clients, the default configuration of either Telnet server accepts regular cleartext Telnet clients.

You can address this issue on both the client and server. On the client, you can delete or move the commands associated with regular Telnet clients. You could create links between the expected client commands in their expected directory locations and the actual SSL or Kerberos-enabled client commands. For example, the following commands create soft links between the standard Telnet client command and the more secure options described in this section:

```
# ln -s /usr/bin/telnet-ssl /usr/bin/telnet
# ln -s /usr/kerberos/bin/telnet /usr/bin/telnet
```

You can also make sure the Telnet server accepts only those connections for which it was designed. The SSL version of Telnet is configured in the internet super server configuration file, /etc/inetd.conf. In that file, you can add the **-z secure** switch, which accepts only secure connections. If a remote user tries to connect with a regular Telnet client, that person will see the following message before he or she can enter a username or password:

```
telnetd: [SSL required - connection rejected].
Connection closed by foreign host.
```

In a similar fashion, a Telnet client attempt to connect to a Kerberos-enabled Telnet server works only when appropriate Kerberos server and client options are configured as described earlier in this chapter. Without those settings, a user who attempts to connect to a Kerberos-enabled Telnet server will get the following message before he or she can enter a username or password:

```
Unencrypted connection refused. Goodbye.
Connection closed by foreign host.
```

Securing Modem Connections

If you have to administer connections from telephone modems, the standard used is the Remote Authentication Dial In User Service (RADIUS). The open source implementation of this software is based on the work of the FreeRADIUS project, available from <http://freeradius.org/>. The implementation of a RADIUS server, including FreeRADIUS, will not be covered here. However, it can be downloaded from the noted Web site, and is also available from standard Ubuntu software repositories. If you need remote user authentication, you may also look into DIAMETER, which is an enhancement of RADIUS—the joke, of course, being that a DIAMETER is twice a RADIUS.

The Basics of RADIUS

RADIUS is designed to provide authentication, authorization, and accounting for remote users who want to connect to a network service. In RADIUS documentation, these functions are known as AAA. First, with respect to authentication, different modules allow you to link its authentication directives with the username and password databases of your choice. Those databases are used to authenticate users, as well as to set up what such remote users are authorized to do. You can set up accounting for the actions of users in various RADIUS configuration files.

In other words, it uses authentication systems such as the Password Authentication Protocol (PAP), Challenge-Handshake Authentication Protocol (CHAP), and Extensible Authentication Protocol (EAP). The first two options are commonly associated with connections over telephone modems. EAP is an authentication system often used on wireless networks.

Such authentication systems require access to a username and password database. RADIUS can work with a variety of such databases, including LDAP, Network Information Service (NIS), Microsoft's Active Directory, Novell's eDirectory, and even the old standby, the local shadow password suite.

RADIUS Configuration Files

When installed on an Ubuntu system, configuration options are commented into a number of files in the /etc/freeradius/ directory. On some other systems, you'll find these files in the /etc/raddb/ directory. Table 8-2 describes the function of each of these files and subdirectories.

Moving Away from Cleartext Access

To understand the risks associated with cleartext access, look at [Figure 8-4](#). It's an excerpt from the Wireshark network analyzer. It can read the contents of every packet sent over the local network. Every system on an Ethernet network can listen in on all network communications, even those between two remote systems on the LAN. That's just the way Ethernet works. [Figure 8-4](#) illustrates the contents of one packet, which happens to be one letter of a password.

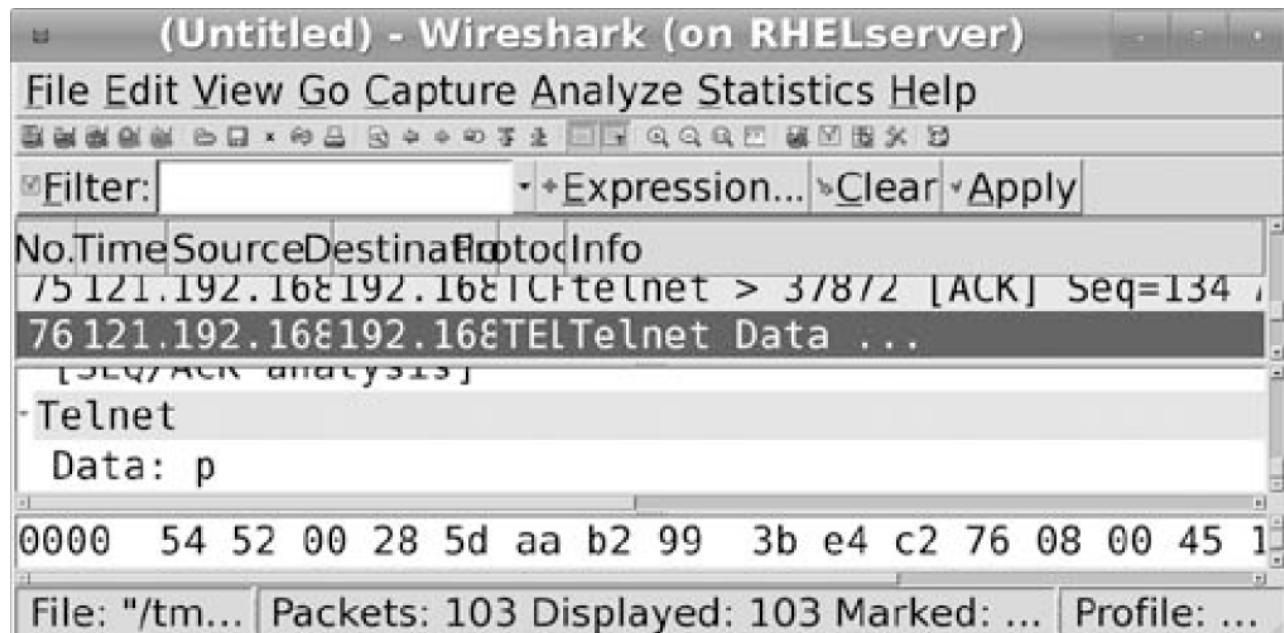


FIGURE 8-4

The Wireshark network analyzer looks at the contents of packets.

TABLE 8-2 Basic FreeRADIUS configuration files.

FILE	DESCRIPTION
aacct_users	Specifies the accounting method for active users
attrs	Includes security and configuration information for RADIUS realms; related to the rlm_attr_filter module
attrs.access_reject	Enforces minimal information attributes when access to a RADIUS server is rejected or for accounting; related to the rlm_attr_filter module
attrs.accounting_response	
attrs.pre-proxy	Specifies authentication information to be sent through a proxy to another RADIUS server; may be coupled with post-proxy information
certs/	Sets the directory with SSL certificates, when applicable
clients.conf	Defines client configuration
dictionary	Refers to mostly hardware-specific dictionary files in the /usr/share/freeradius/dictionary/ directory for different attributes
eap.conf	When EAP authentication is used, defines different certificates and responses
experimental.conf	Defines experimental modules
hints	

	Specifies hints to be added or stripped; commonly used with Point-to-Point Protocol (PPP) and related requests
huntgroups	Defines IP address ranges with restricted access
ldap.attrmap	Matches RADIUS and LDAP directory items
modules/	Sets the directory with RADIUS modules
otp.conf	Supports one-time password authentication
policy.conf, policy.txt	Describes sample policies
preproxy_users	Specifies user information for a remote RADIUS server
proxy.conf	Defines a proxy server configuration
radiusd.conf	Defines the main RADIUS configuration file
sites-available, sites-enabled	Sets directories with enabled and available virtual servers
sql.conf	Includes Structured Query Language (SQL) modules
sqlippool.conf	Specifies a pool of IP addresses to use
templates.conf	Preconfigures multiple options
Users	Defines how users are handled

Wireshark is readily available for most Linux distributions, including Red Hat and Ubuntu. Earlier in this chapter, you saw alternatives to regular Telnet that are better at hiding usernames and passwords in network communication.

The Simple `rsync` Solution

The `rsync` command is frequently used for backups. This section examines the basic features of the `rsync` command. The first time you back up a system with the `rsync` command, everything might seem slow. With its archive features, the `rsync` command can back up more than just the information in regular files. It can preserve more, including the following:

- Ownership by users and groups
- Last file-access times
- Permissions
- Symbolic links between files

An archive `rsync`-based backup with all such original information can form the basis for a gold baseline. The dates associated with files on the backup provide assurance that such files have not been changed after a certain date. Such a date can be important in case of a security breach, especially if your managers suddenly refuse to trust files created after that date.

One of the risks associated with the `rsync` command is that it normally transmits data in cleartext. If SSH clients are installed on the local system, `rsync` doesn't have to take such risks. For example, the following command takes the contents of user michael's home directory recursively (with all subdirectories) and sends it to the remote system named `backup.example.org` in the `/backups/` subdirectory:

```
$ rsync -e ssh -aHz /home/michael backup.example.org:/backups/
```

The **-e ssh** tunnels the backup packets over an SSH connection. If you haven't yet connected to the noted remote system, the remote SSH server prompts for a password or sends a message to verify the passphrase described earlier in this chapter. The **-a** transmits the files in archive mode. The **-H** includes hard-linked files in the archive. The **-z** compresses the data being transmitted, speeding the backup.

E-mail Clients

Standard e-mail clients connect to servers using three major protocols:

- **Simple Mail Transfer Protocol (SMTP)**—An Application Layer e-mail protocol, the **Simple Mail Transfer Protocol (SMTP)** is used primarily for outgoing messages from clients.
- **Post Office Protocol version 3 (POP3)**—Another Application Layer e-mail protocol, **Post Office Protocol version 3 (POP3)** supports e-mail client downloads of incoming messages.
- **Internet Message Access Protocol version 4 (IMAP4)**—Also an Application Layer e-mail protocol, **Internet Message Access Protocol version 4 (IMAP4)** supports client access to remote servers.

TABLE 8-3 Cleartext e-mail protocols and secure alternatives.

CLEARTEXT E-MAIL PROTOCOL/PORT	SECURE ALTERNATIVE/PORT
IMAP4/143	IMAPS/993
POP3/110	POP3S/995
SMTP/25	SMTPS/465

Most users configure e-mail clients such as Evolution, Thunderbird, and especially Microsoft's Outlook Express to send and receive their e-mail communications using these protocols.

Every time a request to send or receive e-mail is sent from a client to an e-mail server, the username and password are also sent over the network. With these cleartext protocols, it's easy for a malicious user to identify those usernames and passwords with tools such as Wireshark.

Fortunately, there are secure alternatives to each of these protocols, briefly described in [Table 8-3](#). In these cases, the alternative includes an *S* in the name, which is short for *secure*. In all cases, such protocols communicate over different port numbers. In general, the IMAP4 and POP3 protocols serve e-mail to clients, and the SMTP protocol transmits e-mail from clients.

These are just standard ports. Some e-mail administrators have been known to use nonstandard port numbers in the hope that obscurity will minimize attempts by spammers to use their e-mail servers.

Such secure alternatives won't work unless they're configured in appropriate e-mail server configuration files. Just as a preview, some e-mail administrators use port 587 for e-mail transmission using the submission protocol, which is for practical purposes interchangeable with port 465 for the secure SMTP (SMTPS) protocol. Some open source SMTP e-mail services can be modified to use either port 465 or 587.

Best Practices: Networked Filesystems and Remote Access

While it's best to minimize the number of networked services on any one system, that may not be enough for a secure system. Ideally, all you need on a server is one networked service and an SSH server to administer that service remotely. But if you want Kerberos authentication, you'll need a Kerberos server as well. And keeping Kerberos servers in sync requires access to NTP servers. If you do use Kerberos servers, stick with version 5. Once

these are configured, you'll note that those clients that connect to your servers have been previously authenticated with genuine Kerberos tickets.

One benefit of Kerberos is with services such as NFS. While NFSv4 doesn't directly authenticate by user, it can help to authenticate NFSv4 clients with Kerberos. Shared NFS directories that look for Kerberos tickets are less likely to fall into the hands of malicious users.

The vsftpd service can be secured in a number of ways with respect to special directories, privilege management, timeouts, user lists, and more. While most vsftpd options are configured in the vsftpd.conf file, users can be limited in the ftpusers and user_list files. When enabled, PAMs can further enhance vsftpd security.

With Samba, a Linux system can take a role as a more secure Windows server. It can be configured as a PDC, as a standalone server, and as a member of an Active Directory domain. Various Samba options can enhance obscurity, limit access by network interface and IP address, set up WINS name resolution, and more.

Samba can be configured with local databases for domains or can refer to other systems to verify usernames and passwords. Samba can be configured as a master browser for the LAN or even the Microsoft domain. Samba can be configured to serve Microsoft logon scripts and profiles to support a consistent level of access on different remote systems.

The SSH server can be configured in detail in the sshd_config file. Options can help regulate how users log in, the use of public keys, authentication with Kerberos tickets, and more. SSH commands even support secure encrypted connections to FTP servers, along with secure copying to and from remote SSH servers. SSH passphrases can be configured to enhance security even further. With a private/public key pair, the passphrase isn't even seen on the network.

Many users have to communicate over the Internet. Enterprise networks are connected over the Internet. Without encryption, such communications would be especially vulnerable. You can reduce the risks with IPSec connections. Such connections can be configured on a host-to-host or network-to-network basis. To support IPSec, the key protocol numbers are 50 and 51; the key port numbers are 50, 51, 500, and 4500. Red Hat and Ubuntu configure IPSec connections in different file locations, in the /etc/sysconfig/network-scripts/ and /etc/racoon/ directories, respectively.

Telnet is still a popular option for remote connections. Because it's a cleartext protocol, it's easy to find usernames and passwords sent over Telnet with tools such as Wireshark. If you can't convince users to convert to SSH, you can at least set up Telnet services associated with SSL or Kerberos.

If you need to set up remote connections by telephone modem, the most popular standard is RADIUS. It can be configured to authenticate using protocols such as PAP, CHAP, and EAP. It can be configured to authorize users via a variety of username/ password databases. RADIUS includes a number of complex configuration files stored either in the /etc/freeradius/ or the /etc/raddb/ directory.

As you pay attention to cleartext protocols like Telnet, you should also remember the cleartext methods used for backups and e-mail. Fortunately, the advantages of SSH can be incorporated directly into rsync commands. In addition, there are secure versions of major e-mail protocols available that can be configured on client systems.



CHAPTER SUMMARY

While it's best to minimize the number of services on any one system, that's not always consistent with the best options for security. In this chapter, you examined Kerberos, along with how it can be used to enhance NFS security with the help of NTP services. Beyond Kerberos, the vsftpd service includes a wide variety of options that have made it the preferred FTP service for Red Hat, SUSE, and the developers of the Linux kernel.

This chapter also described how Samba can be configured in a number of different roles on Microsoft-based networks. It described how you can maximize the security features of SSH, including the use of passphrase-secured private/public key pairs. It showed you basic configuration options associated with IPSec. It described

more secure options to regular cleartext Telnet for remote access. Also in the area of remote access, RADIUS is the major Linux package for connections from telephone modems. In addition, you need to know about secure options for the `rsync` commands and various e-mail protocols, as they normally also send data in cleartext.



KEY CONCEPTS AND TERMS

Andrew Filesystem (AFS)
Authentication Header (AH) protocol
Backup Domain Controller (BDC)
Encapsulating Security Payload (ESP) protocol
Generic security services application program interface (GSSAPI)
Internet Key Exchange (IKE) protocol
Internet Message Access Protocol version 4 (IMAP4)
Internet Protocol Security (IPSec)
Kerberos
Kerberos principal
Kerberos realm
Kerberos ticket
Key distribution center (KDC)
Master browser
Network Address Translation (NAT) traversal protocol
Network Basic Input/Output System (NetBIOS)
Nonprivileged user
Open Systems Interconnection (OSI) Reference Model
Post Office Protocol version 3 (POP3)
Primary Domain Controller (PDC)
Simple Mail Transfer Protocol (SMTP)
Ticket-granting server (TGS)
Ticket-granting ticket (TGT)
Windows Internet Name Service (WINS) servers
X11



CHAPTER 8 ASSESSMENT

1. Which of the following services are required with Kerberos?
 - A. Telnet
 - B. NFS
 - C. NTP
 - D. Samba
2. Which of the following files would you expect to contain Kerberos keys?
 - A. krb5.keys

- B. users krb
 - C. michael.key
 - D. user.keytab
3. The protocol that allows Kerberos to work with different file-sharing services is _____.
4. The vsftpd **directory** directive changes the default directory for anonymous access.
- A. True
 - B. False
5. Which of the following Samba directives sets the name for the local server?
- A. **hostname**
 - B. **netbios name**
 - C. **server_name**
 - D. **domain name**
6. If you want to set up a PDC, what should be the value of the **security** directive?
- A. **user**
 - B. **domain**
 - C. **server**
 - D. **ads**
7. Which of the following bits of information is contained in a WINS server?
- A. Usernames
 - B. Permissions
 - C. NetBIOS names
 - D. Hostnames
8. _____ is the name of the full path to the directory with SSH keys for user donna.
9. Which of the following directives specifies that SSH listens on a network card with a network address of 192.168.0.0?
- A. **ListenAddress 192.168.0.1**
 - B. **ListenAddress 192.168.0.0/24**
 - C. **ListenAddress 192.168.0.0/255.255.255.0**
 - D. **ListenAddress 192.168.0.255**
10. In what file on a remote system would you copy an SSH public key?
- A. .ssh/authorized_keys
 - B. .ssh/id_rsa.pub
 - C. .ssh/id_dsa.pub
 - D. .ssh/idAuthorized.pub
11. TCP/IP port 443 is associated with IPSec connections.
- A. True
 - B. False
12. Which of the following directories may contain configuration files for IPSec connections? (Select two.)
- A. /etc/sysconfig/network-scripts/
 - B. /etc/ipsec/
 - C. /etc/network/
 - D. /etc/racoon/
13. Which of the following services or protocols can be used to add security to Telnet?
- A. SSL
 - B. NTP

- C. SSH
 - D. PAP
- 14.** Which of the following authentication systems is not normally configured with RADIUS?
- A. CHAP
 - B. PAP
 - C. PPP
 - D. EAP
- 15.** Which of the following ports is a secure alternative for SMTP?
- A. 25
 - B. 110
 - C. 993
 - D. 465

CHAPTER

9 Networked Application Security

N

ETWORK SERVICES generally have a wide variety of security-related features and options. These of course include Apache; Squid; the Berkeley Internet Name Domain (BIND); mail transfer agents (MTAs) such as sendmail, Sendmail, and Postfix; Voice over IP (VoIP) services such as Asterisk; the Common Unix Printing System (CUPS); and the Network Time Protocol (NTP). Options for obscurity are also addressed. Because Apache and BIND are especially popular options on the Internet, both application services are prime targets for attacks.

An extensive array of applications can be included with Apache. These applications include different databases and scripting languages. Squid is a Web proxy server; as such, it can cache and filter Web content. Attacks on Domain Name Service (DNS) software such as BIND can keep customers away from corporate Web sites. Even worse, problems in the DNS database can redirect customers to malicious Web sites.

Mail servers are often at risk because of the stakes involved in spam operations. Any mail server that doesn't lock itself down will be targeted for either relaying spam or just simply allowing hundreds or thousands of messages per day to be sent to its users. It's for this very reason that security features and options on systems like sendmail and Postfix are especially important. Because Asterisk has developed into the Linux open source Voice over IP (VoIP) application, attackers have looked at it as a different way to tap into corporate communications. Properly configured, CUPS security can be used to restrict the hosts and users that can print through shared printers hosted by a Linux server. While a break into an NTP server may not have many direct consequences, the unsynchronized systems that may result can affect everything from Kerberos tickets to inventory databases.

Chapter 9 Topics

This chapter covers the following topics and concepts:

- Which options you have for secure Web sites with Apache
- How to work with Squid
- How to protect DNS services with BIND
- What the various mail transfer agents are
- How to use Asterisk
- How to limit printers
- How to protect time services
- How to obscure local and network services
- What best practices are for networked application security

Chapter 9 Goals

When you complete this chapter, you will be able to:

- Regulate Web sites, Web access, and name services

- Modify MTAs to focus on secure services
- Protect miscellaneous networked applications, including Asterisk and NTP
- Customize how printers are managed.

Options for Secure Web Sites with Apache

Apache used to be the dominant Web server. According to Netcraft, it had more than a 50 percent market share for years. In recent years, however, it has dropped. Other servers like Nginx have taken some of that market share. It's easy to configure multiple Web sites on a single Apache server, all connected on a single IP address. It's almost as easy to configure these Web sites with Secure Sockets Layer (SSL) certificates to encrypt communications between the Web server and user client browsers. While Transport Layer Security (TLS) has superseded SSL in many cases, TLS is effectively just the next version of SSL. Most documentation still refers to SSL. While some implementations of SSL still allow the use of SSL v3, there are enough issues in SSL to suggest that it should be removed from all servers that need Transport Layer encryption.

One reason Apache is so popular is that it does not work alone. It can be integrated with a variety of database and scripting languages. One popular combination is known as the **Linux/Apache/MySQL/P (LAMP) stack**, where the *P* can stand for **Perl**, **Python**, or **PHP: Hypertext Preprocessor (PHP)**. (Perl is a dynamic scripting language developed by Larry Wall. Python is a multi-paradigm programming language. PHP is a scripting language associated with dynamic Web sites.)

The sections that follow describe the components of the LAMP stack and highlight some of the modules that are most frequently integrated with Apache. In the following sections, you'll see some of the methods that can be used to limit access to a Web site and how you can configure Apache to serve secure Web sites. When it comes to Web sites, there are two components to think about. The first is the configuration, which is discussed here. The second is the actual content of the site. This is outside the scope of this chapter. It will depend on the actual language and other technology used in the development of the site content.

The LAMP Stack

The components of the LAMP stack vary depending on the preferred scripting language. This section is an overview; there are many excellent books on both MySQL and each of the noted scripting languages. If you are interested in learning good programming practices in any of the languages you may use on the back end of a dynamic and functional Web site, you should investigate Web sites and books related to the language and technologies you are using.

The focus of this section is the Apache configuration files. The names and directories with these files vary by distribution. For example, on Red Hat systems, the main Apache configuration file is httpd.conf in the /etc/httpd/conf/ directory. On Ubuntu systems, the main Apache configuration file is apache2.conf in the /etc/apache2/ directory.



TIP

On Ubuntu server systems, a quick way to install the packages of the LAMP stack is with the console-based tasksel browser. Run the `tasksel` command with administrative privileges. It'll install all the packages required for the LAMP stack.

MySQL and Apache

Many Web sites require access to databases. Those databases may help with authentication, logs, inventory, and more. Appropriate packages may include software that links to the desired scripting language, such as Perl,

Python, or PHP. This section describes security-related directives in the default version of the main MySQL configuration file.

The main MySQL configuration file is my.cnf. Depending on the distribution, you may find that file in the /etc/ or /etc/mysql/ directory. Security-related directives from that file are described in [Table 9-1](#).

If you use MySQL, be sure to open port 3306 on any applicable firewall between the Web server and MySQL database server. Don't open that port on an external firewall (unless required for other reasons). If you prefer a different port, that's easy to configure in the main MySQL configuration file with the `port` directive. If you are running your SQL server on the same system as your Web server, you should use sockets rather than a network listener. This ensures the communication remains exclusively on the system.

TABLE 9-1 Default security-related MySQL directives in my.cnf.

DIRECTIVE	DESCRIPTION
<code>bind-address</code>	IP address of associated network card to which MySQL listens
<code>datadir</code>	Directory with MySQL data; may require special protection
<code>port</code>	Port number
<code>ssl-ca</code>	SSL certificate authority file
<code>ssl-cert</code>	SSL certificate file
<code>ssl-key</code>	SSL certificate key file



NOTE

Sun Microsystems acquired MySQL in 2008 but Sun itself was subsequently acquired in 2009. Oracle has its own database system while still maintaining the MySQL database that is available for free. If at any time the development of MySQL were to be compromised in favor of Oracle's enterprise-level database—although it hasn't been affected in the years after the acquisition—other database systems are available. One is PostgreSQL. Packages are available for PostgreSQL on all of the popular Linux distributions.

The first time you install MySQL on Debian-style systems, including Ubuntu, the installation process prompts you for a MySQL root password. This user is different from the standard Linux root administrative user, and should be configured with a different password. To set that password differently, use the `mysqladmin` command, similar to the following. In this command, the quotes are required around the new password.

```
# mysqladmin -u root password "newpassword"
```

Unless a MySQL root account is configured, any user on the local system has access to the MySQL database. That could be a significant security issue.

Apache and Scripting Languages

There are a number of reasons you might want to include a scripting language in your Linux installation. One is that some scripting languages are used to provide programmatic interfaces to users through Web sites. These include PHP, Perl, and Python. All these languages have been used to develop Web content. The challenge is that once you install a scripting language, it can be used by anyone who has or gains access to the system. Scripting languages can be used to accomplish a lot of things, including to run attack tools. However, once

someone has access to a system, that person may already have everything he or she needs to install a scripting language.

Using an overall strategy of limiting the number of packages installed on a system, it's best to limit the number of scripting languages that you install. This limits the risk associated with attackers or malicious users employing the scripting languages against you and your network. It also reduces the risk associated with another package that may have vulnerabilities.

Apache Modules

One of the strengths of the Apache Web server is its modularity. If you have access to both Red Hat and Ubuntu systems, examine the differences. You'll learn more about how Apache can be configured. One list of modules is shown in [Figure 9-1](#).

The process for disabling a module depends on how the Apache configuration files are set up. Current Ubuntu systems include mods-available/ and mods-enabled/ subdirectories in the /etc/apache2/ directory. As the names suggest, those subdirectories list available and enabled modules. Obviously, it's best if the package associated with the module is uninstalled. But you can also use the `a2dismod` command. For example, the following command removes the PHP version 5 (php5.conf, php5.load) configuration files from the mods-enabled/ subdirectory:

```
# a2dismod php5
```

You can reverse the process with the corresponding `a2enmod` command. Once loaded, you'll need to reload or restart the standard Apache script with a command like the following:

```
# /etc/init.d/apache2 force-reload
# /usr/sbin/apachectl -k restart
```

Current Red Hat systems don't include an equivalent command to disable or enable a module. So to disable a module, you'll need to remove or comment out the directive in the Apache configuration file. For example, the following directive enables common gateway interface (CGI) scripts:

```
LoadModule cgi_module modules/mod_cgi.so
```

On both systems, you can review currently loaded modules with the `apachectl` (Red Hat) or `apache2ctl` (Ubuntu) command, with the `-t -D DUMP_MODULES` option.

On Red Hat systems, some modules are loaded with configuration files in the /etc/httpd/conf.d/ directory. For example, if PHP is installed, you'll see the `php.conf` file on that directory. Such modules are loaded courtesy of the following directive in the main Apache configuration file:

```
Include conf.d/*.conf
```

```
michael@UbuntuHH:~$ ls /etc/apache2/mods-enabled/
alias.conf          autoindex.conf  mime.load
alias.load          autoindex.load  negotiation.conf
auth_basic.load     cgi.load       negotiation.load
authn_file.load     dir.conf      setenvif.conf
authz_default.load  dir.load      setenvif.load
authz_groupfile.load env.load     status.conf
authz_host.load     headers.load  status.load
authz_user.load     mime.conf

michael@UbuntuHH:~$
```

FIGURE 9-1

Apache modules on Ubuntu.

If you were to change the extension of the php.conf file, Apache would not load that module the next time it is reloaded or restarted.

The modules that should remain loaded depend on the desired functionality of the Apache server. For a list of available modules, refer to the relevant section of Apache documentation at <http://httpd.apache.org/docs/2.2/mod/>. The developers of the Apache project have been consistent with their documentation. In other words, if you're working with Apache version 2.0, modules are documented at <http://httpd.apache.org/docs/2.0/mod/>.

Security-Related Apache Directives

In this section, you'll look at the main Apache configuration file and focus on security-related directives not otherwise covered in the sections that follow. This section is focused on the default Red Hat version of the Apache configuration file.

A couple of security-related Apache directives are **Listen** and **ServerTokens**. The **Listen** directive can specify an alternative port for Web pages. For example, the following directive can be found in the configuration for an SSL-enabled Web site. Port 443 is the standard for the secure Hypertext Transfer Protocol (HTTPS).

Listen 443

The **Listen** directive can also specify the IP address of a network card, which is useful if there are multiple network cards on the local system. For example, the following directive would listen for requests through port 443 on the noted IP address:

Listen 192.168.100.10:443

The **ServerTokens** directive is of special interest. It determines what's seen if a user navigates to a nonexistent Web page. While it's nominally an error message, that message can provide far too much information to an adversary. If the configuration **ServerTokens Full** is set, the following error message is sent in response to a nonexistent Web page request:

Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.1e-fips PHP/5.4.16

If a malicious user recognizes a version of one of these applications that happens to be vulnerable, he or she is in business, and your systems will be at risk. So ideally, this information should be kept as minimal as possible. For example, the **ServerTokens Prod** directive leads to the following error message:

Server: Apache

The **Prod** directive tells Apache to disclose only the product name and nothing else. There are other places where Apache may leak information that could be useful to an attacker in addition to the headers the server provides. Look for the **DocumentRoot** and **ErrorDocument** directives in the Apache configuration file. It'll help you identify an appropriate location for a custom error file. By putting a custom error file in a place where Apache can find it, you can create an error page that doesn't have any information about the server. By default, Apache error pages yield information about modules and versions. An attacker can use this information to target vulnerabilities known to be in the versions of the modules or the Apache software itself.

KeepAlive Directives

The next important security-related directive in the Apache configuration file is **KeepAlive**, which supports persistent connections from remote systems.

In a more secure system, the values of **KeepAlive** directives are kept to a minimum. But such a configuration is not consistent with the availability of Web pages served by Apache. Good values for **KeepAlive** directives do improve client performance.

Related directives (**MaxKeepAliveRequests**, **KeepAliveTimeout**) are almost self-explanatory and are well explained by the comments in the Red Hat version of the configuration file. In brief, while **MaxKeepAliveRequests** specifies the maximum number of allowed keep-alive requests on each connection, the **KeepAliveTimeout** directive specifies the wait period for the next request, in seconds.

Multi-Processing Modules (MPMs)

Closely related to the **KeepAlive** directives are the **Multi-Processing Modules (MPMs)** directives. Both the default Red Hat and Ubuntu versions of the Apache configuration file use nearly all the same directives, which are described in [Table 9-2](#). These directives relate most closely to the availability of the Apache system. While you could maximize each of the directives shown, they're limited by the hardware of the local system, other processes that may be running, and the space available to the system. Some trial and error may be required to find optimal settings for these directives. These directives are not in alphabetical order, so they more closely match how they're configured in the Apache configuration file.

TABLE 9-2 Standard MPM directives.

DIRECTIVE	DESCRIPTION
StartServers	Child server processes created during the Apache start process
MinSpareServers	Minimum idle child server processes
MaxSpareServers	Maximum idle child server processes
ServerLimit	Maximum number of clients; also see ThreadLimit in the Apache documentation
MaxClients	Maximum number of client connections processed simultaneously
MaxRequestsPerChild	Maximum number of requests per child server process
MinSpareThreads	Minimum number of idle threads to handle additional requests
MaxSpareThreads	Maximum number of idle threads to handle additional requests
ThreadsPerChild	Number of threads created per child process
MaxRequestsPerChild	Maximum number of requests in the life of a child process

An Apache User and Group

Apache processes should be configured and owned by a single unprivileged user, typically apache or www-data. Such ownership can be configured with the **User** and **Group** directives:

```
User apache
Group apache
```

To ensure such users are unprivileged, make sure anyone who logs into that account doesn't get a shell. That can be configured in the /etc/passwd file. If a distribution configures such users with a regular shell like /bin/sh, that is not as secure as a fake shell such as /bin/false or /sbin/nologin. With a fake shell, a user who is actually able to log into such accounts does not get a command-line shell. If the Apache server were compromised, you wouldn't want the attacker to have a useful shell on the system. Also, because the attacker would inherit permissions from the compromised process, you want to make sure that the user and group associated with the Apache server process doesn't have access to anything on the system. There shouldn't be any directory where the attacker can create files unless the Web application needs to create temporary files—and that should be the only directory that the Apache user and group can write to.

Do Not Allow Overrides with .htaccess

An .htaccess file can be a handy way to add Apache directives for specific directories or virtual hosts. But the .htaccess file can be dangerous. With the dot in front, it's a hidden file, outside the view of a normal search.

If a malicious user is able to add an .htaccess file to a Web server directory, that user can override anything you've configured with directives in that file. To disable the use of the .htaccess file, add the following directive in any stanza with the **Directory** tag:

```
AllowOverride None
```

Do Not Allow Access to User Directories

It's probably far too easy to configure access to user home directories on the system with the Apache server. With the mod_userdir module and the **UserDir enable** directive, user directories can be shared over the Apache Web server. Unless you want files in user home directories exposed to the world, it's something to disable.

To disable Apache-based access to user home directories, disable the mod_userdir module. If you're not prepared to do that, at least avoid the **UserDir enable** directive unless the system includes only guest accounts with files that can be shared publicly.

Minimize Available Options

The **Options** directive is important. It allows you to configure extra features within a **Directory** stanza. The default setting is **Options All**, which includes a number of features shown in [Table 9-3](#) that are inconsistent with a secure system. It includes all features except **MultiViews**.

TABLE 9-3 Apache Options and associated features.

DIRECTIVE	DESCRIPTION
ExecCGI	Allows the execution of CGI scripts if the mod_cgi module is enabled
FollowSymLinks	Supports the following of symbolically linked files in the directory
Includes	Allows the use of server-side includes if the mod_include module is enabled
IncludesNOEXEC	Allows the use of server-side includes except for executable files
Indexes	Lists files in directories where no DirectoryIndex file, typically index.html , is included if the mod_autoindex module is enabled

MultiViews	Supports error-tolerant reads of filenames
SymLinksIfOwnerMatch	Supports the following of symbolically linked files in the directory if the link and the file are owned by the same user

Each of these **Options** directives may be a potential security risk. Because the default includes all but one of the directives, any malicious user who is able to insert a file into one of the listed directories may cause problems. Alternatively, if that user is able to delete an [index.html](#) file, it may expose that directory to all who browse that system.

The way to avoid the security risks associated with [Table 9-3](#) is to disable all **Options** directives with the following command:

```
Options None
```

Configure Protection on a Web Site

Typically, multiple Web sites are configured on individual Apache Web servers. That's made possible with an appropriate configuration of virtual hosts. Web site accessibility can be limited to a certain IP address network with the following directives, which deny access from all systems except those that send requests from addresses with the noted IP network address and network mask:

```
Order deny,allow
Deny from all
Allow from 192.168.0.0/255.255.255.0
```

The same group of directives can be used in a stanza for a specific directory. If desired, you can substitute a partial or complete domain name such as example.org or .com for the noted IP address.

One other protection system is based on users created with the **htpasswd** or **htdigest** commands. If the file with users and passwords is /etc/httpd/conf.d/apacheusers, you could take advantage with the following directives restricting access to a specific Web site or directory:

```
AuthName "Authorized users only"
AuthType Digest
AuthUsersFile /etc/httpd/conf.d/apacheusers
Require valid-user
```

Users who are allowed to access the resource can be limited by username or groupname.

Configure a Secure Web site

By default, Web sites that use the HTTPS protocol use port 443. But enabling the appropriate SSL module requires different commands on different systems. On Red Hat systems, when the mod_ssl package is installed, an ssl.conf file with sample directives for a secure Web site can be found in the /etc/httpd/conf.d/ directory. On Ubuntu systems, when the **a2enmod** command is applied to the ssl module, similar sample directives can also be found in an ssl.conf file, this time in the /etc/apache2/mods-enabled/ directory.

Because the contents of these files are normally included in the main Apache configuration file with an appropriate **Include** directive, all you need to do is customize the ssl.conf file for the desired secure Web site. But a secure Web site requires a certificate authority.

Configure a Certificate Authority

A **certificate authority (CA)** is an entity that issues digital certificates. The actual certificates are based on a private/public key pair. While such key pairs can be generated locally, they would not be official certificates associated with major online enterprises. Such key pairs can be expensive. However, official certificates are required if you want a Web site that doesn't lead to an "invalid security certificate" error. If you're interested, such official certificates are available from companies such as Symantec and GoDaddy.

The standard filenames associated with CAs are arbitrary. However, there are standard file extensions for CAs, as listed in [Table 9-4](#).

TABLE 9-4 Certificate authority (CA) file extensions.

FILE EXTENSION	DESCRIPTION
.crt	Configured for the server certificate
.csr	Created with the certificate signing request
.key	Associated with the CA key

To create a self-signed certificate, you can use the `openssl` command. The following command creates a key file using the Advanced Encryption Standard (AES) cipher with a key size of 256 bits. Certificates are part of a hybrid cryptosystem that uses both symmetric and asymmetric ciphers. On the asymmetric side, this uses the Rivest, Shamir, Adelman (RSA) cipher with a key strength of 1,024 bits.

```
# openssl genrsa -aes256 -out server.key 1024
```

This command prompts for a passphrase. Once the `server.key` file is created, you can create a certificate-signing request file. The following command uses that `server.key` file to create that `server.csr` file.

```
# openssl req -new -key server.key -out server.csr
```

You're prompted for the following information. If you're setting up an official certificate-signing request, the answers must be true and verifiable.

- The passphrase used to create the `server.key` file.
- A country name, based on a standard two-letter country code. Official two-letter country codes can be found in the ISO standard 3166.
- The full name of the state or province associated with the Web server.
- The full name of the locality associated with the Web server.
- The full name of the company or organization behind the Web server.
- An appropriate organizational unit name.
- The name of the responsible administrator.
- The e-mail address of the responsible administrator.
- A challenge password and alternative company name that can be used to verify the request.

The `server.csr` file that's created by the given command can be used as part of an official certificate request. If you want to purchase an official certificate, that file is part of what you'd send to a certificate such as Symantec or GoDaddy.

If you choose to create your own self-signed certificate for secure Web sites, the following command creates the actual server certificate file—in this case, `server.crt`.

```
# openssl x509 -req -days 30 -in server.csr
  ↗-signkey server.key -out server.crt
```

The private/public key infrastructure is based on the x.509 standard of the International Telecommunications Union (ITU). Based on the command, the newly created server.crt certificate is good for 30 days. If you were to really create a certificate that you wanted to use internally, you would typically create a certificate that was valid longer than a month. When you install this certificate into your Apache installation, your browser will generate an error indicating the certificate isn't trusted. This is because the browser doesn't have a root Certificate Authority certificate that generated the certificate used by the server.

The server.crt and server.key files (the filenames can vary) can then be used in conjunction with the `SSLCertificateFile` and `SSLCertificateKeyFile` directives. But these two directives just scratch the surface of what can be done with SSL and Apache. For a more complete list of Apache SSL directives, see http://httpd.apache.org/docs/2.2/mod/mod_ssl.html.

`mod_security`

When you have a Web site that allows users to send data to the server, you allow programmatic access. Your site will be susceptible to a number of attacks, including cross site scripting, SQL injection, command injection, and others. The developers responsible for your Web site may know how to perform appropriate validation of the user input, or they may not. They may also use components that don't check user input appropriately, making the Web application, associated data, and maybe even your server susceptible to attack. What can you do to protect yourself against the things you can't control? You can install an Apache module that is capable not only of detecting these attacks but also preventing them.

The Apache module `mod_security` was developed as a **Web application firewall (WAF)** to help protect your Web application. It allows you to write rules that will detect, log, or deter attacks against your Web application. You can also deploy a set of rules targeted at protecting against common attacks like those just mentioned. The `mod_security` module is available for common Web servers like Apache, IIS, and Nginx. It's also available as a package you can install from the repositories of most popular Linux distributions like Debian, Ubuntu, and Red Hat. Typically, you must install two packages: the Apache module and a separate package called the Core Rules Set (CRS). CRS is managed by the Open Web Application Security Project (OWASP). CRS is a collection of rules that protect your Web application from common attacks as well as a number of other attack types.

The software package installs the module and configuration necessary to enable the module. Typically, to turn on the module, you include the following line in a configuration file (although that configuration file is generally included with the package installation):

```
LoadModule security2_module modules/mod_security2.so
```

The package associated with the distribution will likely have a separate configuration file that has all the module-specific directives in it. This includes telling the module where the active rules are and how to handle logging. Most importantly, you would set a configuration directive telling `mod_security` which mode the security engine is in. By default, it is set to `DetectionOnly`. This means none of the rules are actively deterring attacks. The directive you would use in the configuration file to switch `mod_security` from `DetectionOnly` to "On" is as follows:

```
SecRuleEngine On
```

Having the directive set to `DetectionOnly` is very important when you are first installing `mod_security`. This is because you can't be certain how the rules may interact with your Web site and the application you are hosting. Some rules may break functionality if they start dropping messages that are otherwise legitimate. As a result, setting the entire engine to `DetectionOnly` means that you will get log messages only while you are testing. If, after reviewing the log messages, you determine that some of the rules will break your site functionality, you can either disable the rule or rewrite it so it doesn't block something that may be specific to your site.

An example of one of the rules follows. Although it may appear to be difficult to understand, each rule that's part of the CRS is documented with what it does so you can decide whether to activate it. The only time you'd have to fully understand a rule is if you were to create a rule yourself.

```

SecRule REQUEST_LINE "^GET /$" ?
"chain,phase:2,id:'981020',t:none,pass,nolog"
    SecRule REMOTE_ADDR "^(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})$" "chain,t:none"
    SecRule TX:'/PROTOCOL_VIOLATION\\MISSING_HEADER/' ".*" ?
        "chain,setvar:tx.missing_header+=1,setvar:tx.missing_header_% ?
            {tx.missing_header}=%{matched_var_name}"
        SecRule TX:'/MISSING_HEADER_/' "TX\:(.*)" ?
            "capture,t:none,setvar:!tx.%{tx.1}"

```

This rule is an exception rule indicating that the check Apache does for SSL is OK. Some of the rules consist of regular expression matches, while other sections are specific syntax for mod_security itself. Because this is an exception rule, there is nothing here that would generate a lot of messages. In fact, you can see that it explicitly says `nolog`, and the disposition of any match to this rule should be to pass it through, as indicated in the rule (`pass`).

The mod_security module can be a critical component to your Web application architecture. Because the package installations for the Linux distributions do the bulk of the work for you by installing complete configuration settings, it's one that you should definitely enable if you have any type of form or other dynamic content on your Web site.

Working with Squid

Any Web proxy server can be configured for two basic purposes. It can cache frequently accessed information, and it can regulate user access to remote Web sites.

As for caching, the proxy server for an enterprise where hundreds of users access the same online dictionary Web site can store that information locally. The definitions on that dictionary would be accessed from the Internet only once. Further requests for the same definitions would be returned in this case from the Squid proxy server. A time limit is associated with cached data to make sure that data isn't stale. Because access to locally cached data is faster, perceived Internet performance is improved for local users.

technical TIP

With a few firewall rules, you can channel all Web requests from inside your network through a proxy server. On the gateway between the internal network and the Internet, you'll need to regulate outgoing traffic. Specifically, with the following iptables rule, all traffic destined for port 80 for regular outgoing Web connections is redirected to port 3128, the standard for proxy server connections:

```

iptables -t nat -A PREROUTING -p tcp --dport 80
    ↵ -j REDIRECT --to-port 3128

```

Squid uses the **Inter-Cache Protocol (ICP)** for transfers between peer and parent/child caching servers. It can be used in two directions. As a traditional proxy server, it caches data on behalf of other systems on the local network. As a front-end accelerator, it caches on behalf of clients on remote networks.

Of course, Squid can be configured to avoid caching Internet information that may change in real-time, such as information related to scores of sports events. And obviously, it should also be configured to avoid caching otherwise private information such as user logins to remote financial Web sites.

As for regulation, Squid can block access to certain domains. If you want to keep users from navigating to certain domains, Squid can access lists of such domains. Of course, the reliability of such limits depends on

restrictions. In other words, if a client can bypass a Squid proxy server, users can bypass Squid limits on those domains.

Basic Squid Configuration

Squid is perhaps the leading open source proxy server. It is a caching proxy that supports Hypertext Transfer Protocol (HTTP), HTTPS, and FTP communications. The basic Squid configuration file is `squid.conf`, in the `/etc/squid/` directory. Although the file includes more than 4,000 lines, it is simple to configure. First, the standard port number associated with Squid communication is 3128, which is documented in the following directive:

```
http_port 3128
```

Generally, access is limited based on access control lists (ACLs). Although they share the same acronym as ACLs for Linux files, they are different within Squid. ACLs on Squid can limit access by IP address, hardware address, source domain name, destination domain name, time, uniform resource locator (URL), port number, protocol, browser, referrer, username, and more.

The simplest use of Squid ACLs limits access by IP address. For example, the following directive specifies an ACL rule for a local source private IP address network along with the localhost system:

```
acl localhost src 127.0.0.1/32
acl some_net src 192.168.0.0/24
```

Be aware, this second command replaces the following default, which allows access from all systems:

```
acl all src 0.0.0.0/0.0.0.0
```

The following commands allow access from the localhost system, along with systems on the noted private IP address network, before denying access to all other systems:

```
http_access allow localhost
http_access allow some_net
http_access deny all
```

This is the minimum configuration required to set up Squid on a local area network (LAN). The next step is to create basic cache directories in the `/var/spool/squid/` directory with the following command:

```
# squid -z
```

Once the Squid service is running, you should be able to set up client browsers on the local network to connect through the Squid service on port 3128. Of course, you'll also have to make sure that port is open in any applicable firewalls.

If you want to configure access to additional ports, examine the `acl SSL_ports` and `acl Safe_ports` directives. In the default `squid.conf` file, these directives are used to specify ports for secure and other connections. These are often coupled with the following directives:

```
http_access deny !Safe_ports
http_access deny !SSL_ports
```

The exclamation point (!) is also known as a bang. In code, it's a reference to "anything but." In other words, the directives shown here deny access to all but configured safe ports and SSL ports.

Of course, with more than 4,000 lines, Squid can do a lot more. The sections that follow just scratch the surface of what Squid can do.

**TIP**

If you've activated Security Enhanced Linux (SELinux), be sure to activate the `squid_connect_any` boolean in the `/selinux/booleans/` directory.

Security-Related Squid Directives

Squid can do a lot more. This section just scratches the surface of security-related Squid directives. This section is based on the default `squid.conf` configuration file for version 2.6. These directives include the following:

- **`auth_param`**—A directive that can be configured with a basic, a digest, or an NT LAN Manager (NTLM) authentication scheme.
- **`authenticate_*`**—A series of directives that provide a timeout for connections from clients.
- **`acl`**—Basic Squid-related `acl` directives were described earlier in this chapter.
- **`http_access`**—Basic Squid-related `http_access` directives were described earlier in this chapter.
- **`http_reply_access`**—Specifies systems allowed to answer replies. While the default is `http_reply_access allow all`, limits can regulate what your users can access online.
- **`icp_access`**—Access should be limited to other Squid caching servers.
- **`htcp_access`**—Based on the Hypertext Caching Protocol (HTCP), it sets ACLs to discover other caching servers.

Limit Remote Access with Squid

You can configure limits on remote client access to undesirable domains with the `http_access` and `acl` directives. The simplest way to put this into effect is with a text file of undesirable domain names. For this section, call it `baddomains.txt` in the `/etc/squid/` directory.

In the Squid configuration file, the following line creates an ACL associated with destination domains:

```
acl baddomains dstdomain "/etc/squid/baddomains.txt"
```

Once the ACL is created, the following `http_access` command can deny access to that list of undesirable domains:

```
http_access deny baddomains
```

Protecting DNS Services with BIND

This chapter does not address the detailed configuration of a DNS server. However, it does address security issues related to such configuration. The focus of this section is the Berkeley Internet Name Domain (BIND) software. With that in mind, you should be aware of the four different types of DNS servers available:

- **Master**—A master DNS server is the authoritative server for a domain.
- **Slave**—A slave DNS server relies on a master DNS server for data.
- **Caching-only**—A caching-only DNS server refers requests to other DNS servers. It also stores recent requests in a local cache.
- **Forwarding-only**—A forwarding-only DNS server refers all requests to other DNS servers.

The Basics of DNS on the Internet

DNS is a distributed database of domain names and IP addresses. Domain names like www.example.com. are properly shown with a period at the end. That period is the root domain. That top-level database of DNS servers is known as a *root nameserver*. If you've installed DNS software on a local system, you can review the list of root nameservers in a file like named.root or db.root.

Each of these root nameservers can refer to authoritative nameservers at the next level, for domain names such as com., net., or org.. Those authoritative nameservers can go one level deeper to identify the authoritative nameservers for domains such as jblearning.com.

When a client computer searches for the IP address associated with a domain name, it first looks in the database associated with the local DNS server. If that domain name is not found locally, that DNS server may be configured to send the request to another DNS server. If the domain name is still not found, that request may be sent to one of the root nameservers. That search process is known as a **recursive query**.

A substantial number of DNS servers may be involved in a search for a domain name. That provides plenty of opportunity for a malicious user to jump into the search to redirect unsuspecting users to fake domains. To that end, DNS is a frequent target of attacks. One way to redirect unsuspecting users to fake domains is based on an attack known as *cache poisoning*.

DNS Network Configuration

With such potential vulnerabilities in mind, you may consider configuring different kinds of DNS servers on different parts of a network. To that end, it's important to protect the master authoritative DNS server for your network. Review the network configuration shown in [Figure 9-2](#).

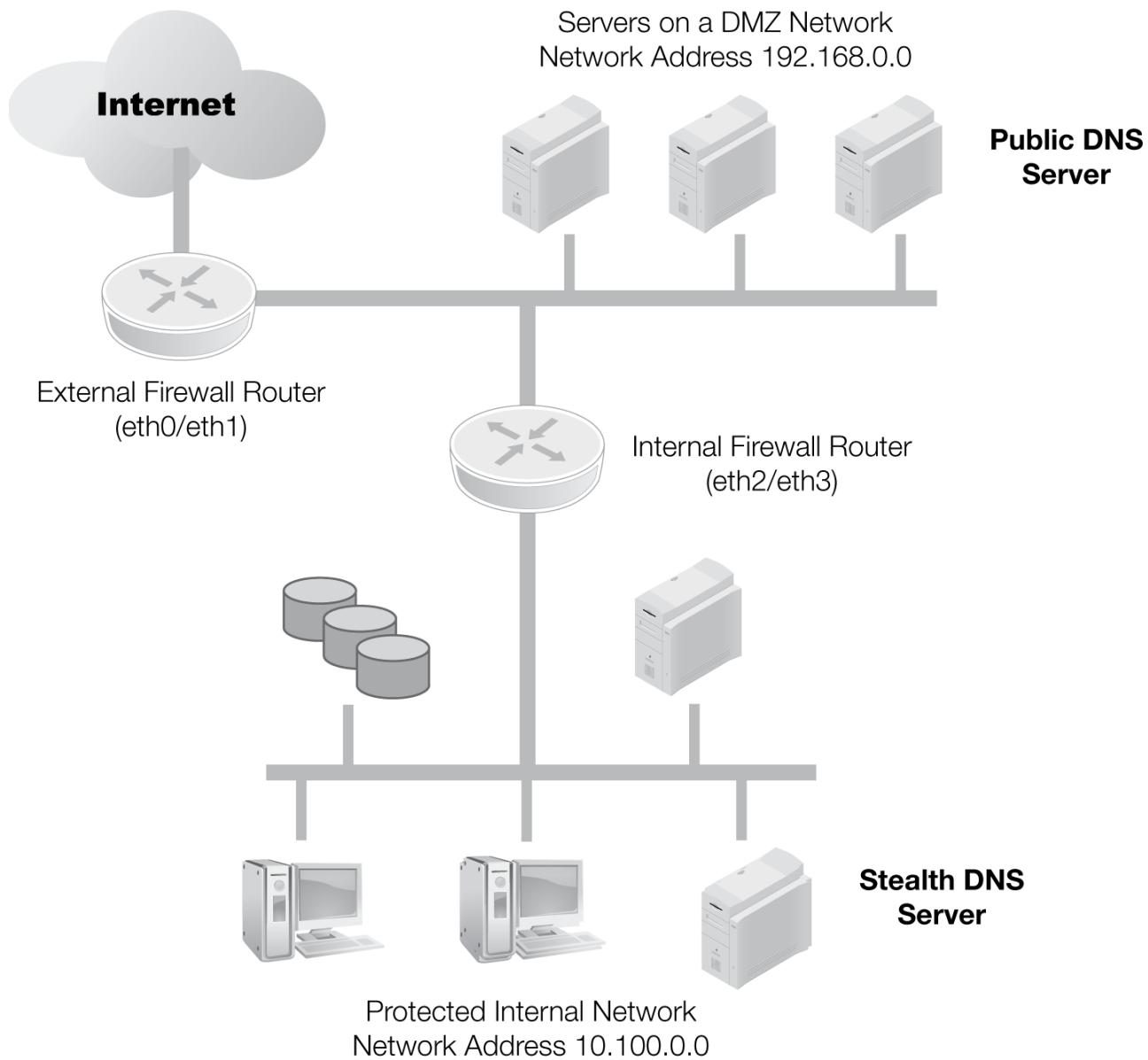
The best place to protect the master authoritative DNS server for a domain is on an internal network, protected by multiple firewalls. You can then configure a second master server with stronger limits that acquires information directly and only from the master DNS server on the internal network.

A common DNS configuration is to use **split DNS**. With split DNS, you have two DNS servers. One is configured to allow public inquiries and the other is for internal clients only. The public DNS server would provide authoritative-only responses. It would not cache any information. It would not accept any recursive queries. The internal server would accept recursive queries, meaning it would allow clients to ask about external hostnames. It might also have authoritative information about the corporate domain for servers that are available on the inside of the network and not exposed through the firewall. The internal DNS server will provide information strictly about internal servers. The IP addresses provided from these lookups may, and probably should, be private addresses.

Secure BIND Configuration

A typical secure BIND configuration file would include a number of directives to minimize its vulnerabilities to attackers. For example, the following directive does not provide any information on the current version number of BIND software:

```
version "not listed";
```

**FIGURE 9-2**

An internal network, a DMZ, and the Internet.

An authoritative-only name server should not search other DNS servers for additional information. The following directive disables such recursive searches:

```
recursion no;
```

Any secure DNS server should not allow transfers of zone databases to minimize the risk of cache poisoning. The following directive allows transfers to no other DNS servers:

```
allow-transfer {none;};
```

A second element of a secure BIND configuration involves a chroot jail. It's automatically configured on Red Hat systems based on the bind-chroot package. It can be configured on Ubuntu systems as well. One set of instructions is available online in the Community Ubuntu Documentation, at

<https://help.ubuntu.com/community/BIND9ServerHowto/>. In either case, any compromise of the DNS server

application would result in an attacker with no administrative privileges and no permissions getting into the directories of other services on that system.

If you have to allow transfers between DNS servers, take advantage of the **Transition SIGnature (TSIG)** key, normally stored in the `rndc.key` file. (TSIG is a protocol used to authenticate data exchanges between DNS servers.) It can be coupled with a **keys** directive in the DNS configuration file. An **allow-update** directive can be coupled with configured TSIG keys. Because it is standard practice to create new TSIG keys on a regular basis, you should become familiar with the `dnssec-keygen` command.

Security may include limits on access. That's made possible with the **acl** directive. Properly configured, it can help define a group of IP addresses. One example is shown here:

```
acl "ipgroup1" { 192.168.0.1; 10.20.30.0/24; };
```

The `ipgroup1` term can then be used in other directives.

A BIND Database

A standard authoritative BIND database should include records for all IP addresses on a domain. Such records are stored in zone files. These are discussed momentarily.

If the functionality of a DNS server is split as described earlier in a stealth configuration, you'd see two different zone file databases. One database, for the DNS server within the internal network, would include IP addresses for all systems on the network. The second database, for the authoritative-only DNS server in the DMZ, would include just the information required for communications from outside systems. Specifically, that database would include the IP address of any other DMZ DNS servers, any other public servers such as Web servers, and any e-mail servers for delivery.

DNS Targets to Protect

To summarize, when configuring an authoritative DNS server for a public system, you'll want to protect **zone files** and **zone updates**, as well as prevent **cache poisoning**.

- **Zone files**—A zone file is a database of hostnames and IP addresses for a specific authoritative domain. A complete zone file includes hostname and IP-address data on all systems on the local domain. Public zone files should be limited to information required for Internet communications, including Web servers and e-mail servers.
- **Zone updates**—A zone update is a reference to a data exchange between DNS servers with respect to hostnames and IP addresses of a specific domain. A slave DNS server relies on a master DNS server for data.
- **Cache poisoning**—Cache poisoning is a frequently malicious insertion of non-authoritative DNS data as if it were authoritative. Data forwarded from DNS servers maintained by an attacker or malicious user could redirect your customers to malicious Web sites.

Domain Name System Security Extensions

The **Domain Name System Security Extensions (DNSSEC)** offer a way of performing validation of the source of information being provided about hostnames and IP addresses. One problem with DNS is that it needs to be fast. As a result, it uses User Datagram Protocol (UDP) packets to transmit information. There is no connection setup between client and server, so you gain speed. Packets also tend to be smaller because there is a little less overhead. You don't want to hold up network requests while a lot of negotiation goes on between the requestor and the DNS servers. The problem is that attackers can spoof UDP easily because of the lack of connection, meaning there is no verification between the server and client. Because spoofing is easy and DNS uses UDP, an attacker can falsify DNS requests if the attacker can get a response to the client more quickly than the server can. To protect against that, the administrator of a DNS server can use cryptographic keys to sign their zone files. The administrator can then use these keys to verify that the information being provided is good.

Fortunately, it's easy to turn on DNSSEC on your DNS server. Assuming you have a BIND9 server, you can perform the following steps to enable DNSSEC. First, turn on DNSSEC in your BIND configuration by editing the /etc/bind/named.conf.options file to add the following lines:

```
dnssec-enable yes;
dnssec-validation yes;
dnssec-lookaside auto;
```

Next, you need to sign your zone files. Your zone files may be located under /etc/bind or in /var/cache/bind. When you are there, you can sign the zone files by using the following command:

```
dnssec-keygen -a NSEC3RSASHA1 -b 2048 -n ZONE example.com
```

This creates an NSEC3 key using the RSA algorithm with SHA1 as the hash algorithm. NSEC3 is a cryptographic mechanism developed for and used specifically by DNSSEC implementations.

After you have created a zone key, you need to create a key-signing key. You can create a key signing key using the following command:

```
dnssec-keygen -f KSK -a NSEC3RSASHA1 -b 4096 -n ZONE example.com
```

This uses the same command but adds a parameter to indicate that you are creating a key-signing key. This is similar to a certificate authority root certificate that is also capable of signing other certificates.

When you have your keys, you need to include them in your zone file. Suppose the filename for one of the keys is [Kexample.com.key](#). You would add the following line to your zone files. You need to add a similar line for all of your .key files.

```
$INCLUDE Kexample.com.key
```

Finally, you need to actually sign the zone file. The following command will sign the file [example.com.zone](#) for the [example.com](#) zone.

```
dnssec-signzone -3 <salt> -A -N INCREMENT -o example.com -t
↳ example.com.zone
```

Notice there is a **<salt>** parameter. This is not intended to be used as it is shown. Instead, it should be a 16-character random value that you will use as a seed to the cryptographic function. There are a number of ways to create this 16-character value, including feeding the output of /dev/urandom into a base64 function or a hashing function and then extracting 16 characters. After you have used the salt, you don't need to remember it. It should be as random as possible.

DNSSEC is a simple feature to enable. There just aren't many good reasons to avoid using it to help protect yourself and the Internet infrastructure from attacks against such a core and critical component. If you can't trust the system that translates the hostnames you use to correct IP addresses, you have some serious issues.

Mail Transfer Agents

The alphabet soup associated with many Linux services continues with various e-mail services. E-mail client applications such as Evolution and Thunderbird are known as **mail user agents (MUAs)**. Services such as fetchmail that collect e-mail from networks are known as **mail retrieval agents (MRAs)**. E-mail services that send such mail along to client applications are known as **mail delivery agents (MDAs)**. When e-mail client applications send messages for delivery, they use **mail transfer agents (MTAs)** such as sendmail, Sendmail, and Postfix.

Sendmail is both an application and a company. The company offers enterprise mail solutions, founded on the application, which has long been an open source mail transfer agent. Sendmail has been around as a mail transfer agent for decades and it was long considered the de facto MTA for many Linux distributions.

Sometimes, **mail submission agents (MSAs)** are integrated with MTAs in services such as sendmail and Postfix. When MSAs and MTAs work together, the MSA can focus on authenticating users and the MTA can prevent relaying from users who don't have local e-mail addresses.

In any case, MTAs such as sendmail and Postfix are configured primarily for the Simple Mail Transfer Protocol (SMTP) and the submission protocol on ports 25 and 587, respectively. MDAs such as Dovecot can be configured to manage traffic using the Post Office Protocol version 3 (POP3) and the Internet Message Access Protocol (IMAP), along with the related protocols with security in their names: POP3S and IMAPS.

Open Source sendmail

The open source sendmail MTA is the default outbound e-mail engine for many Linux distributions. When installed, configuration files can be found in the /etc/mail/ directory. Because the main configuration file (sendmail.cf) includes nearly 2,000 lines, it's a highly customizable system. But because the main configuration file is so large, it has intimidated many Linux administrators. Complexity can be a drawback with respect to security. If you don't understand a security update, the changes you make to the sendmail software may not address the security issue.

That complexity led to the development of the sendmail macro configuration file, sendmail.mc. It's written in a format that is easier to explain. The `dnl` directives in that file are comment characters. All information after the `dnl` on a line is ignored. Some important security-related directives from the default version of the configuration file are described here. First, any `include` directives include the contents of the filename described.

Next, the `VERSIONID` directive provides a comment on the file. In some cases, it may be based on the actual version number of the released sendmail software, which is something that you want to avoid providing a malicious user. To that end, the following directive may be safer:

```
VERSIONID(`setup for linux')dnl
```

One odd format in sendmail configuration files is the quotes. Such quoted directives have a back quote (`) at the start and a regular single quote (') at the end of the quote. Any `define` options that follow specify parameters such as files or limits. For example, the following directive specifies the name of the file with user aliases:

```
define(`ALIAS_FILE', `/etc/aliases')dnl
```

More important, they may include authentication methods. For example, the following `TRUST_AUTH_MECH` directive specifies a list of trusted authentication mechanisms. The `confAUTH_MECHANISMS` directive lists options offered for authentication. So if these directives from the default Red Hat version of the sendmail.mc file are activated, users who connect to these systems from an MUA client such as Evolution or even Microsoft Outlook can choose from external Message Digest 5 (MD5) associated with the **Java Cryptography Extension (JCE)**, MD5 associated with a **challenge-response authentication mechanism (CRAM)**, regular, or plaintext logins. (JCE is a framework for encryption associated with the Java programming language. It may also be used with open source sendmail. CRAM is a group of protocols where a service such as open source sendmail presents a challenge such as a request for a username and password.)

```
dnl TRUST_AUTH_MECH(`EXTERNAL DIGEST-MD5 CRAM-MD5 LOGIN PLAIN')dnl
dnl define(`confAUTH_MECHANISMS', `EXTERNAL GSSAPI DIGEST-MD5
      ↪CRAM-MD5 LOGIN PLAIN')dnl
```

Generally, regular or plaintext logins should not be used without some encryption. But that may not always be possible with some older MUA clients, such as an older version of mutt. Fortunately, several authentication

mechanisms are available and configured, including the generic security service application programming interface (GSSAPI) as well as the MD5 options just described.

You may want to create SSL certificates using the private and public key pair techniques described earlier in this chapter for secure Web sites. The following directives, if active, search for needed certificates and keys in the noted files:

technical TIP

With the FEATURE directive, avoid specifying the `relay_local_from`, `relay_mail_from`, and `accept_unresolvable_domains` options. Doing so can open your system to use by spammers. Networks served by the sendmail service can be configured in `/etc/mail/access`.

In addition, you can take advantage of the `/etc/mail/access` file to reject e-mails from the domains of known spammers and malicious users. For more information on anti-spam features in sendmail, see http://www.sendmail.org/m4/anti_spam.html.

```
dn1 define(`confCACERT_PATH', `/etc/pki/tls/certs')dn1
dn1 define(`confCACERT', `/etc/pki/tls/certs/ca-bundle.crt')dn1
dn1 define(`confSERVER_CERT',
  ↳ `/etc/pki/tls/certs/sendmail.pem')dn1
dn1 define(`confSERVER_KEY',
  ↳ `/etc/pki/tls/certs/sendmail.pem')dn1
```

The FEATURE directive includes an extensive array of options. Some important options for this directive are listed in [Table 9-5](#).

TABLE 9-5 Open source sendmail FEATURE options.

OPTION	DESCRIPTION
<code>accept_unresolvable_domains</code>	Supports access from domains; avoid this option if possible
<code>always_add_domain</code>	Adds the domain name even on e-mail to local users
<code>blacklist_recipients</code>	Allows you to specify users in <code>/etc/mail/access</code> that should not receive e-mail
<code>domaintable</code>	Maps addresses from one domain to another
<code>mailertable</code>	Overrides routing to specific domains
<code>redirect</code>	Rejects mail sent to redirected e-mail addresses
<code>relay_local_from</code>	Supports access from all e-mails that specify a local e-mail address; avoid this option if possible
<code>relay_mail_from</code>	Supports access from all e-mails that specify an e-mail address in <code>/etc/mail/access</code> ; avoid this option if possible
<code>smrsh</code>	Uses a shell more restricted than the regular bash shell
<code>use_ct_file</code>	Reads <code>/etc/mail/trusted-users</code> for a list of users who are trusted
<code>use_cw_file</code>	Reads <code>/etc/mail/local-host-names</code> for a list of alternative hostnames

The DAEMON_OPTIONS directive in default files limits access to the local system. If you want to allow the server to listen to all systems, comment out the line. If you want to allow the server to listen to the network card with an IP address of 192.168.0.10, change the directive as shown for Internet Protocol version 4 (IPv4) addresses.

```
DAEMON_OPTIONS(`Family=inet, Name=MTA-v4, Port=smtp,
  ↘Addr=192.168.0.10')dnl

DAEMON_OPTIONS(`Family=inet, Name=MSP-v4, Port=submission,
  ↘Addr=192.168.0.10')dnl
```

The port numbers, as listed in /etc/services, are 25 and 587. If you're configuring e-mail services for remote systems, be sure that appropriate firewalls allow access through those ports. Unfortunately, if you use a specific IP address in that directive, it can lead to problems. Some firewalls block messages from certain IP addresses, such as those from private IP address networks. In that case, you could comment out these directives.

As suggested in the opening comments in the sendmail.mc file, once configuration changes are made to this file and other files in the /etc/mail/ directory, the following command processes those changes to the sendmail.cf and .db files in that directory:

```
make -C /etc/mail
```

The Postfix Alternative

Because of the complexity of open source sendmail software, many administrators prefer alternatives. One relatively popular open source alternative is Postfix. Methods for configuring Postfix vary. For example, on Ubuntu systems, the following command provides a number of prompts for configuring this MTA:

```
# dpkg-reconfigure postfix
```

Basic options include the following:

- **Internet site**—This configures Postfix to send and receive e-mail using SMTP.
- **Internet with smarthost**—This integrates the procmail MRA with Postfix.
- **Satellite system**—This sets up Postfix to transmit e-mail to a different system.
- **Local only**—This limits the Postfix configuration to users on the local system.

This section assumes you've selected Internet with smarthost, the most capable of the Postfix options. Once selected, you're prompted for the following information:

- The domain name covered by this Postfix server, such as example.org
- The relay host for outgoing e-mail, such as an SMTP server for an ISP (if this e-mail server is not a satellite system, you may choose to leave this blank)
- An e-mail address of the actual mail system administrator
- A list of domains and hostnames governed by this server
- Whether to force synchronous updates, which is unnecessary unless the Internet connection is not reliable
- A list of IP address networks governed by this server; a sample list would look like the following line:
127.0.0.0/8 192.168.0.0/24 10.100.0.0/16
- Whether to configure procmail for local delivery, to appropriate directories for local user accounts
- A limit of mailbox sizes, in bytes
- A local address extension (rarely changed)
- Whether to use IPv4, Internet Protocol version 6 (IPv6), or both for addressing

The choices you make here are written to the main.cf file in the /etc/postfix/ directory. Take a look at this file. You may want to change a few more things. For example, the following directive would deny information about the server to any malicious user:

```
smtpd_banner = "some email service"
```

If you want to secure this service with SSL or TLS certificates, the following directives provide a guide on the types of files to be created and their expected directory locations:

```
smtpd_tls_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem
smtpd_tls_key_file=/etc/ssl/private/ssl-cert-snakeoil.key
```

As noted by the `alias_maps` directive, e-mail aliases are stored in the same /etc/aliases file configured for sendmail. Finally, some systems limit Postfix access to the localhost system with the `inet_interfaces` directive. While you could specify a certain IP address on the local system, that IP address is attached to the message. That could cause problems for the message at a firewall that filters out messages from private IP addresses. In that case, you'd want to leave the default value for that directive:

```
inet_interfaces = all
```

Dovecot for POP and IMAP

Dovecot is an MDA that receives e-mail messages from SMTP services and collects them for delivery to MUAs such as Evolution, Outlook, and Thunderbird. The configuration file is dovecot.conf, which may be located in the /etc/ or /etc/dovecot/ directory, depending on the distribution.

While the default Dovecot configuration file is long, the steps needed to configure this file are pretty simple. First, the `protocols` directive lets you define which protocols are covered by this server:

```
protocols = imap imaps pop3 pop3s
```

If you've included secure protocols, be sure to retain the following directive:

```
ssl_disable = no
```

Of course, secure protocols require SSL support, made possible in Dovecot with the following directives:

```
#ssl_cert_file = /etc/ssl/certs/ssl-cert-snakeoil.pem
#ssl_key_file = /etc/ssl/private/ssl-cert-snakeoil.key
```

The protocol directive can help define the IP address of the network card to which Dovecot listens, along with any nonstandard port numbers. The following examples use nonstandard port numbers for the four addressed protocols on the noted IP address:

```
protocol imap
  { listen = 192.168.0.10:10143 ssl_listen = *:10993 }
protocol pop3
  { listen = 192.168.0.10:10110 ssl_listen = *:10995 }
```

More E-mail Services

These are just a few of the excellent e-mail services available. Other important SMTP services include Exim and Qmail. Related to security on e-mail services is spam.

Using Asterisk

Asterisk is the open source software implementation of a private branch exchange (PBX) program for telephone networks. As such, it is a VoIP solution for telephone communications. While much of Asterisk is released under an open source license, some Asterisk software is proprietary. The software available from <http://www.asterisk.org/> is developed by the Open Source Telephony Project. The project is sponsored by Digium, known as the Asterisk Company. The software available from Digium includes proprietary components. For more information on proprietary options, see <http://www.digium.com/en/>.

This section is focused on security issues related to the open source Asterisk software available from <http://www.asterisk.org/> or already loaded on Linux repositories for major distributions including Ubuntu and Fedora. With more than 60 configuration files, detailed configuration options for Asterisk are beyond what can be covered in this book.

When considering Asterisk, be aware that while an occasional dropped packet may not be a big deal, reliably speedy packet flow is important to keep conversations from becoming jumpy. Based on the characteristics of various protocols in the Transmission Control Protocol/Internet Protocol (TCP/IP) suite, the best fit for such communications is UDP. Asterisk communications, like other VoIP communications, use the **Session Initiation Protocol (SIP)** and the **Real-time Transport Protocol (RTP)** to send data through UDP packets. (SIP is an IP network protocol that is frequently used in VoIP communications. RTP is a standard packet format for VoIP and video communications.)

Basic Asterisk Configuration

Asterisk configuration files are typically installed in the /etc/asterisk/ directory. Most of the configuration filenames relate to different features. Asterisk configuration is centered around the dialplan, which can be a directory, a set of menu options, or even a group of channels. These are known as *contexts*, and can be used to implement features such as the following:

- **Access limits**—Limit access to long-distance calls to certain telephones.
- **Routing**—Route calls based on their extensions.
- **Autoattendant**—Set up automated greeting with message prompts.
- **Menus**—Set up automated menus for different departments.
- **Authentication**—Require passwords to access certain extensions.
- **Callback**—Return calls to route through the presumably cheaper Asterisk system.
- **Privacy**—Create blacklists for unwanted calls.
- **PBX multihosting**—Configure multiple PBXs on the same Asterisk server.
- **Daytime/nighttime**—Set up different actions at different hours.
- **Macros**—Create scripts for different functions.

These features are described at <http://www.voip-info.org/>.

While there are too many configuration files in the /etc/asterisk/ directory to detail in this section, they do include features that range from A to Z. For example, the adsı.conf file specifies the analog display services interface (ADSI) associated with messages on the small screens included with many common business telephones. The zaptel.conf file configures a telephony hardware driver application programming interface (API), now known as the Digium Asterisk Hardware Device Interface (DAHDI).

Security Risks with Asterisk

When configuring Asterisk, you must be aware of the legal responsibility associated with telephone systems. While such laws vary by jurisdiction, in general, you should not configure spoofing of other telephone numbers, you should not overload or otherwise disrupt service on other telephone networks, and you should not use Asterisk to intercept calls from other users.

Naturally, as a security professional, you need to make sure that these situations don't arise on your systems.

Denial of Service on Asterisk

Asterisk is different from other services in that it can support multiple connections between the same two IP addresses. And you want such connections, to enable multiple users from one company to talk with multiple users at another company simultaneously. As a result, the rules that you create to prevent denial of service (DoS) attacks on Linux are somewhat different from those associated with the standard service.

- **Virtual LANs (VLANs)**—The network partitions enabled by a **virtual LAN (VLAN)** can keep traffic separate. (A VLAN is a LAN created on the same physical network as another LAN. Because both LANs are separate and distinct, they are virtual.)
- **Appropriate firewall rules**—Voice traffic requires some priority. Firewall rules should be set with a type of service (TOS) that minimizes delay with a switch such as `-j TOS --set-tos Minimize-Delay`.
- **Limit packet floods**—Firewall rules should allow connections over the SIP and RTP protocols but should stop SYN flood attacks.

Asterisk Authentication

Without authentication, malicious users may be able to hijack a call on either end of the connection. Asterisk authentication is possible by a number of methods, including the Lightweight Directory Access Protocol (LDAP), the Remote Authentication Dial In User Service (RADIUS), and host-based authentication.

More Asterisk Security Options

Additional Asterisk security options use information at the Data Link Layer of the TCP/IP protocol suite. These security options include the following:

- **Dynamic Host Configuration Protocol (DHCP) snooping**—DHCP snooping limits network access to systems with known IP and hardware addresses.
- **Dynamic address resolution protocol (ARP) inspection**—This checks the binding between the IP address and media access control (MAC) hardware addresses, minimizing the risk of spoofed hardware addresses.
- **IP Source Guard**—This checks IP addresses against associated hardware addresses based on information stored in a DHCP lease file.
- **VLAN ACLs**—This reviews ACLs created for a VLAN to minimize the risk of spoofing.

Limiting Printers

The standard service for Linux printer configuration and management is the Common Unix Printing System (CUPS). It's a versatile system. While the default version of CUPS communicates using the Internet Printing Protocol (IPP) on port 631, it can be used to manage printers from a variety of other services, including Samba, the older Line Printer Next Generation (LPRng) service, HP JetDirect printers, and more. Microsoft operating systems support IPP. Apple, Inc. owns CUPS software and has maintained it under open source licenses. Because the Macintosh operating system is based on a relative of Linux, the Berkeley Standard Distribution, Apple has made CUPS the default print server for its operating systems.

CUPS is configured in files in the `/etc/cups/` directory. The main CUPS configuration file is `cupsd.conf`, the focus of the rest of this section.

Printer Administrators

One directive typically located early in the `cupsd.conf` file is `SystemGroup`. Groups that are listed after this directive are authorized to administer printers on this CUPS system. You can add users to the specified group in the `/etc/group` and `/etc/gshadow` configuration files. The groups may vary. On Red Hat systems, members of the

sys group are default print administrators. On Ubuntu systems, members of the lpadmin group are default print administrators.

Shared Printers

CUPS printers are local, unless remote access is allowed. Remote access starts with the **Listen** directive, which by default enables access through port 631 on all IP addresses. If you want to limit access to one of multiple network cards on a system, you could set up a directive like the following, where 192.168.0.10 is the IP address of the local network card that is allowed to receive CUPS communications:

```
Listen 192.168.0.10:631
```

If printers are shared through other systems such as Samba, ports for that service must also be open.

Shared printers can be shown on the local network with the following default directives, which are almost self-explanatory. They activate browsing of local printers and determine whether the **BrowseAllow** or the **BrowseDeny** directive is considered first. Either or both of these directives could be included with an IP address or domain name for a single host or a network. The final line, the **BrowseLocalProtocols** directive, specifies the use of printer browsing using CUPS and the DNS-based Service Discovery (DNSSD) protocol for CUPS print browsing.

```
Browsing On
BrowseOrder allow,deny
BrowseAllow all
BrowseLocalProtocols CUPS dnssd
```

Remote Administration

Remote administration of CUPS servers is based on the three **Location** stanzas that follow in the cupsd.conf file. The format of the three stanzas is quite similar. They are all confined by **Location** directives. They all specify host- or possibly user-based access. The difference is based on the label associated with each **Location** stanza.

The **Location** stanza supports access to locally configured printers. The default version of this stanza limits such access to the local system:

```
<Location />
    Order allow,deny
</Location>
```

You can extend that access to the local network with the following directive:

```
Allow @LOCAL
```

If desired, you can substitute an IP address and network mask for the target network, such as **from 192.168.0.0/255.255.255.0**. Systems on the noted network might also have access to administrative tools and configuration files, depending on the stanzas that follow. Alternatively, you could limit access even further with a couple of directives such as the following:

```
Allow from 192.168.0.10
Deny from 192.168.0.0/255.255.255.0
```

The second stanza, with the **<Location /admin>** directive, allows access to administrative areas, to allow a review of current print settings.

The third stanza defines users and systems allowed to add and modify printers on the local CUPS service. The **AuthType Basic** directive refers to the regular user-authentication database. **Require user @System** points to the **SystemGroup** directive. Because the @ in this case specifies groups, it means user members of the groups listed with the **SystemGroup** directive are CUPS print administrators.

```
<Location /admin/conf>
    AuthType Basic
    Require user @SYSTEM
    Order allow,deny
    Allow @LOCAL
</Location>
```

Other stanzas in this file specify other administrative functions such as the ability to move or delete print jobs. The **@OWNER** option, when included, allows the user owner of such print jobs to perform the noted print functions.

The CUPS Administrative Tool

Of course, you can administer CUPS from a local system. In fact, CUPS is one rare service where the tool preferred by most open source administrators is a Web-based tool. On a local CUPS system, you can access the tool by navigating to <http://127.0.0.1:631>. Despite the http:// in front of the URL, traffic to the CUPS administrative tool travels over port 631, which is the standard port for the IPP protocol.

Administrative actions within the CUPS Web-based tool lead to a prompt for a username and password. Authorized user accounts are members of the group defined in the **SystemGroup** directive in the cupsd.conf configuration file.

If remote administration is enabled, as described in the previous section, you can navigate to the domain name or IP address of the CUPS server, also using port 631.

Protecting Time Services

Time services on Linux rely on the NTP protocol. This section describes those directives in the /etc/ntp.conf file that can help secure a local NTP server. If an NTP server is overloaded in any way, it may not be a security issue per se. However, because it affects the reliability of the time seen from that server, it can have other consequences. For example, if the NTP server of an inventory system is not synchronized to the NTP server of a sales system, customers may try to order products that are out of stock.

The security-related directive in this file is **restrict**. Take a look at the directive as configured in the default ntp.conf file. The -4 and -6 correspond to IPv4 and IPv6 networking.

```
restrict -4 default kod notrap nomodify nopeer noquery
restrict -6 default kod notrap nomodify nopeer noquery
```

The other options affect security of the NTP server in the following ways:

- **default**—Access to all NTP clients is allowed by default.
- **kod**—This prevents attacks from kiss of death (KOD) packets.
- **notrap**—This denies access to the message trap service for logging.
- **nomodify**—This prevents other NTP servers from modifying the status of this server.
- **nopeer**—This prevents attachments from other NTP servers.
- **noquery**—This disallows queries of the NTP service.

Assuming you don't want to open this up as a public NTP server, the `default` option should not be used. However, some address is required. One option is the IP address of the local network. In addition, you may want to allow time queries, and allow other NTP servers, perhaps from remote networks administered by colleagues, to communicate with your NTP server. So the IPv4 restrict directive would change to the following:

```
restrict -4 192.168.0.0 mask 255.255.255.0 kod notrap nomodify
```

In a similar fashion, you could set up similar restrictions on an IPv6 network:

```
restrict -6 0011:838:0:1:: mask ffff:ffff:ffff:ffff::  
→ kod notrap nomodify
```

Obscuring Local and Network Services

To some extent, when you installed Linux, you made a choice for obscurity. But as you've seen so far in this book, obscurity can mean different things when applied to different services. Obscurity goes beyond the use of nonstandard ports for network communications.

Obscurity can mean using less-popular services. For example, you might choose Postfix over sendmail for an e-mail service. You could choose Bernstein's djbdns DNS server over BIND. You could even choose a different bootloader such as the trusted Grand Unified Bootloader (TrustedGRUB) or the Linux Loader (LILO).

Obscurity can also mean using different labels. For example, you could use the Samba `server string` directive to label that Microsoft-style file server as if it were something else. You could use the Postfix `smtpd_banner` directive to apply a label from sendmail to make attackers think they're dealing with that alternative SMTP server. You could include incomplete information in a login banner for the Secure Shell (SSH) or secure Telnet services. You could keep version information to a minimum in Apache with the `ServerTokens` directive.

The methods you can use to obscure local and network services are almost as diverse as the number of services available on Linux. This section is just a reminder to look at or even audit the services on your networks to make sure they use obscurity in the best way to secure local systems.

Best Practices: Networked Application Security

As with services in general, it is best to keep installed and enabled features to a minimum. This maxim is especially important with a complex service like the Apache Web server. With Apache, it's important to keep included modules to a minimum.

Of course, with Apache, you can do more to improve security. With the LAMP stack, MySQL can include security certificates. It's best if you choose only one scripting language from those associated with LAMP (Perl, Python, or PHP). SSL certificates can be configured for MySQL as well as for secure Apache Web sites.

The Squid Web proxy server can be used for two basic purposes. It can cache frequently accessed data, improving perceived Internet performance. It can also limit access to undesirable Web sites. Key Squid directives include `acl` and `http_access`.

Because DNS is a distributed database of domain names and IP addresses, many DNS servers are potentially vulnerable. DNS servers can be manipulated to redirect unsuspecting users to sites that benefit the attacker in some way. This may include spreading malicious software. To minimize such risks, you should disable recursion on authoritative name servers. TSIG keys can authenticate DNS servers that do exchange data. Using `dnssec` allows you to sign your zone files to help provide verification of the information being provided.

Linux supports a number of excellent MTAs, including open source sendmail, commercial Sendmail, Postfix, and more. MTAs can send data using the SMTP and submission protocols. Related to MTAs are MSAs such as Dovecot, which can receive e-mail using a number of regular and secure protocols. It's important to set up authentication for such services to prevent spammers from using them to distribute their malware.

Asterisk is the open source PBX solution. As such, it provides detailed VoIP configuration options for routing telephone calls through the Internet. But as a network solution, it may also be vulnerable to attacks such as DoS, DHCP snooping, and hardware address spoofing. To minimize risks such as unauthorized toll charges and spoofed phone calls, it's important to set up authentication with Asterisk.

CUPS is an open source print service that can be configured with a special group of print administrators. With the **Location** stanzas, you can configure CUPS to share printers, to share administrative views of different systems, and to allow remote administration.

The Linux time service is based on the NTP protocol. The basic **restrict** directive in the ntp.conf file can limit access by IP address. That option is superior to the default, which allows access to all systems. It can also be used to stop KOD packets and prevent access, attachments, or queries from other NTP servers.



CHAPTER SUMMARY

In this chapter, you examined some of the security-related features of a number of servers. Because the Apache Web server is complex, it's important to minimize the modules loaded with that service. The Squid Web proxy server can be used to speed access via caching and prevent access to undesirable Web sites.

While mail services such as sendmail, Postfix, and Dovecot have their own security issues, options such as SSL keys and domain checks can minimize those issues. As an open source PBX service, Asterisk has its own set of security issues. CUPS printers can be regulated to share and permit administration from remote systems. It can configure print administrators in special groups. It's important to keep time services secure to ensure the reliability of the time seen from these NTP servers.



KEY CONCEPTS AND TERMS

- Asterisk**
- Cache poisoning**
- Certificate authority (CA)**
- Challenge-response authentication mechanism (CRAM)**
- Domain Name System Security Extensions (DNSSEC)**
- Inter-Cache Protocol (ICP)**
- Java Cryptography Extension (JCE)**
- Linux/Apache/MySQL/P (LAMP) stack**
- Mail delivery agents (MDAs)**
- Mail retrieval agents (MRAs)**
- Mail submission agents (MSAs)**
- Mail transfer agents (MTAs)**
- Mail user agents (MUAs)**
- Multi-Processing Modules (MPMs)**
- Perl**
- PHP: Hypertext Preprocessor (PHP)**
- Python**
- Real-time Transport Protocol (RTP)**
- Recursive query**

Session Initiation Protocol (SIP)

Split DNS

Transition SIGnature (TSIG)

Virtual LAN (VLAN)

Web application firewall (WAF)

Zone files

Zone updates



CHAPTER 9 ASSESSMENT

1. Which of the following services is *not* part of the LAMP stack?
 - A. Linux
 - B. Apache
 - C. MySQL
 - D. Postfix

2. Which of the following commands sets a password for the MySQL administrative user?
 - A. mysqladmin -u admin password
↳ "newpassword"
 - B. mysqladmin -u root password
↳ "newpassword"
 - C. mysqladmin -u mysql password
↳ "newpassword"
 - D. mysqladmin -u user root
↳ "newpassword"

3. Which of the following commands removes the php5 module in Apache on an Ubuntu system?
 - A. a2enmod php5
 - B. a2rmmod php5
 - C. a2dismod php5
 - D. a2modprobe php5

4. What is the command that can create users and passwords for access to a Web directory in Apache?
 - A. htaccess
 - B. htpasswd
 - C. apachepass
 - D. apacheaccess

5. The server.csr file includes identifying information about your system.
 - A. True
 - B. False

6. Which of the following port numbers is associated with Squid?
 - A. 80
 - B. 3128
 - C. 443
 - D. 8080

7. Which of the following should *not* be included on a public DNS server?

- A. Mail server IP addresses
 - B. DNS server IP addresses
 - C. Web server IP addresses
 - D. Squid server IP addresses
- 8.** Which of the following commands reads all changes made to files in the /etc/mail/ directory for open source sendmail?
- A. `make -C /etc/mail`
 - B. `m4 /etc/mail`
 - C. `make /etc/mail/sendmail.mc`
 - D. `m4 /etc/mail/sendmail.mc`
- 9.** Which of the following open source sendmail directives is used to specify e-mail protocols?
- A. `define`
 - B. `DAEMON_OPTIONS`
 - C. `FEATURE`
 - D. `MAILER`
- 10.** Which of the following configuration files is most important for Postfix?
- A. `main.cf`
 - B. `master.cf`
 - C. `maps.cf`
 - D. `submit.cf`
- 11.** In Dovecot, if you want to activate both regular and secure POP3 and IMAP services, what options would you add to the `protocols` directive?
- A. `imap/ssl pop3/ssl`
 - B. `ssl_enable`
 - C. `imaps pop3s`
 - D. `tls_only`
- 12.** Which of the following is *not* a protocol closely associated with Asterisk?
- A. IPP
 - B. SIP
 - C. RTP
 - D. UDP
- 13.** Which of the following directives in the main CUPS configuration file specifies groups of users who are allowed to administer CUPS?
- A. `Lpadmin`
 - B. `Admin`
 - C. `System`
 - D. `SystemGroup`
- 14.** Which of the following NTP `restrict` options relate to logging?
- A. `kod`
 - B. `notrap`
 - C. `nopeer`
 - D. `noquery`
- 15.** Which of the following directives specify and can limit the information given about an Apache system?
- A. `banner`
 - B. `System`
 - C. `ServerTokens`

D. **server string**

CHAPTER

10 Kernel Security Risk Mitigation

T

HIS CHAPTER IS ALL ABOUT THE KERNEL, the essence of the Linux operating system. Because the kernel is highly configurable, it represents an opportunity to better secure your systems. Unfortunately, it also offers a way for attackers to take advantage of mistakes made by users who are less skilled at kernel configuration.

Because of the complexity of the kernel, most Linux professionals rely on kernels configured by reliable developers. Many such developers work for the corporate sponsors of distributions such as Red Hat, Novell, and Canonical. Corporate Linux kernel developers start with the work of volunteer developers affiliated with the Linux Kernel Organization. The **stock kernel** incorporates the latest features requested by users. It incorporates many of the features developed for specialty Linux distributions. Open source licenses make such development possible.

When choosing a kernel, you can take one of two basic approaches. You can stick with the kernel customized for a distribution or you can start with the stock Linux kernel. The developers behind both the Red Hat and Ubuntu distributions do an excellent job customizing their Linux kernel for enterprise-ready systems. To keep their customers happy, they generally have updates that address security issues available on a timely basis.

Whichever approach you choose, you can further customize and compile the kernel. It's best if this work is done on a developmental system. After the custom kernel and associated files are developed, compiled, and tested on the developmental system, you should be able to transfer those files to production systems.

Chapter 10 Topics

This chapter covers the following topics and concepts:

- Which options are available with distribution-specific functional kernels
- What the stock Linux kernel is
- How to manage security and kernel updates
- What development software is available for a custom kernel
- What Linux kernel-development tools are available
- How to build your own secure kernel
- How to increase security using kernels and the /proc/ filesystem
- What best practices are for kernel security risk mitigation

Chapter 10 Goals

When you complete this chapter, you will be able to:

- Choose the kernel best suited to your systems
- Customize options within the Linux kernel
- Compile the source code for a kernel, with patches

- Configure how the kernel works in real-time

Distribution-Specific Functional Kernels

The developers behind most Linux distributions want to make things as easy as possible for users. They provide installation programs to load and configure their distributions on a wide variety of hardware. They release software in packages, compiled in binary format. All the user has to do is load and install the package.

Kernels released by Linux developers vary by architecture. Because the kernel includes interfaces between hardware and software, kernels vary with that most fundamental architectural hardware component, the CPU. The developers behind some Linux distributions may also configure, customize, and release different binary kernels for different functional situations. For example, because administrators may not install a graphical user interface (GUI) on many servers, developers may choose to deactivate features associated with high-end graphics cards for a server kernel.

Kernels by Architecture

Linux kernels are available for a wide variety of architectures. Because developers at the Linux Kernel Organization release only the source code for the kernel, it's left to the developers behind a distribution to customize, compile, and test a kernel for each specific architecture. Such testing takes time and effort. At last count, Debian Linux had released production kernels for more than a dozen different architectures, including the following:

- Standard 32-bit**—The normal 32-bit PC architecture. This may be labeled with an i386, i586, or i686 in the package name.
- Standard 64-bit**—The normal 64-bit PC architecture. This frequently includes amd64 or x86_64 in the package name.
- Alpha**—The CPU developed by the original Digital Equipment Corporation, now owned by HP.
- Advanced RISC Machine (ARM)**—The CPU originally developed by the former Acorn Computers, Ltd., now licensed by a number of companies.
- Itanium**—A 64-bit CPU from Intel. In addition to **Itanium**, Other Intel 64-bit CPUs are available.
- Motorola 68000 (m68k)**—A 32-bit CPU developed by Motorola.
- PowerPC**—A 32-bit CPU developed by Apple, IBM, and Motorola.
- s390**—The IBM system 390 32-bit CPU.
- Sparc**—A 64-bit CPU originally developed by Sun Microsystems.

In addition, Red Hat builds its Enterprise Linux distributions for the IBM **iSeries** and **pSeries** CPUs. (**iSeries** is an IBM system that uses IBM Performance Optimization with Enhanced RISC [POWER] CPUs. **pSeries** is the IBM Reduced Instruction Set Computing [RISC] server and workstation product line designed for Unix systems.) However, it does not support any of the other CPUs except for standard 32-bit and 64-bit systems, along with 64-bit Intel Itanium systems on its older releases. Red Hat does not support the Itanium CPU on its newest releases.

One feature incorporated into different kernels on mostly older Linux distributions is symmetrical multiprocessing (SMP), used for systems with multiple CPUs. On most current Linux distributions, that feature has been incorporated into the main kernel.

On the other hand, Ubuntu limits its officially supported releases to two different architectures: the 32- and 64-bit Intel-compatible CPU. The package names of such binary kernels may be a bit misleading because they commonly include i386 for the 32-bit kernel and amd64 or x86_64 for the 64-bit kernel. Such kernels work on systems with both Intel and Advanced Micro Devices (AMD) CPUs. That focus enables Ubuntu to customize kernels for different purposes.



NOTE

Although Ubuntu developers build a kernel that emulates the ARM architecture, Canonical, Ubuntu's corporate sponsor, does not officially support that kernel.

Kernels for Different Functions

The developers behind most Linux distributions create kernels for different functions. Those who use Xen for virtual machines should be familiar with the specialized kernel for that purpose. Among the most popular Linux distributions, Ubuntu builds kernels for a wide array of functions. [Figure 10-1](#) shows one list of such Ubuntu kernels. Some of the packages shown are meta packages. A **meta package** is a Linux package that refers to other packages. For example, the `linux-image-generic` package is automatically linked to the package built for the latest production Linux kernel release.

[Table 10-1](#) describes the meta packages shown in [Figure 10-1](#). [Figure 10-1](#) is based on selected output from the `apt-cache search linux-image` command. It does not include debug or development kernels. Such kernels are not suitable for production, in part because they have not been tested for security. It specifies the meta package associated with the kernel, as the installation of a meta package automatically installs the latest version of the kernel so described. Of course, if something other than the latest version of a kernel is best for a system, you should be able to choose that package by version number. A list of such earlier kernel version packages should be shown when you run that `apt-cache search linux-image` command.

```
kilroy@stevedallas:~$ apt-cache search linux-image
alsa-base - ALSA driver configuration files
linux-image-3.13.0-24-generic - Linux kernel image for version 3.13.0 on 64 bit x86 SMP
linux-image-3.13.0-24-lowlatency - Linux kernel image for version 3.13.0 on 64 bit x86 SMP
linux-image-extra-3.13.0-24-generic - Linux kernel extra modules for version 3.13.0 on 64 bit
x86 SMP
linux-image-extra-virtual - Transitional package.
linux-image-generic - Generic Linux kernel image
linux-image-generic-lts-quantal - Generic Linux kernel image
linux-image-generic-lts-raring - Generic Linux kernel image
linux-image-generic-lts-saucy - Generic Linux kernel image
linux-image-generic-lts-trusty - Generic Linux kernel image
linux-image-lowlatency - lowlatency Linux kernel image
linux-image-server - Transitional package.
linux-image-virtual - This package will always depend on the latest minimal generic kernel im
age.
linux-virtual - Minimal Generic Linux kernel and headers
linux-image-generic-pae - Transitional package
linux-image-lowlatency-pae - Transitional package
linux-image-3.4.0-3-goldfish - Linux kernel image for version 3.4.0 on Android touch emulatio
n
linux-image-goldfish - Linux kernel image for the goldfish kernel.
linux-image-3.13.0-27-generic - Linux kernel image for version 3.13.0 on 64 bit x86 SMP
linux-image-3.13.0-27-lowlatency - Linux kernel image for version 3.13.0 on 64 bit x86 SMP
linux-image-3.13.0-29-generic - Linux kernel image for version 3.13.0 on 64 bit x86 SMP
linux-image-3.13.0-29-lowlatency - Linux kernel image for version 3.13.0 on 64 bit x86 SMP
```

FIGURE 10-1

Ubuntu kernel package options.

TABLE 10-1 Typical Ubuntu Linux kernel packages.

PACKAGE	DESCRIPTION
<code>linux-image</code>	Installs the latest available generic Linux kernel image

linux-image-386	Installs the latest available Linux kernel for 32-bit systems
linux-image-generic	Installs the latest available Linux kernel for 64-bit systems
linux-image-server	Installs the latest available Linux kernel configured for servers
linux-image-lowlatency	Installs the latest available Linux kernel configured to provide fast performance
linux-image-goldfish	Installs a version of the Linux kernel designed for Android devices. Goldfish is a virtual processor specified by Google.

In general, these meta packages are linked to kernels customized for 32-bit systems. Some of these meta packages may not work for 64-bit systems unless the associated kernel is also designed to work for such systems.

There are fewer functional kernel packages for Red Hat Enterprise Linux. But Red Hat does build different kernels for regular and Xen-based systems.



NOTE

The package names associated with the Linux kernel may change from time to time and may vary by distribution. For example, Ubuntu kernel packages used to have the word kernel in the package name. Red Hat kernel packages still include kernel in their package names. For the latest information, use a search command like `yum search kernel` or `apt-get search linux`.

The Stock Kernel

The stock kernel is the generic kernel released by the developers of the Linux Kernel Organization at <http://kernel.org/>. The leader of that organization is **Linus Torvalds**, who developed the first kernel in 1991. A Linux kernel version was released late that year. In 1992, Torvalds adapted the GNU General Public License (GPL) for all Linux kernel releases. Torvalds is still the lead developer of the Linux kernel, coordinating the efforts of thousands of other developers.

The stock kernel is sometimes known as the mainline or vanilla kernel. Many Linux kernel developers work on releases that may be stable or unstable. The current maintainer of the stable release tree is **Greg Kroah-Hartman**. Other key developers are responsible for different kernel subsystems. To understand the sequence of Linux kernels, you must understand the Linux kernel numbering system.

Kernel Numbering Systems

Production versions of most current Linux stock kernels include four numbers, such as 3.19.0-21, in the following format: *majorversion.majorrevision.updateversion-patchnumber*. The following is a brief explanation of these kernel numbers:

- **Majorversion**—Specifies a version with drastic changes from a previous kernel
- **Majorrevision**—Associated with major changes to the kernel
- **Updateversion**—Released for important updates and new features to the kernel
- **Patchnumber**—Associated with bug fixes and security updates to the kernel

**NOTE**

The numbering system for the stock kernel varies slightly from that of many major distributions. The developers behind a distribution incorporate bug fixes and security updates on their own. Therefore, they leave out the fourth number associated with such bugs and security.

Kernels with new update versions are released approximately every three months. The first release of an update version kernel has three numbers, such as 3.19.0. The first patch of that update version is release 3.19.0-1.

The developers behind some of the most important Linux distributions may release their enterprise-ready distributions with slightly older kernels. This supports a longer period of testing with a stable kernel. When new features are incorporated into later kernels, many developers **backport** those features. That is, they implement features from newer kernel versions into an older kernel version. It's not part of the older version, but the feature has been put into the source code so it can be compiled there.

Production Releases and More

Developers associated with the Linux Kernel Organization maintain production kernels associated with multiple update version numbers. When administrators build a kernel to a certain update version, they may choose to stick to that update version for a while so they do not have to test their systems for compatibility with the new features associated with later updates.

Linux kernel developers release security patches for a number of different kernel update versions. A patch is just a small section of source code; the patch file indicates where in the original source code the updated code should be placed. Patches for Linux are designated for the version of the kernel to which they are meant to apply. As of June 2015, the latest patch for the 3.x line of kernels is patch-3.16.8. You would apply this to the 3.16 kernel using the Linux patch utility to get your 3.16 kernel up to the latest code without having to acquire a whole new kernel source distribution. This is especially useful if you have previously built a kernel because you don't have to replace the configuration file and recompile all the source. You will need to recompile only the source that has changed because of the patches.

Generally, if there is an important security issue, it's most efficient to wait for the latest production release. Security issues get special attention not only by the Linux Kernel Organization but also by the developers behind major Linux distributions.

Download the Stock Kernel

To use a stock Linux kernel, start with the version available from the Linux Kernel Organization at <http://kernel.org>. You may choose to start with the latest stable kernel or some older kernel that is still maintained. The stock kernel is available as source code. Before you can put it into effect on a system, you must compile it, as discussed later in this chapter.

Stock kernels are normally available in a compressed `tar` command archive. This is compressed with the Lempel-Ziv Gzip or the Burrows-Wheeler block sorting text compression algorithm with Huffman coding. Kernels compressed in these formats have `.tar.gz` and `.xz` extensions, respectively.

You can download the compressed archive stock kernels from <http://www.kernel.org/pub/> or <ftp://ftp.kernel.org/pub/>. Kernel packages are typically placed in the `/usr/src/` directory.

If you have doubts about the integrity of the source code, download the associated compressed archive file with the `.sign` extension. For example, if you have downloaded the `linux-3.19.8.tar.gz` file, you can also download the `linux-3.19.8.tar.gz.sign` file. Instructions associated with this signature are available at <http://www.kernel.org/signature.html>. Once downloaded, you can unpack the source code into a subdirectory with something similar to one of the following commands. (Substitute the version number of the downloaded Linux kernel for 3.19.8.1.)

```
# tar xzf linux-3.19.8.1.tar.gz
# tar xjf linux-3.19.8.1.tar.bz2
```

The `tar` command is one of those rare Linux commands for which there is no dash in front of command switches. The `Z` switch uncompresses files with a `.tar.gz` extension; the `J` switch uncompresses files with a `.tar.bz2` extension.

If you need to know more about the changes made to a production kernel, download the ChangeLog file associated with the release version.

Stock Kernel Patches and Upgrades

If a downloaded, uncompressed, and unarchived stock Linux kernel is available in an appropriate directory, you do not need to repeat the process for the next release. All you need to do to update the source code is to download the appropriate patch.

Two different types of patches are available. For example, to update source code from version 3.19.8.1 to 3.19.8.2, the appropriate patch is available in the package file named `patch-3.19.8.2.gz` or `patch-3.19.8.2.bz2`. Once copied to the `/usr/src/` directory, you can apply the patch with one of the following commands:

```
# zcat patch-3.19.8.2.gz | patch -p0
# bzcat patch-3.19.8.2.bz2 | patch -p0
```

It's also possible to patch kernel source code between update versions. For example, you can use the `patch-2.6.35.gz` file to update Linux kernel source code from version 3.19.8 to 2.6.35. However, you can only apply patches consecutively. For example, to update source code from 3.9.18 to 2.6.35, you would need to apply both the `patch-3.19.8.gz` and `patch-2.6.35.gz` updates.



WARNING

It's generally a bad idea to use a generic patch on kernel source code released for a specific distribution. For example, source code for the latest Red Hat Enterprise Linux 5 kernels includes backports from the latest Linux kernel releases. A patch applied to that source code may overwrite those backports, removing features you may need.

Managing Security and Kernel Updates

Kernel security can be divided into two interrelated areas. First, there are security issues that arise related to the stock Linux kernel. Second, there are security issues that arise related to the Linux kernel as built by the developers of a distribution.

When updating a kernel, it's best to make sure any existing working kernel is retained on your systems. If there is a problem with the updated kernel, those systems will still be bootable and available with the older working kernel. In general, that is easiest to put into effect with the binary kernels built by the developers of a distribution. Because such kernels are already in binary format, you do not need to compile them. You can install them directly on the distribution. However, custom kernels work better for many systems. You can optimize custom kernels to load more quickly. More importantly, custom kernels can exclude options such as modules that may be loaded by attackers to present security risks.

Stock Kernel Security Issues

The developers associated with the Linux kernel communicate primarily via messages on mailing lists. To keep up to date with the latest developments in the Linux kernel, you can visit <http://lkml.org/>, the Linux kernel

mailing list. However, monitoring this list is time-consuming. Often, there are hundreds of messages between developers on a daily basis. One site that is more focused on generic security issues is available at <http://www.linuxsecurity.com/>. For a digest of information, see the latest Linux advisory watch available from this Web site.

Linux kernel developers cannot test updates against all configurations. Any updates they create may lead to problems. Therefore, it's important to test all updated kernels before installing them on production systems. This warning also applies to security updates.

In limited situations, you might choose not to install a new kernel for a security update. For example, if there is a security issue related to Bluetooth hardware, it's important only if you have Bluetooth components installed on your systems. If you do not, you might choose not to install that update. Be aware, however, that a security issue that affects one system could easily affect a related system such as network hardware.

Distribution-Specific Kernel Security Issues

The Linux kernel as built for specific distributions such as Red Hat or Ubuntu includes a number of features that differ from the stock kernel. With thousands of options, the differences are too numerous to mention.

Although distribution-specific kernels are built from the stock Linux kernel, the security issues related to distribution-specific kernels may vary. As with the stock kernel, however, you may not always want to update a distribution-specific kernel. Some updates do not relate to security issues. Other updates may include hardware that is not installed on your systems. In all cases, it's a risk to update a Linux kernel. Because developers behind Linux distributions cannot possibly test all configurations, it's up to you to test distribution-specific kernel updates in production.

The standard command to update packages on Ubuntu systems is `apt -get update`. By default, it excludes later versions of the Linux kernel from an update.

The standard command to update packages on Red Hat systems is `yum update`. It does not exclude later versions of the Linux kernel unless the `--exclude=kernel-*` switch is included in the command:

```
# yum update --exclude=kernel-*
```

Installing an Updated Kernel

Be careful when implementing a new kernel. Make sure to use appropriate install commands and switches with the package-management commands for your distribution. This ensures the existing kernel remains in place, still available to boot local systems if the updated kernel does not work for whatever reason. In contrast, if you upgrade a kernel, the process overwrites any existing kernel. If the new kernel does not work for any reason, you will have to go through a rescue process to reinstall the older working kernel. This is a time-intensive process.

There are two basic methods to install an updated kernel. The first method relates to update commands such as `yum` and `apt -get` to download and install kernels from existing repositories on remote systems. On Ubuntu systems, you can run the `apt -get install` command with the name of the updated kernel. If you want to install the latest version of the kernel, you can specify the appropriate meta package shown in [Table 10-1](#). For example, the following command installs the latest version of the Ubuntu kernel customized for virtual machines:

```
# apt-get install linux-image-virtual
```

In the same fashion, you can use the name of the Red Hat kernel to install the latest version of that kernel. For example, the following command installs the latest version of the 64-bit kernel:

```
# yum install kernel-x86_64
```

If you have downloaded the actual updated kernel package, you can use the `rpm` or `dpkg` commands. In both cases, the `-i` switch installs the target package.

Binary kernel packages built by the developers of Linux distributions should automatically update applicable boot-loader files such as /etc/lilo.conf and /boot/grub/menu.lst. If properly updated, these files should include stanzas for both the current and the new kernel. If you have set up custom features for an existing kernel in the boot loader, you may need to modify the stanza for the new kernel to include those same messages.

Note that some distributions regulate the number of kernels that can be installed on a system. The kernel and initial RAM disk files associated with a kernel can be quite large. Many distributions, including Red Hat, configure the /boot directory on a small separate partition. By default, Red Hat Enterprise Linux specifies a size of 100 MB for this partition. If you have compiled a custom kernel, the initial RAM disk for that kernel can easily reach several dozen megabytes, which can limit the number of kernels that are practical to install on a system.

Development Software for Custom Kernels

To recompile a kernel, you will need to install packages associated with C language libraries and compilers, kernel header files, preprocessors, and related development tools. If you want to use menu-based kernel customization tools, additional packages are required. When installing such software, keep careful records of the packages installed.

When installed on production systems, the software associated with compiling the kernel may present risks. It may also enable attackers to compile more dangerous malware from within your systems. If kernel development software is installed on a production system, be sure to remove that software after the new kernel is operational.

Red Hat Kernel Development Software

You can use the packages listed in this section to customize and build a kernel on Red Hat systems. While the same basic functionality is required to build kernels on other Linux systems, at least the package names will vary. [Table 10-2](#) lists the key packages.

TABLE 10-2 Red Hat kernel-development packages.

PACKAGE	DESCRIPTION
binutils	Binary utilities
cpp	C language pre-processor
gcc	GNU's not Unix (GNU) C language compiler
glibc-headers	Kernel header files
glibc-devel	Kernel development files associated with C language libraries
ncurses	New curses library for updating character screens; supports the <code>make menuconfig</code> command

The listed packages support the terminal menu-based configuration options associated with the `make config` and `make menuconfig` commands when run in the directory with the kernel source code. If you prefer graphical kernel-customization tools, you will need packages associated with the Tool command language (Tcl) and the associated cross-platform Tk toolkit. Depending on the type of graphical customization tool, you may also need development tools associated with the Qt toolkit, GLib headers, and the Glade interface designer. A number of these packages work with the Perl programming language, so associated packages should also automatically be installed as dependencies if you use the `yum` command to install these packages.

Ubuntu Kernel Development Software

The packages required to customize and compile the Linux kernel on an Ubuntu system vary by release. You can install these tools with the following command:

```
# apt-get install build-essential
```

These commands install packages with equivalent functionality to those described in the previous section for Red Hat systems. The specifics may vary in terms of package names but the necessary utilities are the same across both systems. You will need a compiler, a set of libraries, and make tools as a starting point for building kernels.

As with the noted Red Hat packages, the installed packages support the terminal menu-based configuration options associated with the `make config` and `make menuconfig` commands when run in the directory with the kernel source code. If you prefer to customize systems with graphical tools, you will need tools and toolkits similar to those described for Red Hat systems.

Kernel-Development Tools

This section discusses the options available to customize a kernel. It's based on stock kernel source code from the developers at the Linux Kernel Organization, discussed earlier. To customize a kernel, you also need the kernel-development tools just described.

Although this section focuses on the ncurses-based kernel-customization console tool, the basic options described here also apply to the available command-line or graphical tools. You could also start with the source code released by the developers of a distribution. Instructions for downloading and preparing that distribution-specific source code are available later in this chapter.

Before Customizing a Kernel

Before you customize a kernel, apply any patches. If you are starting from the source code of a stock kernel, download desired patches from the Linux Kernel Organization and apply them to the unpacked source code with the `zcat` or `bzcat` commands.

Navigate to the directory with the kernel source code. If it's the stock kernel, that directory will have a name like `/usr/src/linux-source-3.9.18/`. (The version number may vary.) If you are running these commands on an Ubuntu system, you can do so with `sudo`-based privileges. Alternatively, from a Red Hat system, these commands are most effectively run from the root administrative account. These first commands clean out any remaining object files and remove any existing kernel configuration in the local directory. If you have just installed the kernel source code, these commands are not necessary.

```
# make clean
# make mrproper
```

The next step is to select a baseline. You could accept the baseline inherent in the source code. Alternatively, you could select one of the configuration options available in the `configs` or `arch/x86/configs` subdirectory. If the system architecture is not a standard 32- or 64-bit Intel/AMD CPU, substitute for `x86/` accordingly. A third option is to start with the configuration of the current system, available in the `/boot/` directory in the `config-`uname -r`` file, where ``uname -r`` is the version number of the currently loaded kernel. This option is more appropriate if you are starting with distribution-specific kernel source code. If you use this option with the stock kernel, the results are less predictable. If you start with an existing `config-*` file, be sure to copy that to the hidden `.config` file in the directory with the source code for the kernel.

Start the Kernel Customization Process

Navigate to the directory with the kernel source code. From this directory, there are several methods for customizing a Linux kernel before it's compiled.

You could open the aforementioned .config file in a text editor and change settings in that file directly. That option is considered impractical for all but the most knowledgeable Linux developers, however.

Alternatively, you could run the `make config` command. It's a simple script. As shown in [Figure 10-2](#), it prompts you to make choices for each option associated with the Linux kernel. Although the questions are straightforward, there are literally thousands of options associated with the Linux kernel. If you miss an option to be changed, you must press Ctrl+C to abort the operation and start over by running the `make config` command again. This makes it impractical to customize the kernel with the `make config` command.

```
kilroy@hodgepodge:/usr/src/linux-source-3.19.0/linux-source-3.19.0$ sudo make config
HOSTCC scripts/kconfig/conf.o
HOSTLD scripts/kconfig/conf
scripts/kconfig/conf --oldaskconfig Kconfig
#
# using defaults found in /boot/config-3.19.0-21-generic
#
*
* Linux/x86 3.19.8 Kernel Configuration
*
64-bit kernel (64BIT) [Y/n/?]
*
* General setup
*
Cross-compiler tool prefix (CROSS_COMPILE) []
Compile also drivers which will not load (COMPILE_TEST) [N/y/?]
Local version - append to kernel release (LOCALVERSION) []
Automatically append version information to the version string (LOCALVERSION_AUTO) [N/y/?]
Kernel compression mode
> 1. Gzip (KERNEL_GZIP)
  2. Bzip2 (KERNEL_BZIP2)
  3. LZMA (KERNEL_LZMA)
  4. XZ (KERNEL_XZ)
  5. LZO (KERNEL_LZO)
  6. LZ4 (KERNEL_LZ4)
choice[1-6?]:
Default hostname (DEFAULT_HOSTNAME) [(none)]
Arbitrary version signature (VERSION_SIGNATURE) [Ubuntu 3.19.0-21.21-generic 3.19.8]
Support for paging of anonymous memory (swap) (SWAP) [Y/n/?]
System V IPC (SYSVIPC) [Y/n/?]
POSIX Message Queues (POSIX_MQUEUE) [Y/n/?]
```

FIGURE 10-2

Customizing a kernel with `make config`.

The practical method for customizing a kernel without a GUI uses ncurses-based configuration menus. In the directory with the kernel source code, run the `make menuconfig` command. You should see a menu similar to that shown in [Figure 10-3](#).

As the functionality of the Linux kernel evolves, some of the options in the kernel configuration menus will vary. However, most of the options described in the sections that follow are relatively stable.

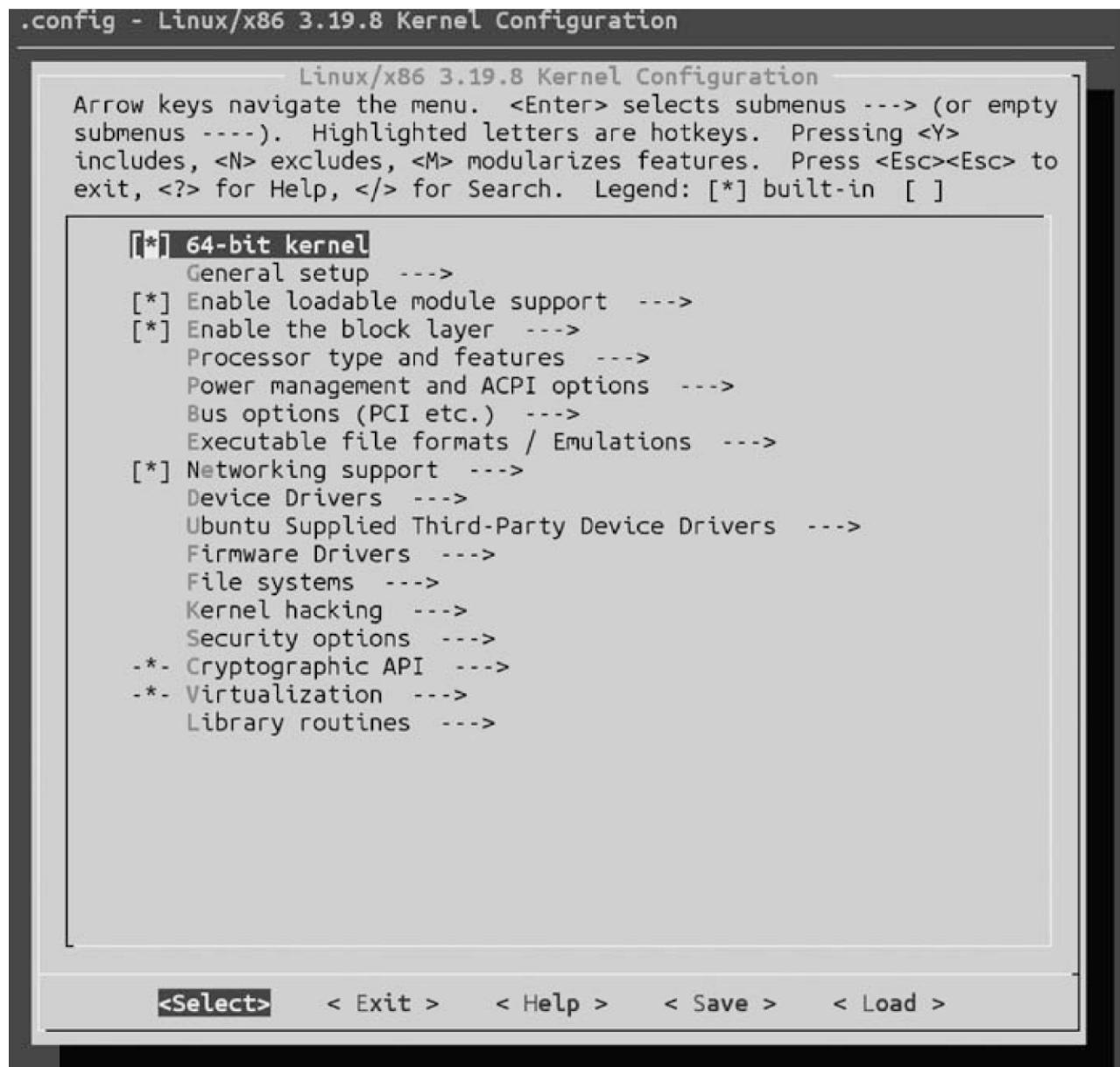


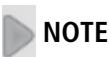
FIGURE 10-3

Customizing a kernel with `make menuconfig`.

Kernel-Configuration Options

Even if you never reconfigure a kernel, do explore the available options. An understanding of these options can help you grasp the strengths and weaknesses of the Linux kernel, including those related to security. The options described in the sections that follow are based on the kernel-configuration menu associated with the stock version of the Linux kernel, version 3.9.18, as released by the Linux Kernel Organization.

Many kernel-configuration options interact with others. The wrong choices can lead to a kernel that fails to boot a system. If you are in doubt about an option, stick with the default.



NOTE

The wording associated with some menu options may vary slightly. For example, newer kernels have replaced the Loadable Module Support submenu with the Enable Loadable Module Support submenu.

Some kernel-configuration options are labeled as experimental and should be treated as such. In other words, they are not proven in a production environment.

Help options are available for most Linux kernel configuration options. The Help menu normally specifies the actual descriptive kernel option, such as `CONFIG_HOTPLUG_PCI` for hot-pluggable Peripheral Component Interconnect (PCI) controllers.

If you configure a kernel with fewer features, there are fewer opportunities for attackers to break into your systems. As long as you are careful to maintain system backups and existing kernels, some trial and error can help you configure a more secure kernel.

64-Bit Kernel

Enabling this will build a 64-bit kernel. Otherwise, you will build a 32-bit kernel.

General Setup

General setup options run the gamut from kernel compression modes to the use of swap space to initial RAM disk support. In most cases, you should not change any options in this section.

Enable Loadable Module Support

The default options automatically load appropriate module devices for existing and newly installed hardware. Because Linux distributions rely on a modular kernel, you should generally not change that basic functionality. One new potentially interesting feature in the stock kernel relates to module verification, which uses GNU Privacy Guard (GPG) keys to verify modules. It's a common option for kernels built for Red Hat systems.

Enable the Block Layer

Block-layer kernel settings are associated with larger files in the terabyte range. Without such settings, common Linux filesystems such as the third and fourth extended filesystems (`ext3` and `ext4`) will not work.

Processor Type and Features

With the development of multi-core CPU processors with different features, this menu includes a number of relatively new options. The stock kernel does not enable paravirtualized guest support, which is essential if you want to configure virtual machines with paravirtualized shortcuts for hardware. With **paravirtualization**, software is installed on the guest operating system such that the system is aware that it's operating inside a virtual system. If this helper software is not installed, the guest cannot function. In some cases, the operating system must be modified to make this approach work. This is not a terribly common virtualization solution.

Power Management and ACPI Options

Because the power management on current computers uses the Advanced Configuration and Power Interface (ACPI), the title of this section is somewhat redundant. It includes power-management options associated with systems that may hibernate or be suspended to RAM. Some administrators believe that hibernation files may themselves present a security risk. To that end, you may consider disabling hibernation in the kernel. Just remember, there is no guarantee that any change to a kernel configuration will lead to a working kernel.

The source code associated with older kernels may include options associated with advanced power management in this section.

Bus Options

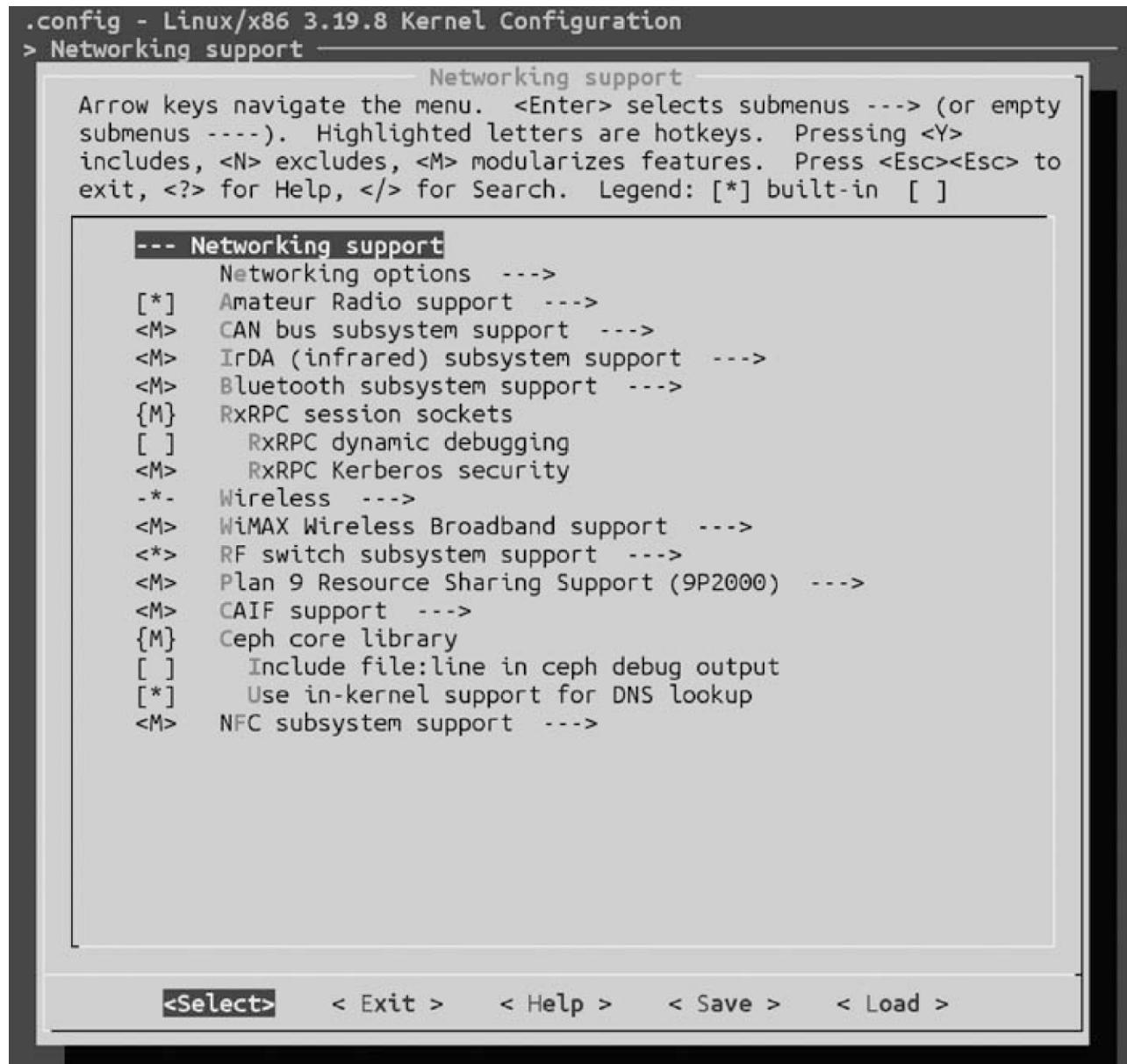
Most computer systems include PCI cards. Support for such cards is typically enabled in this part of the kernel configuration. The Linux kernel normally includes support for a variety of PCI and similar devices. For example, if your systems can connect to PC card and PCI hot-pluggable hardware, consider disabling such options in the kernel—unless, of course, you’re actually planning to use such hardware on these systems.

Executable File Formats/Emulations

Current Linux kernels normally include support for binary commands and scripts in executable linkable format (ELF). This section lists options for such formats. Unfortunately, current kernels do not support fine-grained control of ELF options.

Networking Support

The networking-support menu in the Linux kernel is extensive. Besides those options relating to standard and wireless networking, options in this section can allow or disable support for infrared, Bluetooth, Worldwide Interoperability for Microwave Access (WiMAX), and other networking hardware. The Networking Options submenu shown in [Figure 10-4](#) also provides extensive options related to the kinds of network packets with which the kernel can work. Packets associated with the Dynamic Host Configuration Protocol (DHCP), the Bootstrap Protocol (BOOTP), and the Reverse Address Resolution Protocol (RARP) are recognized in the default Linux stock kernel. You can disable that capability in this menu.

**FIGURE 10-4**

Kernel networking options.

Device Drivers

The device-driver section is extensive in the Linux kernel. It's worth spending some time in this area. If you want to disable access to certain devices, you can do so here. Any device that is disabled in the kernel is not available to an attacker. For example, if you disable parallel port devices, an attacker will not be able to copy files to parallel port drives.

This section is divided into a number of different areas. The following list (and yes, it's a long list) highlights those sections that include options related to physical devices. These sections are presented in the order shown from the stock Linux kernel, version 3.9.18.

- **Memory Technology Devices**—Supports flash cards, RAM chips, and similar devices.
- **Parallel Port Support**—Associated with devices connected to parallel ports.
- **Plug and Play Support**—Enables Linux configuration of plug-and-play devices.

- **Block Devices**—Works with nonstandard hard disks and network block devices.
- **ATA/IDE/MFM/RLL**—Associated with older hard drives based on Advanced Technology Attachment (ATA), Integrated Drive Electronics (IDE), Modified Frequency Modulation (MFM), and run length limited (RLL) standards. The option is deprecated on newer kernels.
- **SCSI Device Support**—Includes drivers associated with Small Computer System Interface (SCSI) connections.
- **Serial ATA and Parallel ATA Drivers**—Specifies drivers associated with Serial Advanced Technology Attachment (SATA) and Parallel Advanced Technology Attachment (PATA) systems, mostly hard drives.
- **Multiple Devices Driver Support**—Includes options for redundant array of independent disk (RAID) arrays and logical volume management (LVM) systems. Can be set to automatically detect RAID arrays or set linear RAID or RAID versions 0, 1, 4, 5, 6, and 10. Be aware that this is software RAID; the acronym is misleading because it relates to independent partitions, not disks.
- **Fusion MPT Device Support**—Supports fusion message passing technology (MPT) devices. Associated with higher-speed SCSI adapters.
- **IEEE 1394 (FireWire) Support**—Specifies modules associated with FireWire hardware, associated with standard 1394 of the Institute of Electrical and Electronics Engineers (IEEE).
- **i2O Device Support**—Works with intelligent input/output (I2O) devices.
- **Macintosh Device Drivers**—Enables the use of Macintosh input devices such as a mouse. Some Linux distributions may be installed on the latest Macintosh systems.
- **Network Device Support**—Specifies the network card drivers available for Linux.
- **Input Device Support**—Configures support for basic input devices such as keyboards, mice, touchpads, and more.
- **Character Devices**—Associated with byte-stream devices such a virtual terminals, serial ports, some video cards, and more.
- **i2C Support**—Works with devices that use inter-integrated circuits (I2C), associated with low-speed peripherals.
- **SPI Support**—Used by Serial Peripheral Interface (SPI) systems that communicate with sensors and flash memory.
- **PPS Support**—Related to pulse-per-second (PPS) signals associated with some antennas from global positioning system (GPS) devices.
- **GPIO Support**—Enables connections to general-purpose input/output (GPIO) devices.
- **Dallas's 1-Wire Bus**—Allows communication over single-pin devices.
- **Power Supply Class Support**—Enables modules associated with the monitoring of devices, such as uninterruptable power supplies (UPSs) and batteries.
- **Hardware Monitoring Support**—Related to devices that monitor the status and health of the system, such as temperature monitors.
- **Generic Thermal Sysfs Driver**—Supports an alternative hardware monitor for thermal management. May control cooling devices.
- **Watchdog Timer Support**—Enables monitoring devices that automatically reboot a nonresponsive system.
- **Sonics Silicon Backplane**—Supports connections to devices with the Sonics Silicon Backplane bus.
- **Multifunction Device Drivers**—Enables access to an array of different multi-function devices.
- **Voltage and Current Regulator Support**—Allows dynamic support of voltage regulators. If the system supports it, such regulators may conserve power and battery life.
- **Multimedia Support**—Configures support for multimedia devices such as video for Linux.
- **Graphics Support**—Enables support for selected graphical devices.
- **Sound Card Support**—Configures hardware support for various Linux sound schemes.

- **HID Devices**—Supports access to human interface devices (HIDs) such as keyboards and mice, if they are not already supported in other driver sections.
- **USB Support**—Allows connections to a wide array of universal serial bus (USB) devices.
- **Ultra Wideband Devices**—Enables connections to ultra wideband devices, a low-power radio technology also associated with wireless USB and future wireless Bluetooth and IEEE 1394 hardware.
- **MMC/SD/SDIO Card Support**—Allows connections to various multimedia cards (MMCs), such as secure digital (SD) and secure digital input output (SDIO) cards.
- **Sony MemoryStick Card Support**—Includes cloned software that enables connections to Sony MemorySticks.
- **LED Support**—Related to devices that control light-emitting diode (LED) lights other than those present on keyboards.
- **Accessibility Support**—Enables access to selected devices that help people with disabilities access computers.
- **InfiniBand Support**—Allows connections through high-performance InfiniBand data connections.
- **EDAC (Error Detection and Correction) Reporting**—Enables logging of core system hardware errors.
- **Real Time Clock**—Supports a variety of clock hardware.
- **DMA Engine Support**—Enables support for direct memory access (DMA), which can bypass the CPU.
- **Auxiliary-Display Support**—Sets up access for auxiliary displays.
- **Userspace I/O Drivers**—Supports a small variety of hardware that supports I/O through user space systems.
- **Staging Drivers**—Includes drivers that do not meet normal standards of Linux kernel developers. Avoid this option if possible.
- **X86 Platform Specific Device Drivers**—Includes drivers primarily associated with brand-specific laptop functionality.

Firmware Drivers

The Linux kernel includes support for a few specialized firmware drivers, normally associated with boot systems such as the basic input/output system (BIOS).

Filesystems

This section includes integrated and modular support for a wide variety of local and network-based filesystems. One way to disable access to certain filesystems such as the Network File System (NFS) and Samba is to disable it in the kernel as configured in this section.

Kernel Hacking

Most kernel-hacking options are designed for kernel developers who want to see what happens with the kernel in detail in basic operation.

WARNING

Be aware that this section includes support for some experimental features. You should not enable such options on production systems. This is true about any experimental feature.

Security Options

As these kernel options are directly related to this book, you will want to analyze them in detail. Many of these options relate to features associated with Security Enhanced Linux (SELinux). These options are presented in the order shown from the stock Linux kernel, version 3.9.18. They include the applicable kernel setting in case you want to search further online.

- **Enable Access Key Retention Support**—Supports the retention of authentication tokens and access keys. According to IBM, you can also use such keys to cache authentication data. Related to the CONFIG_KEYS option.
- **Enable the /proc/keys File by Which Keys May Be Viewed**—Access keys are made visible in the /proc/keys file. Related to the CONFIG_KEYS_DEBUG_PROC_KEYS option.
- **Enable Different Security Models**—Required to implement security models such as SELinux and Application Armor (AppArmor). Related to the CONFIG_SECURITY option.
- **Enable the Securityfs Filesystem**—Associated with Trusted Platform Modules (TPMs). Related to the CONFIG_SECURITYFS option.
- **Socket and Networking Security Hooks**—Supports the use of security modules to regulate the use of sockets and networks. Related to the CONFIG_SECURITY_NETWORK option.
- **XFRM (IPSec) Networking Security Hooks**—Supports per-packet controls associated with policies for communications through Internet Protocol Security (IPSec) tunnels, associated with the transformer (XFRM) transformation framework. Related to the CONFIG_SECURITY_NETWORK_XFRM option.
- **Security Hooks for Pathname Based Access Control**—Supports access controls based on pathnames. Related to the CONFIG_SECURITY_PATH option.
- **Enable Intel (r) Trusted Execution Technology (Intel (r) TXT)**—Associated with the Intel trusted boot module. Related to the CONFIG_INTEL_TXT option.
- **(65536) Low Address Space for LSM to Protect from User Allocation**—Protects the first 64 MB of RAM from allocation as user space for **Linux security modules (LSMs)**. (An LSM is a framework for security support within the Linux kernel, associated with mandatory access control.) Related to the CONFIG_LSM_MMAP_MIN_ADDR option.
- **NSA SELinux Support**—Supports the use of SELinux. Related to the CONFIG_SECURITY_SELINUX option.
- **NSA SELinux Boot Parameter**—Allows you to disable SELinux during the boot process with the `selinux=0` directive. Related to the CONFIG_SECURITY_SELINUX_BOOTPARAM option.
- **NSA SELinux Runtime Disable**—Supports the disabling of SELinux before related policies are loaded on the current system. Related to the CONFIG_SECURITY_SELINUX_DISABLE option.
- **NSA SELinux Development Support**—Supports the development of additional features on SELinux. Related to the CONFIG_SECURITY_SELINUX DEVELOP option.
- **NSA SELinux Avc Statistics**—Collects the use of SELinux permissions, in access vector cache (AVC) format. Supports the use of SELinux. Related to the CONFIG_SECURITY_SELINUX_AVC_STATS option.
- **NSA SELinux Checkreqprot**—Determines whether SELinux checks for the protection requested by a specific application. Related to the CONFIG_SECURITY_SELINUX_CHECKREQPROT_VALUE option.
- **NSA SELinux Maximum Supported Policy Format Version**—Supports the use of SELinux. Related to the CONFIG_SECURITY_SELINUX_POLICYDB_VERSION_MAX option.
- **Simplified Mandatory Access Control Kernel Support**—Enables the use of **simplified mandatory access control kernel (SMACK)** support, an alternative to SELinux and AppArmor. Related to the CONFIG_SECURITY_SMACK option.
- **TOMOYO Linux Support**—Enables the use of TOMOYO, an alternative to SELinux and AppArmor. Related to the CONFIG_SECURITY_TOMOYO option.
- **Default Security Module (SELinux)**—Sets the default mandatory access control option as SELinux. Related to the DEFAULT_SECURITY_SELINUX option.

The options in this section provide kernel support for several dozen cryptographic options, associated with the cryptographic application programming interface (API). Such options are classified into a number of sections, including the following:

- **Crypto Core or Helper**—Includes cryptographic algorithms and related tools.
- **Authenticated Encryption with Associated Data**—Specifies options primarily related to IPsec.
- **Block Modes**—Supports a variety of block ciphers.
- **Hash Modes**—Includes options for hashing messages and more.
- **Digest**—Specifies a variety of digest algorithms.
- **Ciphers**—Includes a variety of cipher algorithms.
- **Compression**—Specifies support for various compression algorithms.
- **Random Number Generation**—Supports random number generation tools for cryptographic modules.
This is beyond the support provided by the /dev/random device.

Library Routines

Options in the library routines section include compression options in the kernel.

Building Your Own Secure Kernel

The previous section included information on some of the many different ways to customize the Linux kernel. That is not enough, however. You need to know how to compile that kernel. This section summarizes the steps required to build a custom kernel.

Details frequently vary by distribution. If details are not described in this section, they were already covered earlier in this chapter. These steps include the following:

1. Install required development tools.
2. Download and unpack the source code.
3. Navigate to the directory with the source code.
4. You may want to make use of an existing configuration file from local configs/ subdirectory or a config-* file in the /boot/ directory. Copy this starting configuration file to the .config file in the directory with the Linux source code.
5. Open a kernel configuration tool with a command like `make menuconfig` and make desired changes.
6. Compile the kernel based on the new custom configuration.
7. Install the new kernel. If required, create and install a new initial RAM disk that matches the new kernel.
8. Make sure the boot loader is updated with stanzas for the existing and new custom kernels along with matching initial RAM disks.
9. Test the result. Try rebooting into the new kernel. If it does not work, boot into the previously working kernel.



WARNING

When configuring a custom kernel, be sure to preserve the files associated with an existing kernel. If the custom kernel leads to an unbootable or otherwise troubled system, you can use the existing kernel to boot that system.

Download Kernel Source Code

There are two basic choices for kernel source code. You could start with the source code released by the developers at the Linux Kernel Organization. Alternatively, you could start with the source code released by the developers of your target distribution. Source code from the Linux Kernel Organization was discussed earlier in this chapter. The process for downloading the source code for a distribution is a bit different.

Download Ubuntu Kernel Source Code

For Ubuntu systems, the easiest way to download the source code for the current kernel is to run the following command:

```
# apt-get install linux-source
```

When running this command, you will notice that the source code has three numbers. As of June 2015, these numbers are 3.9.18. In other words, Ubuntu starts with the noted update version before any patches. It may add its own patches before releasing the associated source code.

Download Red Hat Kernel Source Code

Although Red Hat does not release its binary packages under open source licenses, it does release its source code under those licenses. That source code is publicly available from [ftp://ftp.redhat.com](http://ftp.redhat.com). On that server, source code packages are available in one of the following two directories, associated with Red Hat clients and servers:

- /redhat/linux/enterprise/7Client/en/os/SRPMS/
- /redhat/linux/enterprise/7Server/en/os/SRPMS/

A test of the kernel source code from these two directories with the `diff` command suggests that they are the same packages. While the kernel source code packages on the two directories are the same, the general list of packages on these directories differ.

When you download the source code, make sure it is of the correct version number. To verify the version number of the currently loaded kernel, run the `uname -r` command.

After the source code package is downloaded, it can be written to an appropriate subdirectory of /usr/src/ with the following command. (The `uname -r` command in back quotes is replaced with the version number of the currently loaded kernel.)

```
# rpm -i kernel-`uname -r`.src.rpm
```

This installs the basic source code to /usr/src/redhat/ subdirectories. The next step is to build the source code. Navigate to the /usr/src/redhat/SPECS/ subdirectory. Then run the following `rpmbuild` command. The `-bp` switch unpacks the source code with applicable patches. The `--target` switch specifies the platform. The `uname -m` command with the back quotes is replaced with the architecture, such as x86_64 or i386.

```
# rpmbuild -bp --target=`uname -m` kernel-3.10.spec
```

For Red Hat Enterprise Linux 7, the command unpacks the source code into the following directory: /usr/src/redhat/BUILD/kernel-3.10/linux-2.10.`uname -m`/. As of this writing, the version number for Red Hat Enterprise Linux 7 is 3.10.0.

Install Required Development Tools

If you have not already installed the development tools described earlier in this chapter, do so now. You will need them to compile and customize the kernel. The actual packages to be installed vary by distribution and release.

Navigate to the Directory with the Source Code

As described earlier in this chapter, the directory with the source code varies by distribution. The directory may have a name like `/usr/src/linux-3.9.18/`. On Red Hat systems, it may be named something like `/usr/src/redhat/BUILD/kernel-3.10.0/linux-3.10.0.x86_64/`. Once in that directory, you will need a baseline configuration in the `.config` file in the local directory. Standard architecture-specific configurations are available in a `configs/` subdirectory.

Alternatively, you can copy the configuration of the currently loaded kernel from the `/boot/config-`uname-r`` file. Or you can process the current configuration with the `make oldconfig` command.

If you have compiled the kernel before, it's best to make sure any files that remain from previous custom kernels are clean with the following command:

```
# make clean
```

Open a Kernel-Configuration Tool

It's time to configure the kernel. Open a kernel-configuration tool. For example, the `make menuconfig` command opens the console-based tool described earlier in this chapter. Although other equivalent kernel-configuration tools are available, they are functionally equivalent to this console-based tool.

When changes are made, you will be prompted to save those changes. When those changes are successfully saved, you will see it in the output to the `ls -l .config` command, which should reflect the current date and time.

Compile the Kernel with the New Custom Configuration

Before compiling the kernel, open the file named `Makefile` in the local directory. Look at the `EXTRAVERSION` directive. It specifies the suffix to be added to the newly compiled kernel. You may want to add an identifier to that directive—for example, your name or some other identifier:

```
EXTRAVERSION = -prep.wubble1
```

Note that it takes time to compile a kernel. On slower systems, the process may take a couple of hours. Make sure there is at least a couple of gigabytes of free space on the volume where the kernel is being compiled. If the volume fills up before the process is complete, you will have to delete unneeded files and start again.

If the process stops with an error, pay attention to the last messages before the error. There may be an incompatible option in the custom configuration. The commands required to compile a kernel vary slightly depending on the distribution and release.



TIP

The developers behind the Linux kernel are working to make it easier to compile the kernel. Options for the `make` command are described in the `Makefile` file in the directory with the kernel source code.

Compile a Kernel on Ubuntu Systems

On Ubuntu systems, the following command should create the new kernel as a Debian-style package in the local directory.

```
# make-kpkg buildpackage --initrd kernel_image
```

Compile a Kernel on Red Hat Systems

On Red Hat systems, the command is even simpler—but this is one time when it's important to be in the root administrative account. Compiling a kernel on a Red Hat system, even with `sudo` privileges, can lead to errors.

From the root administrative account, the following command creates a kernel and sets up a custom RPM in the /usr/src/redhat/RPMS/`uname-m`/directory:

```
# make rpm
```

Compile a Stock Kernel

When compiling a stock kernel, the distribution-specific commands may not always work. If they do not work, and you are confident the problem is not related to some custom configuration error in the kernel, run the following command to compile that stock kernel:

```
# make
```

Although not strictly necessary, you should also run the following command. Without it, Linux may not have access to modules that are compatible with the custom kernel:

```
# make modules_install
```

When the process associated with these commands is complete, the following command should set up the new kernel and initial RAM disk in the /boot/ directory. It should make needed changes to the boot loader:

```
# make install
```

Install the New Kernel and More

The distribution-specific commands to compile a kernel should create appropriate binary packages in the directories just described. You should be able to run an associated package-management command such as **rpm** or **dpkg** to install the newly customized and compiled kernel. When you do, be sure to use the **install** switch to avoid overwriting any current and presumably working kernel.

If successful, you should find both the new custom kernel and the existing working kernel in the /boot/ directory. Alternatively, if you have run the **make install** command with the stock kernel, the new kernel should be installed in the /boot/ directory.

If there is no initial RAM disk included with the new custom kernel, you can create one with the **mkinitrd** command. For example, the following command should create an appropriate Red Hat initial RAM disk in the /boot/ directory:

```
# mkinitrd /boot/initrd-3.10.0-229.wubble1.img 3.10.0-229.wubble1
```

The format for initial RAM disks varies by distribution.

Check the Boot Loader

Finally, you need to make sure the new custom kernel has an appropriate and separate stanza in the boot-loader configuration file. Assuming the system is configured to boot from a Linux boot loader, the difference between stanzas for the custom and existing kernel should be limited to the version number of the kernel and initial RAM disk files.

Test the Result

It's time to test the result. For this, you may want to change the boot-loader configuration file to make it easier to bring up the associated boot menu. You can then reboot the system, open the boot-loader menu, and select the new custom kernel.

If the boot process is successful, be sure to test the new kernel with appropriate conditions associated with production systems. When you are satisfied with the results, you can change the boot-loader configuration file to make it difficult to bring up the boot menu and boot directly into the newly customized kernel.

If the boot process or any of the other tests are not successful, you should be able to reboot the system into the existing working kernel.

If the new kernel is a success, you should be able to use the same custom kernel on other systems. When you are satisfied with the results, you can transfer the kernel and associated files to production systems.

Increasing Security Using Kernels and the /proc/ Filesystem

Besides creating a custom kernel, you can do more with it to help secure Linux systems. One method uses the dynamic kernel options documented in the /proc/ directory filesystem. Although the /proc/ directory includes information on detected hardware, currently running processes, memory, and more, actual kernel options are shown in /proc/sys/ subdirectories. For example, the status of the /proc/sys/net/ipv4/ip_forward file determines whether Internet Protocol version 4 (IPv4) packets are forwarded through the system.

Most kernel configurable files in the /proc/sys/ directory are Boolean. In other words, they have a value of 0 or 1. If the file is active, its value is 1. Therefore, to activate IPv4 forwarding on a system, you would write the number 1 to the /proc/sys/net/ipv4/ ip_forward file. One way to do so is with the following command:

```
# echo "1" > /proc/sys/net/ipv4/ip_forward
```

The /proc/ directory filesystem is dynamic. Changes to these files do not survive a reboot. To make such changes permanent, you must add appropriate options to the /etc/sysctl.conf file. The corresponding directive in that file is as follows:

```
net.ipv4.ip_forward = 1
```

Note that there is no reference to **proc** or **sys** in that directive, as all kernel configuration options reside in the /proc/sys/ directory.

The following sections are based on a presentation by Steve Grubb of Red Hat at the 2008 Red Hat Summit. In that presentation, Grubb recommended kernel configuration changes in the areas described. The focus of these sections is IPv4 networking. If you have also enabled Internet Protocol version 6 (IPv6) networking, do not forget to make parallel changes to matching directories in the /proc/ filesystem and matching directives in the /etc/sysctl.conf file.

Don't Reply to Broadcasts

Normally, a system will respond to a **ping** command addressed to the broadcast address of a network. For example, the following command should receive responses from all systems with IP addresses on the 192.168.0.0/24 network:

```
$ ping -b 192.168.0.255
```

An attacker who can identify the IP address of a system can start attacking that system. An attacker who can identify the IP addresses of all systems on a network can attack those systems simultaneously. To prevent that, you activate the following option:

```
/proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
```

As suggested by the name, the option ignores broadcast messages using Internet Control Message Protocol (ICMP). It works because the **ping** command uses ICMP. The corresponding option in /etc/sysctl.conf is as follows:

```
net.ipv4.icmp_echo_ignore_broadcasts = 1
```

This option does not prevent responses to targeted **ping** commands, just broadcasts. One advantage is that it also prevents a smurf attack, in which a broadcast is sent using a **spoofed** or faked source address of the intended target computer.

Protect from Bad ICMP Messages

ICMP messages can be **mangled**, or spoofed, resulting in ICMP messages that do not conform to regular standards. To deny attackers as much information as possible, you can activate the following option:

```
/proc/sys/net/ipv4/icmp_ignore_bogus_error_responses
```

As suggested by the name, the option ignores bogus ICMP messages. The corresponding option in /etc/sysctl.conf is as follows:

```
net.ipv4.icmp_ignore_bogus_error_responses = 1
```

Protect from SYN Floods

A SYN flood is a denial of service (DoS) attack based on a rapid succession of SYN packets requesting synchronization with a target system. A SYN cookie can help prevent dropped connections, which can lead to additional SYN messages. You can activate the SYN cookie option in the following file: /proc/sys/net/ipv4/tcp_syncookies. As suggested by the name, the option activates SYN cookies on Transmission Control Protocol (TCP) connections. The corresponding option in /etc/sysctl.conf is as follows:

```
net.ipv4.tcp_syncookies = 1
```

Activate Reverse Path Filtering

Reverse path filtering can help regulate traffic through routers, limiting such traffic to packets destined for networks associated with that router. In other words, with reverse path filtering, a message destined for IP address 10.10.10.10 that is sent to a router associated with IP network addresses 192.168.0.0 and 192.168.1.0 is dropped.

You can activate reverse path filtering on the following files:

- /proc/sys/net/ipv4/conf/all/rp_filter
- /proc/sys/net/ipv4/conf/default/rp_filter

As suggested by the directory path, it's a configuration option for default and all networks. The corresponding options in /etc/sysctl.conf are as follows:

```
net.ipv4.conf.all.rp_filter = 1
net.ipv4.conf.default.rp_filter = 1 10
```

Close Access to Routing Tables

If an attacker can change the routing tables on your systems, that person can redirect your users to his or her systems. If the attacker can spoof a gateway system, that person can change the routing tables on other systems on the network. To that end, if that attacker could redirect traffic intended for an internal corporate page to a remote site that requests Social Security numbers, that could be trouble. You can prevent outsiders from changing your routing tables by deactivating the following files:

- /proc/sys/net/ipv4/conf/all/accept_redirects

- /proc/sys/net/ipv4/conf/all/secure_redirects
- /proc/sys/net/ipv4/conf/default/accept_redirects
- /proc/sys/net/ipv4/conf/default/secure_redirects

As suggested by the directory path, it's a configuration option for default and all networks. The corresponding options in /etc/sysctl.conf are as follows:

```
net.ipv4.conf.all.accept_redirects = 0
net.ipv4.conf.all.secure_redirects = 0
net.ipv4.conf.default.accept_redirects = 0
net.ipv4.conf.default.secure_redirects = 0
```

Avoid Source Routing

Source routing allows a user to specify the route a packet takes from a system to a destination. If you want to retain control over how packets are sent over a network, it's best to disable this option for network users. You can deactivate such options by deactivating the following files:

- /proc/sys/net/ipv4/conf/all/accept_source_route
- /proc/sys/net/ipv4/conf/default/accept_source_route

As suggested by the directory path, it's a configuration option for default and all networks. The corresponding options in /etc/sysctl.conf are as follows:

```
net.ipv4.conf.all.accept_source_route = 0
net.ipv4.conf.default.accept_source_route = 0
```

Don't Pass Traffic Between Networks

Yes, some systems should be configured as routers to pass information between networks, such as a private local network and the Internet. Such options should be activated only on routers, however—not on regular systems. To deactivate a system as a router, deactivate the following files:

- /proc/sys/net/ipv4/ip_forward
- /proc/sys/net/ipv4/conf/all/send_redirects
- /proc/sys/net/ipv4/conf/default/send_redirects

As suggested by the directory path, it's a configuration option for default and all networks. The corresponding options in /etc/sysctl.conf are as follows:

```
net.ipv4.ip_forward = 0
net.ipv4.conf.all.send_redirects = 0
net.ipv4.conf.default.send_redirects = 0
```

Log Spoofed, Source-Routed, and Redirected Packets

In computer networking, packets normally have source and destination addresses. By definition, one of those addresses should be associated with the local network. It's relatively easy to set up spoofed addresses, however. Packets with addresses that should not be possible are known as **Martian packets** (as in, packets from Mars). To see if and how often attackers may be trying to spoof addresses, you want to check log files. The following files, if active, make sure information on Martian packets is logged:

- /proc/sys/net/ipv4/conf/all/log_martians

- /proc/sys/net/ipv4/conf/default/log_martians

As suggested by the directory path, it's a configuration option for default and all networks. The corresponding options in /etc/sysctl.conf are as follows:

```
net.ipv4.conf.all.log_martians = 1
net.ipv4.conf.default.log_martians = 1
```

Best Practices: Kernel Security Risk Mitigation

Perhaps the key tool in a battle to secure a Linux system is the kernel. This makes sense because the kernel is the essence of the Linux operating system. The developers behind major Linux distributions do an excellent job creating and updating secure kernels in binary format, ready for quick installation. Different distributions may configure and test Linux kernels for one or over a dozen different architectures.

The developers behind different distributions may focus their efforts on other levels of functionality. For example, Ubuntu releases different distributions for servers, guest virtual machines, Xen hosts, and even the Amazon enterprise cloud. All Linux distributions start with the stock kernel released by the developers behind the Linux Kernel Organization, led by Linus Torvalds.

Releases of the stock kernel follow a *majorversion.majorrevision.updateversion.patchnumber* format. New update versions are released approximately every three months. Several stable versions are maintained simultaneously. Administrators who want to update their stock kernel can use patches released by the Linux Kernel Organization.

The Linux kernel mailing list at <http://lkml.org/> can help you monitor security issues related to the stock kernel. The developers behind Linux distributions maintain their own mailing lists that highlight security issues related to custom kernels built for their releases. Because not all kernel updates relate to security issues, you may not want to install all updates. If you do install a kernel update, it's important to preserve the currently installed kernel in case the update causes trouble for your systems.

If you want to customize the Linux kernel, you can start with the stock kernel or the source code released by the developers behind a Linux distribution. You will also need development software to customize the kernel and compile the customized code.

Before customizing a kernel, take care to clean the directories associated with the source code with commands like `make clean`. Consider starting with a baseline configuration either in a configs/ subdirectory or the /boot/ directory. Even if you prefer to leave this work to the developers behind a Linux distribution, review the options available for the Linux kernel with something like the console-based tool accessible with the `make menuconfig` command.

Custom kernel configurations are saved to the hidden .config file in the directory with the kernel source code. Custom extensions can be configured in the Makefile file in that directory. The commands required to compile the custom kernel vary by distribution. Other commands may be used for stock kernels. Once complete, make sure the custom kernel does not overwrite the currently running kernel in the /boot/ directory and any associated boot-loader configuration files.

When you are satisfied with a kernel, you can do more to secure a system in the /proc/ filesystem. Dynamic kernel configuration options are available in /proc/sys/ subdirectories, in Boolean files. Some of these options can protect a system from DoS attacks, spoofed source and destination addresses, remote reconfiguration of routing tables, and more. You can configure the same options in the /etc/sysctl.conf file to make sure desired changes survive a reboot.



CHAPTER SUMMARY

As the center of the Linux operating system, the Linux kernel lies at the core of Linux security. Although most users start with a kernel created by the developers behind a distribution, you can also start with the stock kernel created by developers from the Linux Kernel Organization. Whichever one you choose, it's important to keep that kernel up to date with appropriate patches and updates. Not all updates relate to security, however. Because kernel updates can affect the functionality of your systems, care is required.

You can customize kernels based on source code released either from the Linux Kernel Organization or by developers behind a selected Linux distribution. With the right development tools, you can customize the options associated with a kernel and then compile them into a binary format. You need to take care to preserve any existing kernel, as problems with a newly customized kernel could otherwise leave a system unbootable.



KEY CONCEPTS AND TERMS

Backport

Greg Kroah-Hartman

iSeries

Itanium

Linus Torvalds

Linux security modules (LSMs)

Mangled

Martian packets

Meta package

Paravirtualization

pSeries

Simplified mandatory access control kernel (SMACK)

Spoofed

Stock kernel

TOMOYO

CHAPTER 10 ASSESSMENT

1. Which of the following features is no longer associated with a separate kernel on many Linux distributions?
 - A. Virtual machine hosts
 - B. Xen
 - C. SMP
 - D. Servers

2. When an updated kernel is released with a security update, which of the following numbers in the kernel is changed?
 - A. Major version
 - B. Minor version
 - C. Update version
 - D. Patch number

3. Which of the following commands can be used to unpack and uncompress a stock kernel in .xz format?
 - A. tar xzf

- B. tar xjf
 - C. tar xbf
 - D. tar xuf
4. The Web site associated with the Linux Kernel Organization is _____.
5. Which of the following actions should you *not* take when implementing a new kernel?
- A. Upgrade
 - B. Install
 - C. Patch
 - D. Compile
6. Which of the following directories contain a file with the configuration of the kernel that is currently running on the local system?
- A. /usr/src/redhat/BUILD/kernel-`uname -r`/linux-`uname -r`/
 - B. /usr/src/linux-`uname -r`/configs/
 - C. /usr/src/linux-`uname -r`/
 - D. /boot/
7. Which of the following commands starts a console-based menu-driven tool for customizing the kernel?
- A. **make config**
 - B. **make menuconfig**
 - C. **make xconfig**
 - D. **make gconfig**
8. Which of the following kernel options is *not* related to mandatory access controls?
- A. SELinux
 - B. TOMOYO
 - C. Cryptographic API
 - D. AppArmor
9. Name the section of the kernel configuration tool related to formats such as ext2 and ext3.
- A. Filesystems
 - B. Device Drivers
 - C. Networking
 - D. Security Features
10. In what configuration file can you customize the filename of the compiled kernel?
- A. make
 - B. config-`uname -r`
 - C. .config
 - D. Makefile
11. When a new kernel is compiled and installed, what file should contain different stanzas to two different kernels available during the boot process? (Select two.)
- A. /boot/grub/menu.lst
 - B. /etc bootloader
 - C. /etc/lilo.conf
 - D. /usr/src/linux-`uname -r`/.config
12. If you see the **net.ipv4.icmp_echo_ignore_broadcasts = 1** option in the /etc/sysctl.conf file, which file contains that Boolean option?
- A. /proc/net/ipv4/icmp_echo_ignore_broadcasts
 - B. /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts

- C. /proc/net/sys/ipv4/icmp_echo_ignore_broadcasts
 - D. /proc/icmp_echo_ignore_broadcasts
13. Which of the following directives in /etc/sysctl.conf should be set to 0 to deactivate routing on the local system?
- A. net.ipv4.ip_forward
 - B. net.sys.ipv4.ip_route
 - C. net.ipv4.ip_routing
 - D. net.sys.ipv4.ip_source

PART THREE

Building a Layered Linux Security Strategy

CHAPTER 11**Managing Security Alerts and Updates****CHAPTER 12****Building and Maintaining a Security Baseline****CHAPTER 13****Testing and Reporting****CHAPTER 14****Detecting and Responding to Security Breaches****CHAPTER 15****Best Practices and Emerging Technologies**

CHAPTER

11 Managing Security Alerts and Updates

C

OMPUTER SECURITY IS A CONSTANT BATTLE. From the moment a security alert is released, there is a race between attackers who look for vulnerable systems and administrators who install updates to address the vulnerability. This is tempered by the risks associated with some updates because developers cannot test updates against every possible hardware and software configuration. You may need to make your own judgment on whether to install an update based on the associated bug report. You might base your judgment on your own testing of the updates on your own system configurations.

The battle starts with the distribution. The developers behind Linux distributions do an excellent job releasing security updates on a timely basis. Because they release these updates as binary packages in structured repositories, it is relatively easy to install them with any needed dependent packages. The developers behind a distribution also incorporate updates from the developers of applications such as the [OpenOffice.org](#) suite. Linux administrators have allies in this process, including organizations as powerful as the Institute for Security and Open Methodologies (ISECOM), the U.S. National Security Agency (NSA), and the open source community.

While fewer instances of malware directly affect Linux, Linux can carry some malware when served to other operating systems such as Microsoft Windows. Additionally, there are instances of malware that affect Linux systems. As a result, antivirus and anti-malware systems are important aspects of a Linux security strategy.

The best way to keep a Linux system up to date depends on the number of systems being updated, as well as whether you need to create specialized updates. Some commercial update managers enable you to push updates to one or several systems simultaneously. Both open source and commercial updates are based on repositories maintained by the developers behind a distribution. If you want, you can create customized repositories for an enterprise or even just a local area network (LAN).

Chapter 11 Topics

This chapter covers the following topics and concepts:

- How to keep up with distribution security
- How to keep up with application security
- Which antivirus options are available for Linux systems
- How to use bug reports
- How security works in the open source world
- How to decide whether to use automatic updates or analyzed alerts
- How to manage Linux patches
- What the options for update managers are
- What commercial update managers are
- What open source update managers are
- What best practices are for security operations management

Chapter 11 Goals

When you complete this chapter, you will be able to:

- Keep up with Linux security issues
- Select an appropriate antivirus system
- Get help from the open source community
- Choose an update-management system

Keeping Up with Distribution Security

While Linux is Linux, many security issues are distribution specific. In this section, you will review various ways to keep up with the latest security issues for each distribution. This section reviews security lists associated with Red Hat and Ubuntu and allied distributions.

Although Red Hat Enterprise Linux (RHEL) is the stable enterprise-ready distribution released by Red Hat, some enterprises choose other distributions based on Red Hat software. Two of the major options are Fedora Core, the open source developmental test bed for future Red Hat Enterprise releases; and CentOS, based on the Red Hat Enterprise Linux source code released under the GNU General Public License. CentOS is binary-identical to Red Hat Enterprise Linux. This means that although it may use different badging (icons, wallpapers, and so on), all the executables are identical to those available in RHEL.



NOTE

The URLs referenced here are subject to change over time.

The challenges presented in this section should motivate you to standardize systems on one Linux distribution.

Red Hat Alerts

Red Hat is more than just Red Hat Enterprise Linux (RHEL). While RHEL is the flagship Red Hat distribution, it's not the only option associated with Red Hat software. CentOS Linux will have the same problems as those of RHEL, so any RHEL alert will be relevant to CentOS.

Red Hat Enterprise Linux

For RHEL, two methods are available for monitoring security advisories:

- **The public Red Hat security advisories (RHSAs) mailing list**—Anyone with an e-mail address can sign up for the **Red Hat security advisories (RHSAs)** mailing list at <https://www.redhat.com/mailman/listinfo/rhsa-announce>. An equivalent Really Simple Syndication (RSS) feed is also available.
- **The list of errata on the Red Hat Network (RHN)**—Anyone with a subscription to the **Red Hat Network (RHN)** can access the errata, with associated security advisories. The information on both lists is essentially the same. Anyone with a paid subscription can download the binary packages from the RHN that can address security issues. Moreover, you can configure any system connected to the RHN to automatically download and install such updates. More options for pulling and pushing such updates are described later in this chapter.

If you find a security issue with an RHEL system, Red Hat encourages you to contact its Security Response Team at secalert@redhat.com. Per <https://access.redhat.com/security/team/contact>, e-mails sent to that address are kept confidential if you so desire.

In their efforts to provide enterprise-level support, Red Hat provides security updates for its RHEL releases for at least seven years.

CentOS Linux

The developers of CentOS Linux use the RHEL source code released by Red Hat under open source licenses. As such, the CentOS Linux distribution is binary-identical to RHEL. This means all the binaries on CentOS are identical to those in RHEL.

Most security issues with CentOS software also apply to Red Hat software. There are differences, however. To avoid trademark issues, CentOS modifies some released RHEL source code to omit RHEL trademarks.

Many Linux adherents use CentOS, as there is no charge for access to that distribution in its binary format. Of course, CentOS cannot offer Red Hat support for its distribution. In addition, there is commonly a time lag between the release of a Red Hat security update and a CentOS security update. CentOS does update its distribution when updated RHEL source code is publicly released, but it takes a little while for the volunteers behind CentOS to compile the source code and test it on the CentOS **rebuild** distribution. (A rebuild is a Linux distribution built from the source code released by another distribution. For example, because CentOS uses Red Hat source code, CentOS Linux is a rebuild of Red Hat Enterprise Linux.)



NOTE

The choice is yours: Are the costs associated with a subscription to RHEL worth the faster security updates?

CentOS has a mailing list for security advisories at <http://lists.centos.org/mailman/listinfo/centos-announce/>. Equivalent information is also made available on CentOS message boards. CentOS advisories and updated packages typically lag behind those associated with Red Hat by a few days. But of course, CentOS is more freely available than Red Hat.

Fedora Core Linux

Fedora Core Linux is the test bed for RHEL. New versions of RHEL are released approximately every one to three years. In addition, RHEL gets enterprise-level support and the benefits associated with the RHN. On the other hand, new versions of Fedora Core Linux are released approximately every six months.

Fedora Core Linux is typically released with the latest stable version of the Linux kernel. However, because developers at the Linux Kernel Organization create security patches for multiple Linux kernels, having the latest stable kernel is not necessarily a security advantage.

As of this writing, Fedora security alerts are handled primarily through online wikis. The prime Fedora security wiki page can be found at <http://fedoraproject.org/wiki/Security>.

Although Fedora Core Linux is primarily supported by the community, Red Hat engineers do develop software for that distribution. However, Fedora is officially released by a group of volunteers. Security updates depend on the availability of the volunteer developers and the time allocated by Red Hat to keep Fedora up to date. Fedora Core Linux is supported for a total of 13 months.

Ubuntu Alerts

Strictly speaking, the Ubuntu distribution is the release that includes the GNU Network Object Model Environment (GNOME) as the default graphical user interface (GUI) desktop environment. In contrast, the

Kubuntu distribution includes the K Desktop Environment (KDE). Finally, the **Xubuntu** distribution includes the **Xfce desktop environment**. There are other flavors of Ubuntu as well.

Security issues may arise that are unique to each of these desktop environments. However, these distributions share the same basic core. Ubuntu, Kubuntu, and Xubuntu all include the same software for command line-based applications from Apache to Zebra.



NOTE

An Ubuntu distribution with a different default desktop environment is called a *flavor*. For example, Ubuntu Mate is a flavor of Ubuntu that includes the Mate desktop environment developed by the Linux Mint project.

Security issues for all Ubuntu-based distributions supported by Canonical are collected in **Ubuntu security notices (USNs)**, available online at <http://www.ubuntu.com/usn/>. The availability of a USN depends on the support given to a release. Ubuntu releases a new version of each of its distributions every six months. These releases include Ubuntu, Kubuntu, and Xubuntu on the desktop and Ubuntu Server Edition for the server. Standard Ubuntu releases are supported for 18 months. Releases are given code names and numbers. For example, Ubuntu's Karmic Koala distributions were released in October of 2009 with a code number associated with the date (9.10).

Every two years, Ubuntu releases its distributions with long-term support (LTS). Its first LTS release, made in June of 2006 and code-named Dapper Drake, was assigned a code number of 6.06. Its latest LTS release, code-named Trusty Tahr, was made in April of 2014 with a code number of 14.04. Ubuntu LTS releases have extended levels of support. Desktop releases are supported for three years, while server releases are supported for five years.



TIP

Because Ubuntu is based on the Debian Linux distribution, you may also choose to monitor Debian security notices.

Because many security notices affect multiple Ubuntu releases, there is no differentiation between the server and desktop in the USN list. A properly configured USN specifies the affected releases. If you prefer to monitor Ubuntu security notices via mailing lists, subscribe to the Ubuntu Security Announcements mailing list at <https://lists.ubuntu.com/mailman/listinfo/ubuntu-security-announce/>.

Keeping Up with Application Security

The developers behind most Linux distributions incorporate solutions to application-related security issues in updated packages. Red Hat and Ubuntu incorporate announcements on security issues related to applications on their main security lists, including USN and RHLA.

Just as there is a lag between the release of a security update for RHEL and CentOS, there may also be a lag between the release of a security update for an application such as the [OpenOffice.org](#) suite on the application Web site and the repositories associated with the Red Hat and Ubuntu distributions. Because enterprises pay RHEL and Canonical for support on the RHEL and Ubuntu distributions, their developers may be more strongly motivated to keep that time lag to a minimum.

The following sections highlight how you might choose to stay up to speed with several important user applications and server services. Be aware, this is an option for an aggressive administrator who is allowed to take extra time to keep up with security issues from a number of sources.

One major source of vendor-neutral vulnerability announcements is maintained by the MITRE Corporation, a nonprofit research organization, on behalf of the National Cyber Security Division of the U.S. Department of Homeland Security. MITRE's list of vulnerabilities is called the **Common Vulnerabilities and Exposures (CVE) list**. For more information, see <http://cve.mitre.org/cve/>. You can use a CVE number to easily look up information about a vulnerability. This is especially important with packages installed on multiple Linux distributions. Each distribution may release its own security advisory, but they will all commonly refer to the general CVE announcement.

This section is focused on user applications, primarily those found on GUI desktop environments. In fact, the three examples listed in this section are GUI applications. The [OpenOffice.org](#) suite is one open source alternative to Microsoft Office. LibreOffice is another. If your users work with console-based applications such as mutt for e-mail or elinks for Web browsing, you may also consider monitoring the mailing lists or similar notification systems provided by the developers of those applications.

The [OpenOffice.org](#) Suite

As with standard Linux distributions, [OpenOffice.org](#) has its own security team. When problems are found, they are highlighted online in security bulletins. You can sign up for [OpenOffice.org](#) security alerts at <http://www.openoffice.org/security/alerts.html>.

The developers behind [OpenOffice.org](#) update their software on a regular basis. When available, [OpenOffice.org](#) may be highlighted in an alert associated with one of the [OpenOffice.org](#) applications. Some security alerts may be addressed by an update to the [OpenOffice.org](#) suite. Others may be addressed in policy. For example, if a virus is being transmitted through some image format, the bulletin may suggest that you and your users avoid loading files with that type of image—at least until the next version of the [OpenOffice.org](#) suite is made available.

When possible, it's easier to implement an update from the repositories associated with a distribution. Distributions do not always keep up to date with their upstream developers, though. In some cases, the upstream software may update to a higher revision but a Linux distribution may not choose to update the package. It's more work for the system administrator to keep up with all the underlying packages and their most recent releases. In some cases, however, it may be necessary.

Web Browsers

Web browsers are one area where choice may make life more difficult for security professionals. Half a dozen different browsers are available just from regular Ubuntu repositories, easily accessible to users with direct online access. Therefore, it may be in your interest to limit what users can download and install. One method based on custom repositories is described later in this chapter.

A number of browsers will run under Linux. Mozilla Firefox is one open source browser commonly found on both Microsoft and Apple operating systems; it runs under Linux as well. Firefox is also the foundation for the Iceweasel browser, which may be installed on some Linux distributions.

The Mozilla Foundation is the open source organization behind the Firefox Web browser and other open source tools such as the Thunderbird e-mail manager and the SeaMonkey Internet application suite. Mozilla maintains a list of security announcements at <http://www.mozilla.org/security/announce/>. Such announcements are classified by product at <http://www.mozilla.org/security/known-vulnerabilities/>.

In contrast, Konqueror is part of the KDE desktop environment. As such, developers at the KDE Foundation address security issues. Fortunately for KDE, there have been very few security issues directly related to that desktop environment. As you can see at <http://www.kde.org/info/security/>, just a few dozen advisories have been released in the past decade on KDE software, including the Konqueror Web browser.

Chromium is another browser that will run under Linux. It's the open source version of Google's Chrome browser. Chromium looks and acts just like Chrome does. If you are used to Chrome on one of the platforms it runs on, you will be very comfortable with Chromium.

Adobe Applications

Given the popularity of Adobe formats such as the Portable Document Format (PDF) and Flash player, it's important for Linux administrators to monitor the status of such applications. Adobe security advisories can be found online at <http://www.adobe.com/support/security/>.

Updates can be problematic. Repositories for older Ubuntu releases may not have even the most up-to-date packages for the Adobe Reader. However, Adobe does provide packages in both Debian and Red Hat package-management formats, which makes it possible to incorporate those packages in a local repository, as described later in this chapter.

Alternatively, Linux has several native PDF readers along with open source alternatives for Adobe technologies like the Flash player. Such packages are often more readily available through standard Ubuntu and Red Hat repositories. The standard GNOME and KDE alternatives are known as evince and KPDF. For more information, see <http://projects.gnome.org/evince/> and <http://kpdf.kde.org/>. Security issues do appear from time to time on these open source applications.

Service Applications

For those who do not have to administer Linux desktop environments, the challenge will be a bit more limited. In this case, all you need to do is monitor the security alerts associated with the applications running on your servers.

The basic pattern for security with respect to open source service applications is consistent. Just as open source development proceeds in the public eye, open source security alerts and updates are publicly available. To that end, this section describes how you can keep up with security issues related to the Apache Web server, the Samba file server, and various File Transfer Protocol (FTP) servers.

The developers behind most Linux distributions, including Red Hat and Ubuntu, support the installation of these services by providing appropriate packages in binary format. Such packages include a number of preconfigured options.

The Apache Web Server

Administrators who run the Apache Web server can stay up to date with related security issues through the Apache Server Announcements list. To sign up, send an e-mail to announce-subscribe@httpd.apache.org. For a current list of vulnerabilities, navigate to http://httpd.apache.org/security_report.html.

The developers behind Apache maintain several versions of its Web server software in a manner similar to the developers behind the Linux kernel. Two trains of Apache may be available. (A *train* is one way of referring to a particular version path in case there are multiple major versions being developed at the same time.) The first choice for Apache is the 1.x train and the second is the 2.x train. You will typically have the 2.x train installed on Linux distributions. In the case of Ubuntu, the package to install the HTTP server is called apache2, although the apache package that would include the 1.x train is no longer available.

Apache has an excellent guide to keeping Web sites secure on its servers. For more information, see http://httpd.apache.org/docs/2.4/misc/security_tips.html. These tips are updated with each new version.

If you have discovered a security issue that has yet to be reported, the Apache security team encourages reports on its private mailing lists. For more information on the procedures, see <http://www.apache.org/security/>.

Apache 2.4 includes several enhanced security features, including the ability to check an **Online Certificate Status Protocol (OCSP)** server for the current status of a digital certificate. OCSP allows Apache to make determinations about whether users are trustworthy. If an attacker manages to get his or her own official certificate through nefarious means, OCSP access can protect users. OCSP allows software that has implemented it to check to see if a certificate has been revoked. Certificates may be revoked for a number of reasons. A certificate that has been revoked should not be trusted. If a user is presenting such a certificate for the purposes of authentication, that user should not be trusted.

WARNING

Unstable versions of any software may cause unpredictable results. Although security is important, developers of unstable distributions must focus on adding features and removing bugs from that software, which might delay changes that address security issues.

For your convenience, Apache does make its software available in binary packages suitable for Red Hat and Ubuntu systems (in formats suitable for the `rpm` and `dpkg` commands). However, such packages may not have been tested by the developers behind these distributions.

The Samba File Server

Administrators who run the Samba file server can stay up to date on the Samba Security Releases Web page at <http://samba.org/samba/history/security.html>. Updates are provided in patch format. To incorporate these updates, you need the Samba source code. Alternatively, you can wait until the developers behind a distribution process the patch into their own binary files. Once updated, you can use the applicable package-management tool to download and upgrade the new Samba software.

The patch release as described on the Samba security Web site is different from the release made by the developers of a distribution. Such developers may have to do additional work to incorporate the newly released patches in a manner compatible with other packages associated with the distribution.

The developers behind Samba maintain several versions of its file server software in a manner similar to the developers behind the Linux kernel. The developers behind Linux distributions may incorporate the code for those patches in earlier versions of Samba. This process is called a *backport*, in which features or code from a newer version of the software are placed into an older version.

If you believe you have discovered a significant security issue in a Samba release, the developers encourage you to e-mail a report to security@samba.org.

File Transfer Protocol Servers

The great majority of administrators stick with Apache for Web services and Samba for Microsoft-style file-sharing services on Linux. In contrast, Linux administrators make a variety of choices when installing a server for FTP files. While the Very Secure File Transfer Protocol (vsftpd) server is the default for Red Hat, SUSE, and even the Linux Kernel Organization, it is far from the only popular FTP server for Linux. Three other major options are as follows:

WARNING

Just because software can be downloaded and installed from the repositories associated with a distribution does not mean that it is supported or secure. For example, as of this writing, the WU-FTPD service is not maintained or secure. But although it has not been maintained since 2004, it is still available from the unsupported repositories of some major Linux distributions.

- **Pure File Transfer Protocol daemon (Pure-FTPd)**—The Pure File Transfer Protocol daemon (Pure-FTPd) is an actively supported server that can run 100 percent with non-root accounts. For more information, see <http://www.pureftpd.org/>.
- **Pro File Transfer Protocol daemon (ProFTPD)**—The Pro File Transfer Protocol daemon (ProFTPD) is an actively supported server with a basic configuration file similar to the Apache Web server with respect to the configuration of multiple virtual FTP servers. For more information, see <http://www.proftpd.org/>.
- **Washington University File Transfer Protocol daemon (WU-FTPD)**—The Washington University File Transfer Protocol daemon (WU-FTPD) was commonly used during the early part of the 2000s. Although it is not maintained, it is still available from the Ubuntu Universe repositories.

Antivirus Options for Linux Systems

Loosely defined, antivirus systems include systems that can detect all types of malware. Although not all malware affects an operating system, it can be transmitted by e-mail and carried by files with common formats. This section covers some of the available malware systems for Linux.

On any computer system, malware falls into several categories:

- **Rootkit**—A **rootkit** is malware that can help an attacker gain administrative control over a computer system.
- **Spyware**—Malware that collects user information is called **spyware**. This may include keystroke loggers.
- **Trojan horse**—A **Trojan horse** is a type of malware that hides as a desirable program but facilitates unauthorized access.
- **Virus**—Strictly speaking, a **virus** is a program that can copy itself and infect a computer. Loosely speaking, the term *computer virus* refers to all malware.
- **Worms**—Self-replicating malware that can send itself to other systems is known as a **worm**. A worm is different from a virus in that it does not require direct user involvement to spread.

Responsible Linux administrators include antivirus software on their systems.

Although there are fewer viruses targeted at Linux, they do exist. One problem associated with Windows operating systems is the number of users who run it with administrative privileges. Although Linux is configured to discourage running most commands with administrative privileges, it's possible to configure Linux with only the root account. Therefore, Linux is not immune. Although Linux may be a more difficult target for attackers, it's still a target. Some Linux applications, such as the Apache Web server, have a dominant market share.

Even though attackers create fewer malware components for Linux systems, Linux can still be infected or even act as a carrier. In other words, Linux services such as e-mail servers, FTP servers, file servers, and more can carry malware that remains dormant until exposed to a Microsoft or an Apple operating system. As noted, there is also malware that targets and infects Linux systems.

Perhaps the most common malware for Linux systems is based on the rootkit: software that replaces system-level binaries to hide the existence of malicious software. A rootkit may also install a back door into the system so an attacker can maintain access to it even in cases where the original entry point has been repaired. In fact, some attackers patch the vulnerability that allowed them to enter the system to prevent other attackers from following them in. The rootkit enables them to maintain access, even after the original exploit is no longer viable.



NOTE

Only some of the antivirus tools described in this section are released under the GNU General Public License (GPL).

The Clam AntiVirus System

The **Clam AntiVirus (ClamAV)** system is an antivirus system for Linux. Although you can use it to perform normal file scanning, it was initially designed to be used on mail gateways. In other words, it's meant to work with e-mail servers such as sendmail, Postfix, and Exim. It works on Unix-type operating systems, including **Solaris**, the **Berkeley Standard Distribution (BSD)**, and Linux. (Solaris is a variant of Unix originally developed by the former Sun Microsystems. The BSD is a clone of Unix, similar to Linux.) There is also an implementation of ClamAV for Mac OS X.

ClamAV supports searches within archives, compressed files, and more. It is updated on a regular basis. If you use ClamAV, it may be in your best interest to subscribe to the associated mailing lists at <http://lists.clamav.net>.

AVG Antivirus

Although **AVG Technologies** is focused on the Microsoft antivirus market, it does make anti-malware tools for Linux. The cost and degree of support varies. These tools are as follows:

- **AVG Antivirus for Linux**—AVG makes a freely downloadable antivirus tool targeted primarily for Linux clients. This software is available in a number of formats, including those based on Red Hat and Ubuntu (Debian) package-management systems. As noted by AVG, this software is for private and non-commercial use. Although AVG does maintain message boards for this software, few messages on these boards relate to Linux.
- **AVG Antivirus Server Edition for Linux**—AVG also makes a commercial antivirus tool. This edition includes antivirus, anti-spyware, and anti-spam tools.

The Kaspersky Antivirus Alternative

Kaspersky Lab has a suite of anti-malware products available for Linux. These are commercial products. Kaspersky EndPoint Security for Business Core runs on Linux systems. It provides a single management console to allow control over all the different endpoint types within your organization. For more information, see <http://www.kaspersky.com/>.

SpamAssassin

Although there are a variety of commercial solutions to minimize the amount of spam present in e-mail messages, **SpamAssassin** is perhaps the open source standard in spam prevention. It is a common e-mail filter for many Internet service providers (ISPs).



NOTE

SpamAssassin is known more formally as the Apache SpamAssassin project. This is because its developers are sponsored by the same Apache project that is behind the Apache Web server.

SpamAssassin is actually part of the standard repositories for Red Hat Enterprise Linux 7 (RHEL 7). That means it is supported by Red Hat. In other words, if you have an appropriate subscription to the Red Hat Network, Red Hat engineers may actually support the work you do with SpamAssassin.

For more information on SpamAssassin, see <http://spamassassin.apache.org/>.

technical TIP

If you want to use Google to search within the public archives of a mailing list, use the `site:` tag. For example, the following search in Google searches for all instances of the word Samba in the Red Hat Enterprise Linux 7 mailing list:

```
samba site:redhat.com
```

Similar searches are possible of public bug reports using the associated URL with the `site:` tag.

Detecting Other Malware

Linux tools for detecting other malware include the following:

- **chkrootkit**—One system designed to check for rootkits is chkrootkit. It checks a system for known rootkits along with typical symptoms of unknown rootkits, such as deletions of and changes to key log files. For more information, see <http://www.chkrootkit.org/>.
- **Rootkit Hunter**—Functionally similar to chkrootkit is Rootkit Hunter (http://www.rootkit.nl/projects/rootkit_hunter.html). Although it was originally developed by Michael Boelen in the Netherlands, other open source developers have since taken the lead. For the latest information, see <http://rkhunter.sourceforge.net/>.

As suggested in the README file for Rootkit Hunter, rootkit-detection tools (including chkrootkit) are just components in a layered security strategy.

Using Bug Reports

If you have not installed a service, such as WU-FTPD, security reports on that particular service may not matter to you. On the other hand, security reports on an Apache project could refer to any of the services developed through the Apache Software Foundation, as listed at <http://apache.org/>.

In many cases, you will need to get into the details of the security advisories, which are often also included in bug reports. One place to start when reviewing the status of security on Linux is <http://www.linuxsecurity.org/>. That Web site collects security updates. It is sponsored by Guardian Digital and has been for well over a decade. It includes the latest advisories, divided by distribution.

In most cases, the developers behind distributions, applications, and services maintain mailing lists associated with their software. If you think there is a problem with your distribution, most developers encourage you to discuss the issue on their mailing lists. Others may have already encountered a similar problem. If they do not already have a solution, they may be able to help you better determine the scope of the problem. You can search many mailing lists either on their Web sites or through a regular search engine such as Google.

There are exceptions. If you believe you have identified an important security issue and have the experience to understand the issue, many projects provide an e-mail address to which you can send the report privately.

Ubuntu's Launchpad

Launchpad is a Canonical contribution to open source software development. In some ways, it's similar to SourceForge. It has many other functions, however. As noted at <https://launchpad.net/>, Launchpad is a software-development platform. It provides the functionality described in **Table 11-1**. Ubuntu gathers issues related to security in the USN described earlier in this chapter. The Ubuntu security team triages security-related bugs. Currently, you can find triaged bugs at <https://bugs.launchpad.net/~ubuntu-security/>.

If you believe there is a security issue with an Ubuntu release, navigate to the noted URL and search the active bugs. Some of the bug discussions include exchanges with other users. It takes some judgment to determine whether a bug is a security issue that affects you.

If you are unsure whether a problem is a security issue, report it on Launchpad. Sometimes, a user with more experience in the subject area can help you address the problem. If the issue is a bug, Ubuntu developers will classify it as such. If the issue is in fact a security problem, someone from the Ubuntu security team will classify it as such and triage it for attention.

TABLE 11-1 Security-related functions of Launchpad.

FUNCTION	DESCRIPTION
Bug tracking	Launchpad goes beyond user reports to identify packages and distributions for later evaluation by Ubuntu developers. It may include bugs associated with other projects and even other Linux distributions. It was formerly known as Bugsy Malone. It may include security issues.

Code	Launchpad provides a platform for developing source code on open source projects. It is based on the Bazaar version control system at http://bazaar.canonical.com/en/ . It is closely related to the Code Review function at https://help.launchpad.net/CodeReview . It may be used to track the development of new security updates.
Ubuntu	Launchpad hosts development work on package updates and new distribution releases. It may be used to track the development of new security updates.
Mailing lists	Launchpad sponsors mailing lists related to security and more.

Red Hat's Bugzilla

A **Bugzilla** is a Web-based bug-tracking and management tool commonly used on open source projects from Red Hat to the GNOME desktop environment. The Red Hat Bugzilla is derived from the bug-tracking system used for projects sponsored by the Mozilla Foundation. It has been adapted by Red Hat as the system for its bug reports at <https://bugzilla.redhat.com/>.

If you believe there is a problem with a Red Hat distribution, you can first discuss it on a Red Hat mailing list. Many excellent developers monitor these lists with a desire to help others. If the problem that you present is a security issue, many of these developers can help you highlight the issue with the Red Hat Security Response Team.

The next step would be to search the bug reports at the noted URL. If you have a subscription to the Red Hat Network, you will have access to more bug reports. A few bug reports are limited to Red Hat employees and official beta testers.

If the problem you have found has not already been reported, the next step is to create a bug report. Red Hat developers will assess that report. If they don't believe it's a problem, they will close it, noting the reason for their decision. If it is similar to another bug report, they may close it, citing it as a duplicate. If it is a unique problem, they will list an action and priority. If it is a security issue, they will highlight it as such.

Application-Specific Bug Reports

A lot of software that is included with a distribution is developed by others. Although distributions such as Red Hat and Ubuntu put it all together, some bugs are application specific.

To make sure a problem is addressed by the right developers, you may want to report the problem on the bug systems provided by those developers. If you report a problem in the bug system associated with a distribution and it's judged to be a problem, the responsible developers will report the issue upstream. In other words, they will create an application-specific bug report for you or otherwise notify the developers behind an application. That takes time, however. If you have found an actual security issue, time may be of the essence.

The bug reports for many applications are based on the associated graphical desktop environment. Most Linux GUI applications are developed under the auspices of one of the two major desktop environments: GNOME and KDE. Other desktop environments frequently use applications built for these GUIs. For example, the Xfce desktop environment uses a number of GNOME applications. In addition, it is common for users to include KDE applications in the GNOME desktop environment and vice versa. Whichever bug-tracking system applies, remember to check for a previous report first. You may even find a solution to your problem. Otherwise, you may be able to create a better report for appropriate developers.

GNOME Project Bugs

You can find and report bugs associated with the GNOME project at <https://bugzilla.gnome.org/>. This Bugzilla is associated with a number of GNOME applications. A few examples include the Brasero disc burner, the Evolution personal information manager, the Nautilus file manager, the Totem media player, the Vino **Virtual Network**

Computing (VNC) server, and the GNU Image Manipulation Program (GIMP). (VNC is a system for sharing views of graphical desktop environments over a network.) In other words, if you identify a problem with almost any application in the GNOME desktop environment, the first step is to check the bug reports associated with the GNOME Bugzilla. If they confirm that a bug is a security issue in released software, GNOME developers give it their highest priority.



NOTE

A couple of major GNOME applications have their own bug-reporting systems: the Firefox Web browser and the [OpenOffice.org](#) suite. Although GNOME developers do contribute to these products, they are sponsored and developed by different organizations.

KDE Project Bugs

You can find and report bugs associated with the KDE project at <https://bugs.kde.org/>. This Bugzilla is associated with a number of KDE applications. A few examples include the K3b disc burner, the KMail e-mail client, the Konqueror file manager and Web browser, the Kaffeine media player, the digiKam photo manager, and the KOffice suite. (The focus of the KDE project is somewhat different from GNOME in that it includes its own native office suite, Web browser, and e-mail manager.)

Other Applications

The developers behind several major applications have their own bug-reporting tools. The following are just three examples:

- **OpenOffice.org Suite**—The reporting procedure is shown at <https://bz.apache.org/ooo/>.
- **Mozilla Project Applications**—You can make bug reports for Mozilla applications, including the Firefox Web browser and the Thunderbird e-mail manager, at <https://bugzilla.mozilla.org/>.
- **Adobe products**—You can make bug reports for Adobe products, including Acrobat Reader and Flash Player, at <https://www.adobe.com/cfusion/mmform/index.cfm?name=wishform>.

Service-Specific Bug Reports

Bug reports for a number of different open source services may also be security related. Alternatively, they may just be your contribution to helping developers create better software.

This section briefly describes the bug-reporting tools for a few major applications. Some services have fully featured Bugzilla-style systems; others have so few bugs that you can make any such reports directly to mailing lists. These are just a few of the many available services, but should give you an idea of the process for other services.



NOTE

If you have developed a patch to a bug that you have identified, most developers will gladly review your patch for inclusion in their service software.

- **Apache**—The developers behind the Apache project explicitly request that users make reports of security issues per the instructions at http://httpd.apache.org/security_report.html. Nevertheless, there will be times that bug reports are escalated into security issues. Apache bug-report procedures are described at https://httpd.apache.org/bug_report.html. Briefly, Apache developers encourage users to first discuss the possible problem on their users' mailing list, based on the procedures listed at <https://httpd.apache.org/userslist.html>.

- **Squid**—Although Squid maintains an invitation-only mailing list for bug reports, it also maintains a more publicly available Bugzilla system at <http://bugs.squid-cache.org/index.cgi>. As with the developers of other bug databases, Squid developers encourage users to first search their FAQ at <http://wiki.squid-cache.org/SquidFAQ/>. There are a number of guidelines for bug reports at <http://wiki.squid-cache.org/SquidFAQ/BugReporting/>. If you follow these guidelines closely, you may even find the solution to the problem before reporting a bug. Note that there are very few Squid developers. They evaluate bug reports for any security implications. Incidentally, Squid has a variety of sponsors, from the U.S. National Science Foundation to Kaspersky Labs.
- **Samba**—The developers behind Samba also maintain a Bugzilla-based bug-tracking system. Their process is described at <http://www.samba.org/samba/bugreports.html>. Of course, you should search their documentation and bug reports first. (Incidentally, the bug-reporting Web page includes procedures for proposed source code patches.)
- **vsftp**—As of this writing, very few bugs are reported for the vsftp server. At least, that is the report on the home page for vsftp developers, <http://freecode.com/projects/vsftpd/>. Because there is no bug-reporting option listed on the vsftp Web page, this is one area where it may be more efficient to report the bug through a distribution. Vsftp is the default FTP service for Red Hat systems and is in the main repository for Ubuntu systems. Therefore, the developers behind both distributions have committed to support this service.

Security in an Open Source World

The paradigms in an open source world are different. Most Linux users do not have support contracts from Red Hat, Canonical, or any other company that may support this operating system. Nevertheless, as you should already know, there is a substantial open source community available on mailing lists, blogs, Internet Relay Chat (IRC) channels, and more.

Eric Raymond, a leader in the open source movement, notes in *The Cathedral and the Bazaar* that one of the secrets of the success of open source is the power of the community. Raymond dubbed this “Linus’s Law,” in reference to Linus Torvalds: “Given enough eyeballs, all bugs are shallow.” In other words, open source relies on the power of the community. In fact, it would not be possible for the 2,000 employees at Red Hat or the 200 employees at Canonical to compete with Microsoft’s nearly 100,000 employees without the help of the open source community.

With many thousands of developers, the open source community has organized itself into a number of areas, including security. Three key security-related open source organizations are the Institute for Security and Open Methodologies (ISECOM), the U.S. National Security Agency (NSA), and the Free Software Foundation (FSF).

The Institute for Security and Open Methodologies

ISECOM is perhaps the leading organization dedicated to the security of open source software. Many of the key players at ISECOM are the authors of *Hacking Linux Exposed*.

ISECOM is the organization behind the *Open Source Security Testing Methodology Manual* (OSSTMM), currently undergoing its third revision. ISECOM is also the sponsor of several professional certifications, including the OSSTMM Professional Security Analyst (OPSA), the OSSTMM Professional Security Expert (OPSE), the OSSTMM Professional Security Tester (OPST), and the OSSTMM Wireless Security Expert (OWSE).

ISECOM has taken the lead in promoting the interests of open source software in security forums worldwide. They have pushed open source solutions through the Open Trusted Computing (OpenTC) consortium, sponsored by the European Union.

The National Security Agency

One critical advance in security on the Linux operating system is the release of Security Enhanced Linux (SELinux). SELinux was first developed by the NSA as a set of extensions to Linux to implement mandatory access

controls. Despite the nature of the NSA as an American intelligence agency, the NSA has made it part of its “Information Assurance” mission to enhance the security of major operating systems. To that end, the NSA took the lead in developing SELinux. It has released SELinux under the GPL. In other words, the source code for SELinux is publicly available. Anyone who does not trust the NSA can inspect the source code for themselves.



NOTE

Although the NSA has clearly done significant work on securing Linux as an operating system, it has officially stated that it does not “promote any specific software or platform.”

The Free Software Foundation

The FSF is the group behind the development of the commands and libraries cloned from Unix. When coupled with the Linux kernel as originally developed by Linus Torvalds, these commands and libraries were used to create the original Linux operating system.

The FSF is the organization behind the GPL. As the developer of many commands and libraries cloned from Unix, the FSF and allied developers have released the source code behind those commands and libraries under the GPL. Under that license, the FSF retains the copyrights to their work. Under the GPL, others may use and build upon that work, as long as they acknowledge the work originally done by the FSF. If the source code has been changed, that must also be released under the GPL.



NOTE

This explanation of the GPL is not intended as legal advice. If you need legal advice with respect to open source licenses, one option is the group of lawyers associated with the Software Freedom Law Center, accessible online at <http://www.softwarefreedom.org/>.

User Procedures

If you are relatively new to open source software, be aware that the culture around open source software is different. If you use open source software, you are a part of the community. As such, you are welcome to join mailing lists and similar forums to ask for help. In return, it is hoped that you will provide help when possible on the same lists and forums.

Of course, if you have a support contract from a company such as Red Hat or Canonical, it is in your best interest to take full advantage of that resource. Most Linux users do not have that advantage, however, and work through the community instead.

In general, the gurus who populate many Linux and open source mailing lists appreciate users who have done their homework before asking a question. That means taking advantage of documentation available on Web sites for a distribution, for applications, and for services, and even the HOWTO documents provided through the Linux Documentation Project at <http://www.tldp.org/>.

If you do not find the answer to a problem in your research, the next step is to ask a question on a relevant mailing list. It may take some research to find the right mailing list; there are more than 100 available just for Red Hat software. If and when you do ask a question, make it as easy as possible for those who will read your message. The following is an incomplete list of guidelines that can help:

- **Do your homework**—Make sure this is really a new issue and not one that has been previously reported.
- **If you have a new problem, start a new thread**—Make sure the subject line is short and descriptive. Stay focused on the topic at hand. Post in text mode. Many Linux gurus use text-based e-mail readers.

- **Provide a brief description of the issue**—Include key parts of applicable configuration files and log messages. Note the versions of the software and services in question.
- **Document what you have done**—List the changes you have tried and note the documents you have read. This shows the community you have done your homework before asking gurus to use their valuable time to help.
- **Add responses to the bottom of a message**—Most Linux gurus prefer users who add their responses to the bottom of an e-mail message so they can follow the discussion from top to bottom. Bottom-posting enables people to read through the thread just as you would a book or a magazine article—from top to bottom.

Deciding Between Automated Updates or Analyzed Alerts

It is certainly a viable option to set up a system to automatically download and install all updates, with the possible exception of new kernels. With the number of developers who work on Linux, the reliability of updates to Linux packages is high. Because there are no statistics available on the reliability of such updates, however, the choice boils down to a matter of trust. The question of trust goes beyond a simple assessment of the integrity of open source developers. If you administer systems with special configurations or require high availability, you have reason to check every update on a non-production system first. It is not possible even for the open source community to test every update on every possible configuration of hardware and software.

Do You Trust Your Distribution?

Many Linux distributions are backed by industry leaders, such as **SUSE**. (SUSE is a Linux distribution originally developed in Germany, now owned by Novell.) Some distributions may be more appropriate for different nations or cultures, such as **Mandriva** for France and Brazil, **Turbolinux** for Japan, **Red Flag Linux** for China, and more. SUSE is an important Linux distribution in the European Community. If you are from one of these countries and can get support for a distribution more appropriate for your locale, these are excellent options.

The question of trust goes beyond national and cultural borders, however. If the updates provided for a distribution always work on the systems you administer, then the developers behind that distribution have earned some level of trust.

However, the developers of a distribution to some extent depend on the developers of open source applications and services. The developers of a distribution can test application and service updates for compatibility with the distribution. In some cases, they will test such updates against certified hardware configurations.

Do You Trust Application Developers?

Linux is famous for its diversity of available applications. With such diversity comes risk, however. Diversity in applications may lead to some combination of software that causes problems or fails.

The developers behind an application do their best, with available resources, to test their software on major Linux distributions. They also do their best to test their software with different architectures. The resources available for many applications are severely limited, however. With that in mind, you may want to limit the applications available for systems on any networks that you administer. To do so, you will need to take control of updates, either with custom patch-management repositories or the update managers described later in this chapter.

If you run Linux primarily as a server, applications should be less important. A GUI is not required to run major Linux services such as Apache, Samba, Squid, the Common Unix Printing System (CUPS), and so on.

Do You Trust Service Developers?

If you use Linux as a server, the question beyond trusting the developers of an application is whether you trust the developers behind a service. The answer may vary by service, as most Linux services are developed and maintained by different groups.

Some major services, such as Apache and Samba, are developed and maintained by dozens—perhaps hundreds—of developers, testers, and more. With such resources, they can respond fairly quickly to security issues and other problems. They can also test updates as applied to different distributions and even hardware configurations. Not all Linux services can call upon so many developers and testers, however.

With the choices available for services, a related question is whether you trust other administrators. Some enterprising administrators may substitute a different service, such as Postfix for sendmail to manage e-mail on a network. That may even be the right decision. To retain configuration control over such decisions, however, you may want to create a custom repository for the systems on the local network. That is the province of **Linux patch management**, the system of Linux OS package updates.

Linux Patch Management

Current versions of Linux distributions require Internet access to download updated software. That software is downloaded from a group of packages known as a *repository*. The repositories for a distribution are frequently mirrored on different servers worldwide.

Such connections depend on the speed of the repository or mirror, along with the geographical proximity of that system. Even if you find a repository mirror that is fast and relatively close, the packages that need to be downloaded from that system may be huge. For example, updated packages associated with the [OpenOffice.org](#) suite can easily exceed hundreds of megabytes. If that download is multiplied by a few dozen (or more) systems on the local network, the additional load on an Internet connection may be expensive.

A local repository can be a simple mirror of that remote system. A local repository is downloaded only once, minimizing the load on a corporate Internet connection. One additional reason to create a local repository is to limit what users and administrators can install on their systems. If you support access to all repositories, users may choose that second and third office suite and that fifth Web browser. Then you will be responsible for making sure all such systems are secure.

The methods for configuring repositories vary widely. Repositories for Red Hat Enterprise Linux and Ubuntu systems provide two diverse models that encompass the range of what is available for different Linux distributions.

Before exploring the differences between repositories and update systems, you need to know more about the commands and configuration files that work with different repositories. The **yum** command is most closely associated with Red Hat systems; the **apt - *** commands are most closely associated with Ubuntu systems through their heritage in the Debian distribution. These commands are not distribution specific, however. For example, some repositories associated with the **apt - *** commands have been configured for Red Hat-style distributions such as Fedora Core Linux.

The advantage of the **yum** and **apt - *** commands is that they can automatically identify dependent packages. For example, if you try to install the `system-config-samba` configuration tool on a Red Hat system, the **yum** command automatically determines that packages associated with the Samba file server are also required and includes the associated packages in the installation.

Standard yum Updates

The predecessor to the **yum** command was built for a Linux distribution known as Yellowdog Linux. When it was adapted by developers at Duke University for the Red Hat distribution, it became known as the Yellowdog Updater, Modified, or **yum** for short.

The `yum` command uses repositories configured through the `/etc/yum.conf` configuration file. Normally, that file includes repositories defined in the `/etc/yum.repos.d/` directory. The formats of the `/etc/yum.conf` files differ slightly between Fedora and Red Hat Enterprise Linux systems.

Updates on Fedora

For the Fedora distribution, the update repository is defined with the following stanza in the `fedora-updates.repo` file. In the first directive, the release version is substituted for `$releasever` and the basic architecture is substituted for the `$basearch` variables.

```
[updates]
name=Fedora $releasever - $basearch - Updates
```

The directives that follow support a rotating search of repository mirrors, as defined by the text file associated with the noted `mirrorlist` directive. The `failovermethod` directive starts with the first URL in the noted text file.

```
failovermethod=priorityupdates/$releasever/$basearch/
mirrorlist=https://mirrors.fedoraproject.org/metalink?repo=
  ↪updates-released-f$releasever&arch=$basearch
```

Alternatively, you could set up a single URL for updates. In fact, to set up a local update repository, you would use the `baseurl` directive to point to a local Web or FTP server with update packages:

```
#baseurl=http://download.fedoraproject.org/pub/fedora/linux/
```

The final three directives in the stanza activate the repository, support checks of the packages with a GNU Privacy Guard (GPG) key, and identify the location of the GPG key on the local system:

```
enabled=1 gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-fedora-$basearch
```

Although the mirrors and packages differ, the basic format for updates on rebuild distributions such as CentOS 5 is essentially the same as it is for Fedora Core Linux.

In other words, while CentOS 7 uses the same source code as Red Hat Enterprise Linux 7, it uses the Fedora Core Linux system for updated packages. This allows CentOS to avoid having to replicate the Red Hat Network, which was not always open source.



NOTE

For an example of the list associated with the `mirrorlist` directive, navigate to <https://mirrors.fedoraproject.org/metalink?repo=updates-released>. It should download a text file of mirrors in a file named `metalink`. The file will list available Fedora mirrors in extensible Markup Language (XML) format.

Updates on Red Hat Enterprise Linux

Updates on Red Hat Enterprise Linux systems are done through connections to the Red Hat Network. This is a powerful tool that enables an administrator to pull updates with the `yum` command or push them directly from a Web-based interface to one or many systems. The Red Hat Network is discussed later in this chapter. The connection to the Red Hat Network is based on configuration files in the `/etc/sysconfig/rhn/` directory. Specifically, registration information is stored in the `systemid` file.

Alternatively, you can convert Red Hat Enterprise Linux systems to pull updates from a local repository. However, that choice means that required updates will not be tracked on a Red Hat Network account and may invalidate any support contract you have with Red Hat. Alternatives for locally based updates through Red Hat are based on a Red Hat Proxy server and a Red Hat Satellite server.

Red Hat offers an open source product that can manage updates to Red Hat Enterprise Linux, CentOS, Fedora, and related systems, known as Spacewalk. For more information, see <http://www.redhat.com/spacewalk/>.

Standard apt - * Updates

The `apt - *` commands use repositories configured through the `/etc/apt/sources.list` configuration file. The default version of this file includes connections to a number of repositories. First, the `deb` and `deb-src` directives in this file connect separately to repositories for binary and source code packages. Therefore, the following directives for the Ubuntu Trusty Tahr release specify connections to the main and restricted repositories:

```
deb http://us.archive.ubuntu.com/ubuntu/ trusty main
deb http://us.archive.ubuntu.com/ubuntu/ trusty restricted
```

Although these directives specify the default repository for U.S.-based systems, the associated IP address suggests that the server is actually located in the United Kingdom. Therefore, for faster updates, you may want to replace the <http://us.archive.ubuntu.com/ubuntu/> URL with one of the official mirror sites listed at <https://launchpad.net/ubuntu/+archivemirrors/>. As shown on that Web page, Ubuntu mirrors may use the Hypertext Transfer Protocol (HTTP), the FTP protocol, or the rsync protocol, with a bandwidth and location more suited to your network.

Ubuntu Repositories

The Ubuntu distribution classifies repositories into four categories, with four different groups of functions:

- **Main**—Packages in the Ubuntu **main repository** are supported and released under open source licenses.
- **Restricted**—Packages in the Ubuntu **restricted repository** are supported and have restrictive licenses. Restricted packages are generally limited to those packages that help Ubuntu work out of the box with hardware associated with closed source drivers.
- **Universe**—Packages in the Ubuntu **universe repository** are unsupported but are released under open source licenses.
- **Multiverse**—Packages in the Ubuntu **multiverse repository** are unsupported and are released under restrictive licenses.

Repositories in these categories may be further subdivided by the way they are released. These divisions include the following:

- **Standard**—Standard packages are those released with the final release of a distribution, such as the Trusty Tahr release in April of 2014.
- **Updates**—Update packages are later versions of packages in the standard category.
- **Security**—Security updates are later versions of packages in the standard category that address security issues.
- **Backports**—Backport packages incorporate new features from later releases. For example, a backport of a kernel package may include a new feature from a later version of a kernel.

Therefore, packages may be organized in 16 different groupings. For example, in the main category, there is a standard, an updates, a backports, and a security repository.

In addition, Ubuntu provides packages from partners in separate partner repositories. Such repositories are also further subdivided into standard, updates, backports, and security repositories.

Some third parties also sponsor their own repositories that can be configured like and integrated with Ubuntu repositories. Even if such software has been tested by Ubuntu developers, it may or may not be supported by Canonical. Remember, too, that Canonical does not support software in the universe and multiverse repositories.

As shown in the following directives, security updates come from a different URL. Because of the time-sensitive nature of security updates, it may be best to accept the default, even if your systems are not located in Europe. If desired, you can add `universe` and `multiverse` to the end of these directives.

```
deb http://us.archive.ubuntu.com/ubuntu/ trusty-security main
  ↳ restricted
deb-src http://us.archive.ubuntu.com/ubuntu/ trusty-security main
  ↳ restricted
```

For more information on the types of packages available from different Ubuntu repositories, see the box on the preceding page.

Options for Update Managers

This is a general section that reviews basic alternatives for updating one or many systems on a network. If you are responsible for just a single system, it is most efficient to use the update tools at hand from the local system. If you want, you can configure automated updates. While the default is to pull updates from remote repositories, with the right tools, you can also push updates from systems such as the Red Hat Network. And with the appropriate changes to `yum` or `apt -*` command configuration files, you can configure access from local or remote repositories.

Configuring Automated Updates

You can configure automatic updates for Ubuntu and Red Hat distributions. Updates for both systems depend on the installation of a specialized package. The following sections describe the process for each distribution.

Automatic Ubuntu Updates

You can configure an Ubuntu system to automatically update itself. One of the configuration files used to determine whether you are going to make use of automatic updates is `/etc/apt/apt.conf.d/50unattended-upgrades`. Such upgrades are run on a daily basis, based on the `apt` script in the `/etc/cron.daily/` directory. This relies on the use of the cron task scheduler. Cron is a program that schedules other programs to run without any intervention.

The following directive in the `/etc/apt/apt.conf.d/10periodic` file will enable this unattended upgrade feature:

```
APT::Periodic::Unattended-Upgrade "1";
```

Examine the `50unattended-upgrades` file. The following directives suggest that while security upgrades are downloaded and installed by default, updates from the regular update repositories are not, as they are commented out with the double forward slash (`//`), a typical comment character in C++ scripts.

```
Unattended-Upgrade::Allowed-Origins {
    "${distro_id}:${distro_codename}-security";
//    "${distro_id}:${distro_codename}-updates";
//    "${distro_id}:${distro_codename}-proposed";
//    "${distro_id}:${distro_codename}-backports";
};
```

The remaining directives, shown in [Figure 11-1](#), are from a Trusty Tahr implementation. This file includes a stanza with names of packages to not update. One package you might include in that list is the Linux kernel because automatic updates of that package will most likely lead to problems on production systems.

```
// Automatically upgrade packages from these (origin:archive) pairs
Unattended-Upgrade::Allowed-Origins {
    "${distro_id}:${distro_codename}-security";
//    "${distro_id}:${distro_codename}-updates";
//    "${distro_id}:${distro_codename}-proposed";
//    "${distro_id}:${distro_codename}-backports";
};

// List of packages to not update (regexp are supported)
Unattended-Upgrade::Package-Blacklist {
//    "vim";
//    "libc6";
//    "libc6-dev";
//    "libc6-i686";
};

// This option allows you to control if on a unclean dpkg exit
// unattended-upgrades will automatically run
//   dpkg --force-confold --configure -a
// The default is true, to ensure updates keep getting installed
//Unattended-Upgrade::AutoFixInterruptedDpkg "false";

// Split the upgrade into the smallest possible chunks so that
// they can be interrupted with SIGUSR1. This makes the upgrade
// a bit slower but it has the benefit that shutdown while a upgrade
// is running is possible (with a small delay)
//Unattended-Upgrade::MinimalSteps "true";

// Install all unattended-upgrades when the machine is shutting down
// instead of doing it in the background while the machine is running
// This will (obviously) make shutdown slower
//Unattended-Upgrade::InstallOnShutdown "true";
```

FIGURE 11-1

Ubuntu 50unattended-upgrades configuration file.

If the comment characters were to be removed, the final directive would send an e-mail to the root administrative user on the local system when an unattended upgrade is run. If an appropriate e-mail service is available, you could change that e-mail address to an actual e-mail address.

Automatic Red Hat Updates

Automatic updates in Red Hat Enterprise Linux are handled through a cron task that directs yum to perform the updates on a regular basis. The following is a section of the /etc/yum/yum-cron.conf file that specifies how the cron job should handle the tasks.

```
[commands]
# What kind of update to use:
```

```

# default = yum upgrade
# security = yum --security upgrade
# security-severity:Critical = yum --sec-
                               severity=Critical

upgrade
# minimal = yum --bugfix upgrade-
            minimal

# minimal-security = yum --security upgrade-
                     minimal

# minimal-security-severity:Critical = --sec-severity=Critical
upgrade-minimal
update_cmd = default

# Whether a message should be emitted when updates are available,
# were downloaded, or applied.
update_messages = yes

# Whether updates should be downloaded when they are available.
download_updates = yes

# Whether updates should be applied when they are available. Note
# that download_updates must also be yes for the update to be applied.
apply_updates = no

```

Using this configuration file, you can tell the cron job to perform all updates, just security updates, or only critical updates, or specify some minimal number of updates that may be limited to security updates. You can configure it to provide a message when updates are available and to download the updates. This will save time when it comes to applying them because they will already be on your system. You can also specify whether you want them to be updated automatically or if you want the system to hold them to wait for you to review the updates and determine how you want to proceed. This set of commands gives you a lot of control over how you handle the process of updating your system.

Pushing or Pulling Updates

You can push or pull updates. In other words, with the help of administrative tools like the Red Hat Network, ZENworks, Landscape, or Spacewalk, you can push the installation of updates on remote systems. While that may not be cost effective on a network with one or two systems, it is a time saver when administering a few dozen systems. Properly configured, administrative-update tools can push the same updates to dozens of systems simultaneously. (Of course, such simultaneous updates work only if they have been tested and are then applied on systems with identical hardware and software configurations.) Some update tools also support remote administration. With the right centralized administrative tool, you may not even have to configure a remote login service such as the Secure Shell (SSH).

Local or Remote Repositories

To go a step further, you can limit the number of packages available in local repositories. One way to do so is to limit those repositories available to local systems. On Ubuntu systems, you could configure clients and servers to avoid connections to the unsupported universe and multiverse repositories. On Red Hat systems, you could limit the channels. For example, suppose a RHEL 7 system is subscribed to additional channels named Network Tools,

Virtualization, Supplementary, and Optional Productivity Apps. On the Red Hat Network, you can disable subscriptions from a system to all of these channels.

On all systems, you can avoid connections to third-party repositories. Although many third-party repositories are reputable, they are designed to include popular software not specifically supported by the developers of distributions such as Red Hat and Ubuntu.

Configuring a Local Repository

It is easiest to use the remote repositories already configured for most Linux distributions. The connections to such repositories have already been tested. If a problem arises with such systems, many administrators are likely to complain, and the problem is likely to be fixed quickly, without any additional intervention.

There are advantages to maintaining a local repository, however. At minimum, a local repository reduces the load on a corporate Internet connection. If you want to keep it simple, you could use an **rsync** command to mirror a remote repository or use a service made available for that purpose such as Red Hat's Proxy or Satellite server. The commercial options are described in the next section.

FYI

Third-party repositories designed for Ubuntu and even Red Hat Enterprise Linux are frequently located in countries where copyright laws hold less sway. For example, some third-party repositories include video players that may bypass certain legal requirements such as the Digital Millennium Copyright Act (DMCA). Although such packages are popular among Linux users, the presence of such packages on systems that you administer may expose your organization to legal liability.

Just a few mirrors support rsync connections. Those that do should understand that the **rsync** command is used to synchronize local and remote systems. To review the repositories mirrored on the servers of the Linux Kernel Organization, run the following command:

```
$ rsync mirrors.kernel.org::
```

Be aware: Mirrors—especially if they involve multiple architectures—can occupy dozens of gigabytes of space. Therefore, you need available space, and you should run the command at a time when demand on the Internet connection is relatively low. To explore the Ubuntu-based mirrored repositories, run the following command:

```
$ rsync mirrors.kernel.org::ubuntu/dists/
```

The following command includes all such update packages in the main, restricted, universe, and multiverse repositories and synchronizes them to the local /updates/ directory:

```
$ rsync -avz
  ↗ mirrors.kernel.org::ubuntu/dists/trusty-updates/. /updates/
```

There is always some time delay between the packages on a mirror and the packages at the source. To minimize that delay, you can rsync a local mirror with that maintained by Ubuntu. For more information on what is available on Ubuntu's U.S. mirrors, run the following command:

```
$ rsync us.archive.ubuntu.com::ubuntu/
```

When a mirror is available locally, you still need to share the directory using a server such as Apache or vsftpd. You can then point clients and servers on the local network to that repository in their /etc/apt/ sources.list file.

TIP

One alternative for mirroring Ubuntu repositories is based on the `apt-mirror` command. For more information, see <http://apt-mirror.sourceforge.net/>.

Commercial Update Managers

If you administer a large number of Linux systems, it may be cost-effective to use a commercial update manager. These systems enable you to update a number of different systems simultaneously. For example, you could, with one command, update all regular 32-bit systems with one group of packages. You could update all regular 64-bit systems with a second group of packages. Alternatively, you could set up a group of Web servers, ready for updates to the Apache Web server and related database packages.

There are three major commercial update managers available for Linux. The Red Hat Network can help you administer subscribed systems from clients/servers or from a central Web-based interface.

The Red Hat Network

There are several reasons to subscribe to the Red Hat Network. It supports downloads of binary copies of the Red Hat Enterprise Linux distribution. It supports downloads of new and updated packages from the network, both from the local system and from a remote Web-based interface. The updates available from the Red Hat Network include security updates released on a timely basis.

Red Hat does release the source code of such packages under the GPL. However, it takes some time for third parties to reprocess those packages, remove trademarks, and collect them into a format suitable for installation CDs and DVDs. Perhaps more important, it takes some time for third parties to process the source code of security updates. Who knows what may happen to your systems during that time?

The Red Hat Network enables you to monitor the status of connected systems. It also allows you to control the channels of software available to subscribed systems. More important, it allows you to create scripts remotely to be run on those systems. Those scripts are not limited to updates. Scripts pushed from the Red Hat Network to subscribed systems can perform any administrative function. Red Hat scripts can be written even for Kickstart-based installations on bare metal hardware.



NOTE

Kickstart is a system that allows for automatic deployments of Red Hat installations. If you develop a specific configuration you want all of your systems to look like when they are installed, you can use Kickstart to push that out, either using a local installation medium like CD/DVD or over the network.

One problem associated with the Red Hat Network is that its servers are not mirrored. If many enterprises are downloading updates or DVD-sized ISO files, Red Hat Network servers can get overloaded. In addition, many enterprises prefer a higher degree of control. For these reasons, Red Hat offers two add-on products with the Red Hat Network: the Red Hat Satellite server and the Red Hat Proxy server.

Client/server registration on the Red Hat Network may happen during the installation process. If you want to register a newly created clone—say, of a virtual machine on the Red Hat Network—run the `rhn_register` command and follow the prompts that appear. You will need information from the governing Red Hat Network account.

For more information, navigate to <https://rhn.redhat.com>.

Red Hat Satellite Server

The **Red Hat Satellite server** effectively provides a local version of the Red Hat Network, suitable for enterprises. With a local copy of all associated Red Hat software packages, especially security updates, a Red Hat Satellite

server is not limited by restrictions on an Internet connection. Perhaps just as important, it provides local control over those packages that are downloaded and installed to client/server systems. This server includes an embedded Oracle database manager to monitor and maintain the packages on the systems managed through the server.

Red Hat Proxy Server

A **Red Hat Proxy server** works like a caching server. Update requests first check for available packages from the Red Hat Proxy server on the local network. If requested packages are not available locally, the request is forwarded to the Red Hat Network. Those packages are then copied to the Red Hat Proxy server and downloaded to the requesting client/server. Update requests from a second client/server can then access those same packages on the local Red Hat Proxy server.

Canonical Landscape

Canonical is the privately held corporate sponsor behind the Ubuntu distribution. Like Red Hat, Canonical tries to make money from its work on Linux. To that end, Canonical sells subscriptions to **Landscape**, its Web-based system-management service.

Landscape provides many of the same features as the Red Hat Network. As noted at <https://landscape.canonical.com/>, Landscape includes the following features:

- **Alerts**—Canonical sends security alerts to the e-mail addresses of users with Landscape accounts.
- **Package management**—Landscape enables deployment of one or many packages from the Web-based management tool.
- **System monitoring**—Landscape allows you to monitor the health of connected systems.
- **Cloud management**—Because Ubuntu now includes a release for the Amazon enterprise cloud, the Landscape system provides management tools for systems connected to that environment.

A subscription to Landscape is not required for a binary copy of an Ubuntu distribution CD or DVD. Nevertheless, the features associated with Landscape can be a timesaver for administrators who would otherwise have to create scripts to connect to and monitor remote clients/servers.

To connect an Ubuntu system to Landscape, run the following command to download appropriate packages with dependencies:

```
# apt-get install landscape-client
```

After the appropriate packages are installed, run the **landscape-client** command and follow the prompts that appear. You will need information from the governing Landscape account.

Novell's ZENworks

Novell offers its **ZENworks** platform as a multi-distribution system management tool. You can use it to manage systems that run both the SUSE Linux Enterprise Server and Red Hat Enterprise Linux distributions.

Novell's ZENworks Linux Management platform is a different but related product to its ZENworks system-management product for Microsoft operating systems. As with the Red Hat Network and Canonical Landscape, ZENworks can be used to administer Linux systems in groups. You can also use ZENworks to manage packages on one system or a group of systems.

Although ZENworks has a Web-based interface, it also supports systems management from the command line. For more information, see <https://www.novell.com/products/zenworks/endpoint-management/>.

Open Source Update Managers

If your resources are limited or if you prefer open source solutions on principle, alternatives are available. As with other open source solutions, the additional cost is in time.

Some of the tools are already available. Earlier in this chapter, you read about tools such as yum-cron and unattended-upgrades, which you can configure to automatically download and install updated packages. With appropriate changes, such updates can be limited to security-related updates. You can exclude certain packages from the update. If additional system monitoring is required, you may be able to set up logging to a centralized server.

Of course, you can configure and distribute scripts from a central interface to remote clients and servers. You can configure those scripts to run selected commands as needed. For systems where the SSH service is running, you can copy them directly to appropriate directories with the `scp` command.

In the big picture of update management, the key tools are the `apt - *` and `yum` commands. You can use them to update, install, remove, and otherwise manage the packages of your choice. For this reason, the following sections focus on these commands. The last section briefly describes the Red Hat Spacewalk open source system management solution.

Various `apt - *` Commands

The `apt - *` commands are a series of commands developed for Debian Linux. You can use these commands to install, remove, or provide additional information about packages available from distribution-specific repositories. Standard `apt - *` commands are summarized in [Table 11-2](#).

The `aptitude` command works as a front end to many `apt - *` commands. In addition, the `aptitude` command, when run by itself, opens a console-based configuration tool for Debian-style packages, including those found on Ubuntu systems.

TABLE 11-2 Standard `apt - *` commands.

COMMAND	DESCRIPTION
<code>apt-cdrom</code>	This includes CDs/DVDs (or appropriately mounted ISO files) for updates.
<code>apt-file</code>	This searches for files within uninstalled packages.
<code>apt-ftparchive</code>	This generates a package archive from a list of packages in a directory. It is used for creating key files on a local repository.
<code>apt-get</code>	This installs, updates, removes, and purges packages with dependencies.

Various `yum` commands

You can use the `yum` command to pull updates on Red Hat–style systems, including Red Hat Enterprise Linux, Fedora Core Linux, and CentOS Linux. The `yum` command is rich and diverse. This section addresses just a few basic options for the `yum` command. For a full list, run the `yum` command by itself. [Figure 11-2](#) illustrates just the basic options available for `yum`.

List of Commands:

```

check-update  Check for available package updates
clean         Remove cached data
deplist       List a package's dependencies
downgrade    downgrade a package
erase         Remove a package or packages from your system
groupinfo     Display details about a package group
groupinstall  Install the packages in a group on your system
grouplist     List available package groups
groupremove   Remove the packages in a group from your system
help          Display a helpful usage message
info          Display details about a package or group of packages
info-security Returns security data for the packages listed, that affects your system
install       Install a package or packages on your system
list          List a package or groups of packages
list-security Returns security data for the packages listed, that affects your system
localinstall  Install a local RPM
makecache     Generate the metadata cache
provides      Find what package provides the given value
reinstall     reinstall a package
repolist      Display the configured software repositories
resolvedep   Determine which package provides the given dependency
search        Search package details for the given string
shell         Run an interactive yum shell
update        Update a package or packages on your system
update-minimal Works like update, but goes to the 'newest' package match which fixes a problem that affects your system
upgrade      Update packages taking obsoletes into account

```

FIGURE 11-2

The **yum** command is powerful.

Here are some basic examples of the **yum** command. To install a new package named samba, you run the following command:

```
# yum install samba
```

If the Samba server package is already installed and an upgraded version is available, the preceding command updates the package with any dependencies. Alternatively, the following command updates the Samba package if it is already installed:

```
# yum update samba
```

If you are interested in installing a new kernel package, avoid the **update** option. The **install** option preserves the installation of any existing kernel side by side with any updated kernel.

The **update** (and related) option is powerful when updating a series of packages for a system. For example, the following command checks connected repositories for available updates:

```
# yum check-update
```

If the list of available updates is satisfactory, you can download and install all those updates with the following command:

```
# yum update
```

If you want to exclude a package from the list of updates, the **-exclude** switch can help. For example, the following command excludes the noted packages from an update:

```
# yum update --exclude=samba,samba-common,samba-client
```

Of course, you can use the **yum** command to uninstall a package, with dependencies. For example, the following command removes the Samba file server with any dependent packages:

```
# yum remove samba
```

You can also take advantage of how packages are organized on Red Hat distributions. For example, the following command lists available package groups:

```
# yum grouplist
```

If there is a group of packages to be installed, such as Server Configuration Tools, you can install all the mandatory and default packages in that group with the following command:

```
# yum groupinstall "Server Configuration Tools"
```

To identify the mandatory and default packages in any group, navigate to the /var/cache/yum/ directory. In a subdirectory associated with a repository, find the comps.xml file. The packages associated with each package group can be found in that file.

Red Hat Spacewalk

If you just need an open source system-management solution but don't want the trouble of writing and maintaining scripts for groups of systems, one emerging alternative is based on the **Spacewalk** project. It is being developed from the Red Hat source code associated with the Red Hat Satellite server described earlier.

Spacewalk requires access to different packages associated with the Extra Packages for Enterprise Linux (EPEL) project. Because the Red Hat Network includes Kickstart functionality for new installations, Spacewalk includes equivalent open source functionality from the **Cobbler project**. (The Cobbler project is an open source project for network-based installations of Linux distributions.) For more information, see <http://www.redhat.com/spacewalk/> and <https://cobbler.github.io/>.

Best Practices: Security Operations Management

This chapter focused primarily on the management of security updates. Because many Linux security issues are distribution specific, the developers behind the major distributions maintain their own security-alert systems.

Red Hat maintains a security announcements list and supports updates through its system management tool, the Red Hat Network. Canonical maintains a list of security notices, known as the USN, and supports updates through its system-management tool, Landscape. Because Canonical includes software associated with many applications, its security notices incorporate input from the developers of those applications. If you trust these developers, you may want to take advantage of available automated update tools.

If certain applications are mission-critical for your organization, you may want to monitor appropriate security lists maintained by their sponsoring groups. Just be aware, there are user applications such as the [OpenOffice.org](#) suite and Web browsers, along with service applications such as Apache, Samba, and vsftpd.

Linux has antivirus and anti-malware systems. Some of these systems are designed to minimize the risk of Linux as a malware carrier between Microsoft systems. Such systems also address malware written for Linux, such as rootkits, Trojan horses, worms, and more.

Sometimes, you will need to get into the details of associated bug reports. To that end, Ubuntu maintains its Launchpad system and Red Hat has a Bugzilla system for classifying and tracking bugs. Different user and service applications maintain their own systems.

If you know how to navigate these systems, you will have a better idea whether key developers are taking the problems that you see seriously. If you have done the research, do not hesitate to report a bug. If it is a security issue, some developers ask for private reports.

In the open source world, organizations such as ISECOM, the NSA, and the FSF work to improve security on open source software. If you believe you have identified a problem, be aware of the open source culture before reporting it. If you do a little research first, you may get the attention of prominent Linux developers and a faster solution.

There are several reasons to create a local update repository. The main reason is to reduce the load on the network's Internet connection. Other reasons relate to limiting software available to local systems. In addition, a local update repository enables you to test updates before they are installed on local systems.

If you just need to reduce the load on an Internet connection, solutions such as the Red Hat Satellite server and the Red Hat Proxy server are available. If you need to administer a substantial number of systems, centralized system-management tools such as the Red Hat Network, Canonical Landscape, and Novell's ZENworks can be helpful. Alternatively, if you prefer open source centralized management solutions, monitor Red Hat's Spacewalk project.



CHAPTER SUMMARY

This chapter describes what you can do with security alerts and updates. Security alerts are provided not only by the developers behind Linux distributions but also by the developers behind major Linux applications from the [OpenOffice.org](#) suite to the Apache Web server. The `apt -*` and `yum` commands are two ways to install such updates. Packages that automatically download and install updated packages are also available.

Alternatively, if you administer a variety of Linux computers, system-management solutions such as the Red Hat Network and Canonical Landscape can help you manage groups of systems. With these, you can monitor the status of such systems, pushing different sets of updates to these systems as desired.



KEY CONCEPTS AND TERMS

AVG Technologies

Berkeley Standard Distribution (BSD)

Bugzilla

Clam AntiVirus (ClamAV)

Cobbler project

Common Vulnerabilities and Exposures (CVE) list

Eric Raymond

Kaspersky Lab

Kickstart

Kubuntu

Landscape

Launchpad

Linux patch management

Main repository

Mandriva

Multiverse repository

Online Certificate Status Protocol (OCSP)

Pro File Transfer Protocol daemon (ProFTPD)

Pure File Transfer Protocol daemon (Pure-FTPD)

Rebuild

Red Flag Linux

Red Hat Network (RHN)

Red Hat Proxy server
Red Hat Satellite server
Red Hat security advisories (RHSAs)
Restricted repository
Rootkit
Solaris
Spacewalk
SpamAssassin
Spyware
SUSE
Trojan horse
Turbolinux
Ubuntu security notices (USNs)
Universe repository
Virtual Network Computing (VNC)
Virus
Washington University File Transfer Protocol daemon (WU-FTPD)
Worm
Xfce desktop environment
Xubuntu
ZENworks



CHAPTER 11 ASSESSMENT

- 1.** For at least how long does Red Hat provide security updates for its Enterprise Linux distributions?
 - A. Two years
 - B. Five years
 - C. Seven years
 - D. Ten years
- 2.** For Ubuntu's LTS releases, Canonical provides security updates for its server distribution releases for at least five years.
 - A. True
 - B. False
- 3.** Why would you read a security alert and not just download and install a security update to a key system such as the Linux kernel? (Select three.)
 - A. The update may affect interactions between the operating system and local hardware.
 - B. The update does not affect any systems that you use personally.
 - C. The update may not be bootable.
 - D. The update relates to Xen, which is a special kernel not used on the local system.
- 4.** Which command is commonly used to install and update packages from the command line on Red Hat systems?
 - A. yum
 - B. apt
 - C. get
 - D. update

- 5.** Which of the following types of malware is *not* found on Linux?
- A. Rootkits
 - B. Microsoft viruses
 - C. Trojan horses
 - D. None of the above
- 6.** Which of the following includes a system for tracking bugs in software?
- A. ZENworks
 - B. Red Hat Network
 - C. Landscape
 - D. Launchpad
- 7.** Before creating a bug report, which of the following actions should you take?
- A. Copy all log files to the report.
 - B. Research any FAQs.
 - C. Reinstall the software.
 - D. Reboot the system.
- 8.** Which of the following files contain the addresses of remote repositories?
- A. /etc/apt/sources.list
 - B. /etc/apt/apt.conf
 - C. /etc/apt.conf
 - D. /etc/apt/apt.conf.d/10periodic
- 9.** The multiverse repository includes packages that are not supported and do not include open source software.
- A. True
 - B. False
- 10.** Which of the following files in the /etc/apt/apt.conf.d/ directory determine whether unattended upgrades are run?
- A. apt.conf
 - B. 10periodic
 - C. 50unattend-upgrades
 - D. 99update-notifier
- 11.** Which of the following configuration files is associated with unattended upgrades on Red Hat Enterprise Linux systems?
- A. /etc/yum.conf
 - B. /etc/yum/yum-daily.yum
 - C. /etc/yum/yum-updatesd.conf
 - D. /etc/yum.repos.d/yum-updatesd.conf
- 12.** Which of the following system-management services is open source?
- A. Red Hat Network
 - B. Landscape
 - C. Spacewalk
 - D. All of the above

CHAPTER

12 Building and Maintaining a Security Baseline

T'S IMPORTANT TO TAKE SOME TIME to review the basic steps required to create an appropriate baseline system for both the Ubuntu and Red Hat distributions. A good baseline keeps installed software to a minimum. In addition, a good baseline is secure, with firewalls and other security controls that block unauthorized access from both external and internal sources.

Some secure baseline configurations include read-only filesystems. That is one reason for the popularity of live CD distributions. You can boot such media on most systems, with a fully functional version of Linux loaded into system random access memory (RAM). In this chapter, you will examine several options for live CDs with security-management tools. After these are loaded onto a system, you can use such tools on local filesystems. These filesystems do not even have to be mounted.

You can monitor the performance of a system baseline, along with production systems created from that baseline, through appropriate log files. If you administer a group of systems, you will want to collect logs from those systems. Log files are just a static snapshot of the performance of a system, however. System performance also depends on dynamic measures. These include system load, network status, currently running services, and more. You can check the integrity of a baseline and systems derived from that baseline with integrity scanners.

Chapter 12 Topics

This chapter covers the following topics and concepts:

- How to configure a simple baseline
- What read-only and bootable operating systems are
- How to keep a baseline system up to date
- How to monitor local logs
- How to consolidate and secure remote logs
- How to identify a baseline system state
- How to check for changes with integrity scanners
- What best practices are for building and maintaining a secure baseline

Chapter 12 Goals

When you complete this chapter, you will be able to:

- Create and maintain a secure baseline Linux system
- Use bootable operating systems to your advantage
- Configure logs locally and remotely
- Scan a system for integrity against malicious users

Configuring a Simple Baseline

In this section, you will review appropriate parts of the installation programs for the Red Hat and Ubuntu distributions. When configuring a simple baseline, you must start with these programs. To that end, it's important to have a basic understanding of how best to take advantage of these programs.

The Red Hat installation program is called **Anaconda**. It has been used to install Red Hat systems for more than a decade. The Ubuntu installation program was developed from the one used for Debian Linux, which has an even older heritage. In general, the text-mode installation options for both programs are more flexible.

Normally, you can boot the installation program from a CD, a DVD, or a USB drive. Alternatively, you can install the installation program over a network using a basic input/output system (BIOS) or Unified Extensible Firmware Interface (UEFI) connection to a network card configured with a Preboot Execution Environment (PXE).

A Minimal Red Hat Baseline

When you boot into the installation media for Red Hat Enterprise Linux (RHEL) or CentOS, you have with a choice: to perform an installation or to test the media from which you are booting. Developers introduced this option many years ago because users were burning their own installation discs. These discs—either DVD or CD—have not always been reliable because of the quality of the physical media or of the burner used to create it. To protect against getting partway through the installation and having it fail due to read errors from failing media, the Red Hat installer has long offered to test installation media. Unless you intervene before this step, you will be brought into a graphical installer. To avoid reaching that stage, you must press the Esc key when you get to the menu shown in [Figure 12-1](#).

When you press the Esc key, you are presented with a prompt that says **boot :**. At this point, you can specify what you want to boot. Normally, you might type **linux** to load the kernel. In this case, though, you want to provide a parameter to indicate that you are not going to use the normal GUI installer. Instead, you want to do a text-based install. You will not load any of the drivers for a framebuffer or graphics card because you are just going to do a basic text-based installation. The prompt and the command you type look as follows:

```
boot: linux text
```

After the kernel loads, the installer starts. This is a different text-based installation than has been previously used in Red Hat or CentOS installations. Instead of a curses-based system, which presents menus and options onscreen that you can tab through to make different selections, involving some rudimentary graphics, this is a purely text-based installation. [Figure 12-2](#) shows the new top-level menu.



FIGURE 12-1

Initial installation screen.

```

Starting installer, one moment...
anaconda 19.31.123-1 for CentOS 7 started.
* installation log files are stored in /tmp during the installation
* shell is available on TTY2
* when reporting a bug add logs from /tmp as separate text/plain attachments
21:26:30 Not asking for VNC because we don't have a network
=====
=====
Installation

1) [x] Language settings
    (English (United States))
3) [!] Software selection
    (Processing...)
5) [x] Network settings
    (Not connected)
7) [x] Kdump
    (Kdump is enabled)
9) [!] Set root password
    (Password is not set.)
Please make your choice from above ['q' to quit | 'b' to begin installation |
'r' to refresh]: _

[anaconda] 1:main* 2:shell 3:log 4:storage-log 5:program-log

```

FIGURE 12-2
Top-level installation menu.

This is similar to what you would get in a graphical installation, but instead of buttons that you click on the screen, you get numbered options. You simply type the number of the item for which you want to provide information so the installation can proceed. In this case, you are looking for a baseline that you can use as a generic image from which to create your systems.

If you choose Software Selection, you will see the list of options in [Figure 12-3](#). The best choice here is Minimal Install. This gives you the smallest number of packages you need just to get the system up and running. It will not do much, but you will be able to administer it, including adding more packages.

After you have provided information, the installer will helpfully inform you if it needs more information by showing an exclamation point (!) in brackets next to the number that you would type to select that option. If the installer has everything it needs, you will see an X there. If you see an exclamation point, select that option and provide the information the installer needs. When you have cleared all of the exclamation points, you can proceed with the installation. After you finish the installation and reboot, you will have a basic Linux system that you can use as a baseline install.

```

Base environment

1) [x] Minimal Install           7) [ ] Server with GUI
2) [ ] Compute Node             8) [ ] GNOME Desktop
3) [ ] Infrastructure Server    9) [ ] KDE Plasma Workspaces
4) [ ] File and Print Server   10) [ ] Development and Creative Work
5) [ ] Basic Web Server          station

6) [ ] Virtualization Host

Please make your choice from above ['q' to quit | 'c' to continue |
'r' to refresh]: 7
=====
=====

Software selection

Base environment

1) [ ] Minimal Install           7) [x] Server with GUI
2) [ ] Compute Node             8) [ ] GNOME Desktop
3) [ ] Infrastructure Server    9) [ ] KDE Plasma Workspaces
4) [ ] File and Print Server   10) [ ] Development and Creative Work
5) [ ] Basic Web Server          station
6) [ ] Virtualization Host

Please make your choice from above ['q' to quit | 'c' to continue |
'r' to refresh]: _
```

[anaconda] 1:main* 2:shell 3:log 4:storage-log 5:program-log

FIGURE 12-3

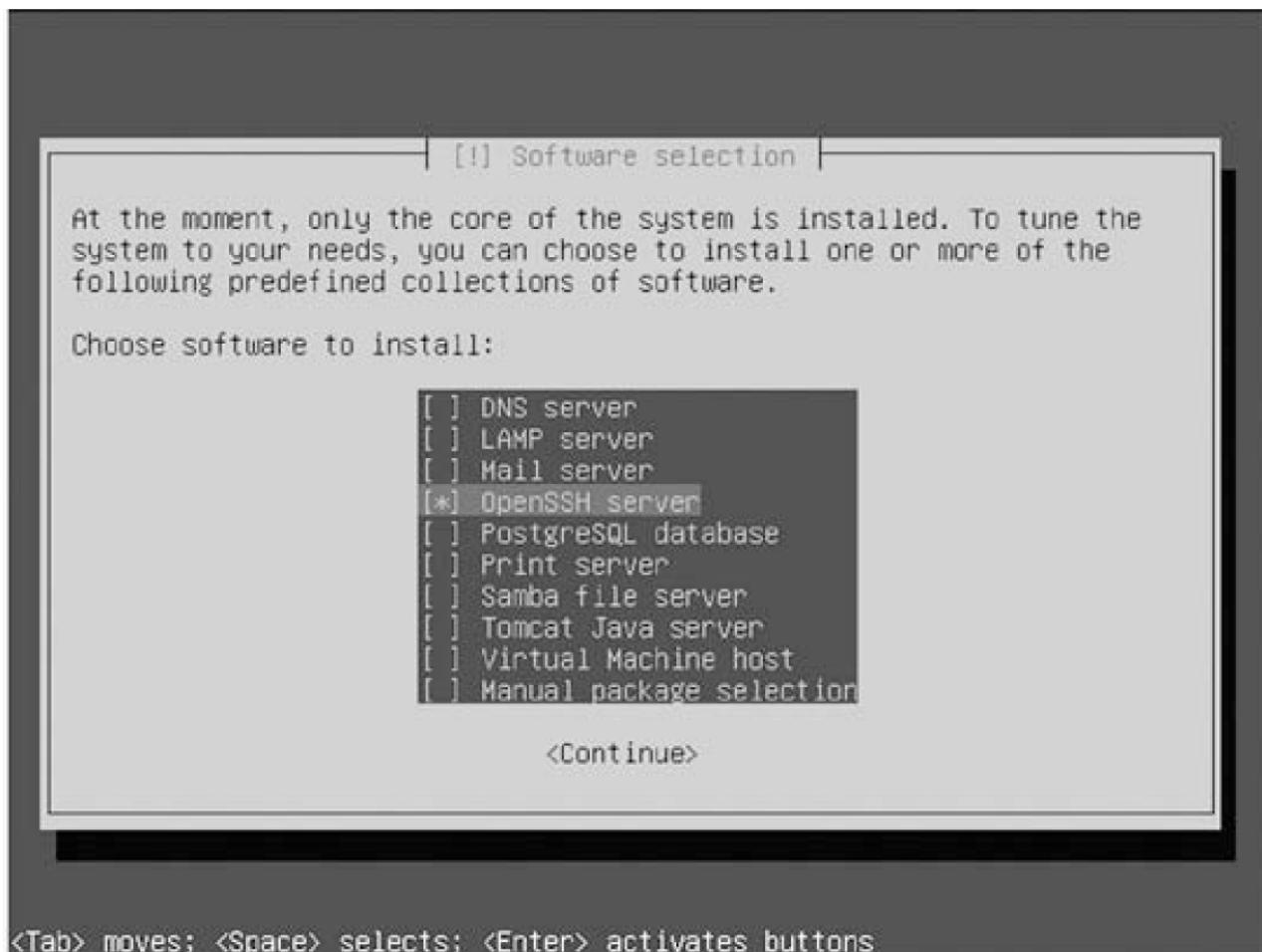
The Software Selection options.

A Minimal Ubuntu Baseline

This section assumes you are using a somewhat recent release of Ubuntu Server Edition. Fortunately, the details of the text-mode installation program have not changed significantly during recent releases. Yes, you can install Ubuntu Server Edition from the DVD. However, if you instead boot from the Server Edition CD, it automatically starts text mode of the Ubuntu installation program. Because incremental changes still happen, however, this description is subject to change.

After you have configured standard options like the language and keyboard, a base Ubuntu installation will be installed. When that process is complete, you will have the following options with respect to how upgrades are managed:

- **No automatic updates**—This is the default. With this, additional configuration may be required.
- **Install security updates automatically**—This configures security using the 50unattend-upgrades file in the /etc/apt/apt.conf.d/ directory. If you are concerned that updates may adversely affect systems so configured, this option may not be suitable for you.
- **Manage system with Landscape**—This installs the Landscape client package on the local system. It assumes you have a subscription to Landscape with a sufficient number of entitlements.

**FIGURE 12-4**

Ubuntu software selection options.

During the installation process, you will be given a series of software selection options, as shown in [Figure 12-4](#). You might only need to select the OpenSSH server option for the Secure Shell (SSH) until you have your baseline complete. OpenSSH will allow you to remotely manage the system once it's up and running. If you have chosen the Manage System with Landscape option just described, you may not even need to install the OpenSSH option, as Landscape supports remote configuration and commands on connected clients.

Read-Only or Live Bootable Operating Systems

You may be concerned about users creating or moving files to places in the filesystem where you would rather they did not write to. Permissions can help here, but an even more direct way to prevent such writes would be to mount filesystems in read-only mode whenever possible. The other way to preserve an operating system is to store it on read-only media such as a DVD.

Linux includes a wide variety of distributions that you can boot directly from CD or DVD. After you have the system booted, you will have virtual filesystems that live in memory and that can be altered as long as the system is running. When you shut the system down, you lose any data you have stored on these virtual drives. This includes any files you as a user have created, as well as any system files created, like logs. Starting back up from the disk creates a completely clean, new copy of the whole filesystem.

Appropriate Read-Only Filesystems

Several filesystems need to be writable for a working operating system, including /home/ for user home directories, /var/ for log files and spools, and /tmp/ for systems where the GUI is in operation. In many cases, you may also set up the /usr/ directory as a read-only filesystem until updates are required.

This list omits those directories that are generally included as part of the top-level root directory (/) filesystem, including /bin/, /dev/, /etc/, /lib/, /root/, /sbin/, and /sys/. Although some of these directories can work in read-only mode, they all need to be part of the top-level root directory filesystem for a Linux system to be bootable.

Directories like /dev/ and /sys/ are dynamic. In other words, files are written to these directories during the boot process and while certain types of hot-swap hardware are attached to the system. Even for a directory such as /usr/, however, it can be a pain to make that filesystem read-only.

One alternative is to mount the noted directory with as many limits as possible. For example, it can be helpful to add the following mount options to certain filesystems:

- **nosuid**—This disables super user ID (SUID) and super group ID (SGID) permissions on files in the local filesystem.
- **noexec**—This disables executable binaries on files in the local filesystem.
- **nodev**—This does not allow block or special character files such as partition devices and terminals in the local filesystem.

You may want to consider options in the /etc/fstab configuration file for several different directories, as described in [Table 12-1](#). These may be very beneficial in preventing attacks.

TABLE 12-1 Special mount options to apply to certain directories.

DIRECTORY MOUNT OPTION AND DESCRIPTION	
/boot	Include noexec and nodev to avoid executable files and partitions with inappropriate mount options.
/home	Include nosuid and nodev to avoid creating files that others can execute and partitions that can be opened with inappropriate mount options.
/tmp	Include noexec and nodev to avoid placing executable files into /tmp that could be used for illicit purposes since this directory is accessible by all users
/var	Include nosuid to avoid copying files owned by one user that others can execute with the permissions of another user.

Live CDs and DVDs

A Linux live CD/DVD is a fully functional version of a Linux distribution. When booted, a live CD or DVD loads a fully functional version of Linux into the RAM of a local system. Because it's independent of any local hard-drive media, it's especially well suited to diagnosing problems with other connected physical media.

A number of groups of Linux developers create live CDs and DVDs. They are available for a number of standard distributions, including Ubuntu, Fedora, SUSE, CentOS, and many more. A more complete list is available at <http://www.livecdlist.com/>. The first and perhaps the most prominent live CD-based distribution is Knoppix. That distribution has gained prominence as one with a variety of rescue and recovery tools.

With the diversity of live CD/DVDs available, one surprise is the lack of a live CD option—at least for RHEL 7. Creating a live CD of a distribution is not that difficult, however. CentOS 7 has built a live CD from RHEL 7 source code. For a somewhat complete list of the major options for live CD distributions, see <http://www.livecdlist.com/>.

You can boot these fully functional Linux operating systems from a CD/DVD drive. The BIOS/UEFI of a system can be set to automatically boot from that drive. With a couple of small changes, you can write log file information to a portable storage device such as a remote server or a USB key. Alternatively, you can configure logs to write to a log server on the network. In some cases, this sort of setup makes for a good firewall installation. No files stored on the

system will be kept past a reboot. There is also no need for a lot of disk space. A number of distributions are developed to function as a firewall or intrusion detection system (IDS), designed to boot from a CD or DVD.

The weakness of some of these live CD systems is that they include too many active services, often with a GUI desktop environment. The strength of such systems is that they frequently include administrative and security-related tools that can help identify and diagnose problems on a troubled system. They are great to have around in case you have a failure on your system. You can boot to the CD, mount your primary drive, and make any changes necessary, and then just boot back to your primary operating system from disk.



TIP

If you do not have a Linux distribution on removable media, you might consider looking into getting one to keep around, just in case.

Keeping the Baseline Up to Date

When you have a baseline Linux system, it's something worth protecting. It's a secure system, ready to copy to different machines, real and virtual. All you need to do to make the baseline into a fully functional server or a desktop environment is to add the programs required for the desired functionality. The `tasksel` command makes this process easy on Ubuntu systems. The `yum group list` command makes the process almost as easy on Red Hat systems. There is a learning process with respect to the baseline, however. Experience with production systems will teach you about things that should be added to—or left out of—the baseline.

A Gold Baseline

A gold baseline configuration is a place to start. With a good gold baseline, you will not need to run through the installation process repeatedly. If you configure the gold baseline as a virtual machine, all you will need to do is copy the virtual machine files from one directory to another. If it's a physical system, the process is a little more involved, but you can use a command like `dd` to copy everything from the partition(s) of one drive to the partition(s) of a second physical hard drive.

A gold baseline should have the following characteristics:

- **Simplicity**—No package should be installed on the system unless necessary for smooth operation and monitoring.
- **Network security**—Traffic is blocked to all ports on the system except those necessary to maintain basic network connections. If you administer a system with SSH, the associated port (port 22) should be open as well.
- **Mandatory access controls**—A system such as SELinux or AppArmor should be enabled to minimize risks if one or more account is compromised.

You may want multiple baseline configurations to reflect separation in different filesystems, such as a separate `/home/`, `/var/`, or `/tmp/` directory. To make that baseline work on a server, you will need to make sure that system is up to date with the latest changes related to security and functionality. Every change involves risk, however. In general, when you know a change works on your systems, you should delete the original packages. Such packages are normally stored locally in subdirectories of `/var/cache/yum/` and `/var/cache/apt/`. If an attacker can see what updates are installed, that person can start to identify weaknesses on those systems.

Of course, if you have created any sort of server that has an application like a Web server from that baseline, you should also update those servers based on the guidelines for security and functional updates that follow. When you make a change to the baseline, especially when it involves removing packages or making significant configuration changes, you need to have a way to make sure all of your already-installed systems pick up those changes.

In addition, if users are currently connected to servers created from a baseline, take care. With many services controlled by scripts in the /etc/init.d/ directory, it's possible to apply the `reload` option to reread any configuration files that may have been changed by an update. For example, if you have just installed a recent security update to the Samba server, you should make sure such changes have not affected custom settings in the associated smb.conf configuration file in the /etc/samba/ directory. If you do make subsequent changes to the smb.conf configuration file, you can minimize the inconvenience to users with the following command:

```
# /etc/init.d/samba reload
```

Depending on the distribution, you may need to substitute `smb` for `samba` in the command shown.

There may be good reasons to avoid installing an update. In fact, every update may be a risk, as described in the sections that follow. Of course, the developers behind major distributions such as Red Hat and Ubuntu do their best to minimize such risks. Nevertheless, if that risk is too large, you may need to revise options associated with local or remote repositories. Such changes can prevent updates from being installed on local systems.

Security Updates

It's easy to say, "Just install all security updates." Some updates, however, may cause problems with some hardware, software, or other installed components. For this reason, you should test all updates—even security updates—before implementing them on production systems.

Security updates related to the Linux kernel may be especially troublesome. Some third-party software such as database managers may be certified only to certain versions of the kernel. The installation of a new version of the Linux kernel—even if it addresses a security issue—may affect any support agreement associated with that software. In that case, you may need to address the issue with the third-party software vendor.

Other issues relate to software compiled against the source code for one version of a kernel. For example, if you have installed virtual machine software, a new version of a kernel may require you to recompile that software with the revised source code.

If a security update causes problems, additional work is required. First, you need to know the reason for the update. For example, if the update relates to Samba interaction with shortwave radio hardware, and local systems do not have such hardware, an update might not be required. On the other hand, if the update relates to something that is done on a regular basis on local systems, you will need to make a choice. Ask yourself the following:

- Is it acceptable to disable the service with the security issue?
- Is it a mission-critical service for your organization?
- Does the security issue relate to any of the issues associated with the confidentiality, integrity, and availability (C-I-A) triad?
- Does the security issue put at risk any commitments to customers?
- Does the security issue put you or your organization at risk of violating a legal, regulatory, or industry standard?

If you can answer no to all of these questions, it may be possible to continue running the service in question. Such cases are rare, however.

If a security update passes your tests, it should be included in gold baseline systems. That way, when such systems are used to create any type of deployed server, the security updates are already included.

Functional Updates

Functional updates do not themselves respond to security issues. To determine whether a functional update enhances or hurts security, you can refer to the concepts associated with the C-I-A triad. Updates such as those that change the default wallpaper on a desktop environment do not enhance security. However, such updates may be encouraged for other reasons, such as corporate policies (for example, if that wallpaper includes a corporate logo).

It may be instinctive to install a functional update. Functional updates can lead to other problems, however. Reasons to avoid the installation of such an update include the following:

- **Kernel updates**—Some kernel updates only enhance the functionality of a system. However, the same issues relating to third-party software certification and compiled software as described earlier may also apply.
- **Interactions with other software**—The developers behind the update may have tested their revisions. The developers behind related software, however, may not have a chance to do their testing. If you depend on that other software, updates could cause problems.
- **No need**—If you do not need the additional functionality, why take the risk?

Baseline Backups

It's important to preserve a baseline configuration as if it were gold. A baseline configuration is the starting point for other services. A baseline configuration can save time when compared to a new installation. Before installing an update to a gold baseline, make sure that baseline is backed up. While methods are available to uninstall backups, such methods can be time consuming.

Monitoring Local Logs

When properly configured, log files are stored in the `/var/log/` directory or subdirectories. There are two services associated with Linux operating system logs: the system log (`syslog`) and kernel log services. You may find different implementations of the `syslog` function installed by the distribution you have chosen. Newer systems use the newer `rsyslog` service by default, which supports secure tunneled connections and database management.

The System and Kernel Log Services

The system and kernel log services originated as two different services. However, their functionality has been combined in a single package on most systems. On both Red Hat and Ubuntu systems, the package is named `sysklogd` and is configured in the `/etc/syslog.conf` file.

When taken together, these services classify log messages in two areas: the facility, which is the logging subsystem; and the priority, which is the importance of the log message. [Table 12-2](#) describes log facilities. Not all log facilities are included in [Table 12-2](#), however. For example, the `lpr` facility is associated with the now rarely used Line Printer next generation (LPRng) service. The Common Unix Printing System (CUPS) has its own logging configuration. The `news` facility is related to systems that maintained older newsgroups through the Network News Transfer Protocol (NNTP). The `uucp` facility was based on remote commands. Its functionality has been taken by `scp` and other programs.

Log message priorities are classified in the following order of importance:

- **debug**—The `debug` priority provides the greatest detail. It's used for troubleshooting.
- **info**—The `info` priority adds logging notes at the information level.
- **notice**—The `notice` priority includes messages that might require attention.
- **warn**—The `warn` priority provides warning messages. It may also be shown as `warning`.
- **err**—The `err` priority adds error messages. It may also be shown as `error`.
- **alert**—The `alert` priority specifies problems that justify an alert and require immediate attention.
- **emerg**—The `emerg` priority is for important messages. It may also be shown as `panic` or `crit`.

There is also an implicit priority of `none`, which negates log messages from the associated facility.

TABLE 12-2 Linux log facility subsystems.

FACILITY	DESCRIPTION
<code>auth</code>	These include login authentication messages.
<code>authpriv</code>	

cron	This adds messages from the cron daemon, typically from /etc/crontab.
daemon	This includes messages associated with the status of services.
kern	This includes messages associated with the Linux kernel.
mail	This adds messages associated with running e-mail services.
syslog	This lists internal syslog service messages.
user	This relates messages from user-mode programs.

System log files organize log messages in *facility.priority log_file* format. For example, the following directive sends all messages of an **info** priority (or higher) but excludes mail log messages. The directive also sends emergency kernel messages to the /var/log/messages file:

```
*.info;mail.none;kern.emerg /var/log/messages
```

Log messages are commonly rotated on a weekly basis, as configured in the /etc/logrotate.conf file. When rotated, older log files are given a numeric filename extension, such as .1, as defined in /etc/logrotate.conf. Messages from these older services can be sent to remote systems. For example, the following line would send all previously noted messages to the system with a hostname of logserver. That logserver system should be configured with a matching configuration that would receive log messages from remote systems.

```
/etc/syslog.conf file to receive such messages. *.info;mail.none;kern.emerg @logserver
```

The weakness of this directive is that log messages are sent over the network in cleartext. You can address this problem with the rsyslog service, described shortly.

The Ubuntu Log Configuration

The default /etc/syslog.conf file for Ubuntu systems is fairly well organized. As noted, not all the facilities configured in this file are still significant. This section ignores such directives.

The first default directive is straightforward. It sends all authentication messages to the /var/log/auth.log file:

```
auth,authpriv.* /var/log/auth.log
```

The next directive sends all messages from all facilities of all priorities except authentication messages to the /var/log/syslog file. The dash (-) in front of the file means that logging messages are not synchronized every time. In other words, although some log messages may be lost in the event of a system crash, the frequency of logging messages does not affect system performance.

```
*.*;auth,authpriv.none -/var/log/syslog
```

Additional directives in the file are relatively self-explanatory. For example, the next active line sends all messages related to the starting and stopping of a daemon to the /var/log/daemon.log file.

```
daemon.* -/var/log/daemon.log
```

The remaining directives in this file are straightforward. For example, the following directives send kernel log messages to the kern.log file; log messages from e-mail servers to the mail.log file; and user-mode log messages to the user.log file, all in the /var/log/ directory:

```
kern.* -/var/log/kern.log mail.* -/var/log/mail.log user.*  
↳ -/var/log/user.log
```

The following messages divide logging messages from e-mail servers into three categories. By default, log messages of the given priority and higher are sent to the given log file. Therefore, the final directive would contain the least amount of logging information:

```
mail.info -/var/log/mail.info mail.warn -/var/log/mail.warn mail.err
↳ /var/log/mail.err
```

The two directives that follow demonstrate the backslash (\), which escapes the meaning of the carriage return. In other words, with the backslashes, the following three lines are combined into a single directive, which sends all **debug**-level messages but nothing related to the **auth**, **authpriv**, **news**, and **mail** facilities to the /var/log/debug file. **=debug** refers to messages at only the **debug** level and nothing of a higher priority.

```
*.=debug; \
auth,authpriv.none; \
news.none;mail.none;\ -/var/log/debug
```

The next four lines are also a single directive, which sends all messages of the **info**, **notice**, and **warn** priorities to the /var/log/messages file. Messages from the facilities with the .none extension are not included in the list:

```
*.=info;.=notice;.=warn;\ auth,authpriv.none;\ cron,daemon.none; \
mail,news.none;\ -/var/log/messages
```

The Red Hat Log Configuration

Although the default Red Hat log configuration file may appear to be less well organized, it simply reflects a different set of priorities with respect to logging information. First, the following line sends all **info**-level messages to the /var/log/messages file:

```
*.info;mail.none;authpriv.none;cron.none/var/log/messages
```

Messages related to all login attempts through the **authpriv** facility are sent to the /var/log/secure file:

```
authpriv.* /var/log/secure
```

All messages related to e-mail servers are sent to the /var/log/maillog file:

```
mail.* -/var/log/maillog
```

Information related to the execution of cron jobs are sent to the /var/log/cron file:

```
cron.* /var/log/cron
```

All users get emergency log messages, which frequently indicate that a system is about to shut down. The asterisk (*) indicates that messages are sent to all consoles.

```
*.emerg *
```

Logs from Individual Services

A number of services have their own individual logging systems. Although the /var/log/ directory includes individual files that collect logs from standard logging services, pay attention to the subdirectories. Many of those /var/log/ subdirectories include log files from certain services.

In the following sections, you will explore just three services with custom log files configured in their own subdirectories. Although the formats of these log files are consistent with what is defined in /etc/syslog.conf and /etc/rsyslog.conf, they are different logging services. The three services selected are just examples; other services with /var/ log/ subdirectories may have their own log files defined in their main configuration files.

CUPS Logs

Three log files are defined by default on CUPS systems: access_log, error_log, and page_log. Generally, log information is set to either the `info` or `warning` level with the following directive:

`LogLevel warning`

Available levels of log information fit within the standard hierarchy described earlier. The names of the log files may be slightly misleading:

- **access.log**—This provides information on CUPS access of printers over the network.
- **error.log**—This includes information on configuration messages with and without errors.
- **page.log**—This specifies successful access to printers from local or remote systems.

Apache Logs

Depending on the distribution, you may find Apache log files either in the `/var/log/httpd/` or the `/var/log/apache2/` directory. The actual names of the log files are subject to custom configuration in the appropriate configuration files associated with regular and secure Web sites. Key directives in these files include the following:

- **ErrorLog**—This specifies the log file to receive error messages.
- **CustomLog**—This is used to log requests, usually coupled with a `LogFormat` directive to specify the information to be recorded.
- **TransferLog**—This specifies a log file based on the previously defined `CustomLog` format or a standard common log format.

TABLE 12-3 Apache LogFormat options.

OPTION	DESCRIPTION
%a	Remote IP address
%D	Time to serve the request, in microseconds
%f	Filename served
%oh	Hostname of the remote system
%l	Login name of the remote user
%t	Time of the request
%u	Remote username
%oU	Requested uniform resource locator (URL)
%v	The name given to the Web site serving the request, normally of the virtual host

Perhaps the most valued part of an Apache log file is based on the `LogFormat` directive, which can collect all sorts of information on users who connect to Web sites configured on the target Apache server. Some significant `LogFormat` options are described in [Table 12-3](#).

This is just a fraction of the information that can be logged from each user click on a Web site. The amount of information from all user clicks collected on your Web sites can easily grow quite large. That information may be quite useful in deciphering the behavior of your users, however.

Samba Logs

Samba is the name of the Linux service for connecting to Microsoft-based networks. Most log files associated with Samba are driven from the following directive in the main Samba configuration file, `/etc/samba/smb.conf`:

```
log file = /var/log/samba/log.%m
```

The `%m` is translated to the hostname or, if a Network Basic Input/Output System (NetBIOS) server is not available, the Internet Protocol (IP) address of the machine that connects to the local Samba server. For example, if the Samba server is on the 192.168.0.0/24 network, you might see a log file such as `log.192.168.0.10` in the `/var/log/samba/` directory. Regular and error messages are sent to that log file. You might see messages related to successful and failed connections in that file. Some machine-specific log files in that directory might be empty, indicating simply that the given system was detected as a browseable server on the local network.

Consolidating and Securing Remote Logs

The rsyslog service builds on the original system and kernel log services, including its basic configuration file, `/etc/syslog.conf`. The weakness of that older service is that it sends logging information to remote or central servers in cleartext. In other words, a malicious user who wants to collect information on the current state of your systems could have a field day if he or she can identify the system configured as a central logging server—unless that service is configured with the rsyslog service. Although the rsyslog service is configured primarily in the `/etc/rsyslog.conf` file, the way the service is started by default is rather important.

Default rsyslog Configuration

While the strength of rsyslog comes as a remote server, remote transmission of logging information is disabled by default. Generally, you should install at least the rsyslog and stunnel packages on client and server systems. If you want encryption, two basic choices are available in separate packages, based on Transport Layer Security (TLS) and the Generic Security Services Application Programming Interface (GSSAPI). On Red Hat and Ubuntu systems, these options are available in the `rsyslog-gnutls` and `rsyslog-gssapi` packages, respectively.

How the rsyslog service starts is driven by a central configuration file, `rsyslog`. The location of that file varies by distribution. On Red Hat systems, it's in the `/etc/sysconfig/` directory. On Ubuntu systems, it's in the `/etc/default/` directory. Depending on the release, the applicable variable in the `rsyslog` file is either `RYSLOGD_OPTIONS` or `SYSLOGD_OPTIONS`. To some extent, the name of the variable does not matter, as both are given as options to the `rsyslog` daemon when it starts.



TIP

If you want to use compatibility-mode options, substitute the `-c 2` option on Red Hat systems or `-c 0` on Ubuntu systems for the value of the aforementioned variable.

The Standard rsyslog Configuration File

The rsyslog configuration file is normally divided into three sections: modules, global directives, and rules, which normally incorporates the basic rules described earlier for the `/etc/syslog.conf` file. As described at http://www.rsyslog.com/doc-rsyslog_conf_modules.html, modules fall into a number of categories. Global directives set parameters such as timestamps and permissions.

rsyslog Modules

You can configure modules with the rsyslog service in a number of areas. First, the default modules are straightforward, as they enable local system and kernel logging:

```
$ModLoad imuxsock
$ModLoad imklog
```

Two additional modules are available in comments, based on communication through either the User Datagram Protocol (UDP) or the Transport Control Protocol (TCP).

Port 514 is the default for the system logging service. If you have replaced it with rsyslog, that port would be available:

```
#$ModLoad imudp
#$UDPServerRun 514
```

If you want to set up the local system as an rsyslog-based logging server, enable these two directives, as TCP communication is required for encrypted transmission of log files:

```
#$ModLoad imtcp
#$InputTCPServerRun 514
```

Some documentation suggests the use of port 61514 (in place of 514) to avoid conflicts with any remaining older system-logging daemons. Whatever port is selected should be considered in any firewall between rsyslog clients and servers.

Other rsyslog modules can be defined in the following categories:

- **Input**—Other input modules of interest relate to TCP communication, GSSAPI communications, and more.
- **Output**—Output modules process messages, especially for databases.
- **Parser**—Parser modules process received messages.
- **Message modification**—Message modification modules are still in development.
- **Library**—Users can configure library modules.

rsyslog Global Directives

Because rsyslog is relatively new, the global directives are more complete in the versions released for Ubuntu Hardy Heron and RHEL 6. The following standard **\$Action** directive uses the traditional format for log message timestamps:

```
$ActionFileDefaultTemplate RSYSLOG_TraditionalFileFormat
```

There are nearly 50 different **\$Action** directives available. These can configure everything from log file size to timeouts to frequencies. One handy option is to filter duplicate messages, which can reduce the load when a system makes the same mistake every few seconds.

The default setting shown here sends all log messages to the local system over a TCP connection using the suggested port of 61514. In these directives, while the double at symbol (@@) is associated with TCP packets, the single at symbol (@) is associated with UDP packets.

```
*.* @@127.0.0.1:61514
```

Of course, if this is a client system sending logging information to a remote server, you would change the 127.0.0.1 IP address to that of the logging server.

TABLE 12-4 Configuration rules from /etc/rsyslog.conf.

RULE	DESCRIPTION
\$FileOwner syslog	This sets the user owner of associated files to the syslog user.
\$FileGroup adm	This sets the group owner of associated files to the syslog user.
\$FileCreateMode 0640	This assigns 640 privileges to files created through the rsyslog service.
\$DirCreateMode 0755	This assigns 755 privileges to directories created through the rsyslog service.
\$Umask 0022	This sets the noted value for umask. It may supersede the \$FileCreateMode and \$DirCreateMode variables.

<code>\$PrivDropToUser syslog</code>	This assigns a user associated with the rsyslog service.
<code>\$PrivDropToGroup syslog</code>	This assigns a group associated with the rsyslog service.

rsyslog Configuration Rules

Some configuration rules set basic parameters for log files. For example, the following directives from the Ubuntu Lucid Lynx version of the /etc/rsyslog.conf file should be conceptually familiar from other services, as they set ownership and permissions on related files. [Table 12-4](#) describes the rules from that version of the configuration file.

Developers of rsyslog recommend that administrators use the `$PrivDropUserID` and `$PrivDropGroupID` directives rather than the `$PrivDropToUser` and `$PrivDropToGroup` directives. If the syslog user does not exist, the `$PrivDropToUser` syslog and `$PrivDropToGroup` syslog directives would mean that rsyslog is run with root administrative privileges. In that situation, if a malicious user were to break into the syslog account, that person would get full root administrative privileges on the local system. Alternatively, you could just be careful to make sure the syslog user and group are properly configured in the password-authentication database.

rsyslog Incorporates syslog Configuration Rules

There are two ways to incorporate configuration rules from the /etc/syslog.conf file. These rules could be included directly into the /etc/rsyslog.conf file. Alternatively, they could be included from smaller files with a directive such as the following:

```
$IncludeConfig /etc/rsyslog.d/*.conf
```

This directive includes the contents of any *.conf files in the /etc/rsyslog.d/ directory. On an Ubuntu Lucid Lynx system, the 50-default.conf file in that directory is essentially identical to the /etc/syslog.conf file from the older system and kernel log services described earlier.

Identifying a Baseline System State

A baseline is a default system configuration. In an ideal world, you could use the gold baseline described earlier as the starting point for production systems. To that end, any new system should differ from the baseline only with respect to the new packages that may be installed for functionality, updates, and security issues. That does not reflect reality, however.

Systems deviate from a baseline in a number of ways. Log files are added. Users add personal and work files to their home directories. Fortunately, such changes affect only a limited number of directories. You should be able to limit additional changes. Some users may install a new service. However, if you have configuration control of the systems on the local area network (LAN), no new services would be installed—at least not without your knowledge.

A baseline configuration includes more, however. It includes the current network state of the system, the interfaces that are used, the firewalls that are configured, the ports that may be open, and so on. Of course, one exception is based on key network settings—specifically the IP address and hostnames of each system.

Finally, a baseline configuration includes information on how the system runs. Information here is more subjective. Based on a given hardware configuration, however, a baseline system could be expected to run within certain parameters. The following sections will help you define a baseline system state in all of these areas.

Collect a List of Packages

Perhaps the simplest tool for a baseline system state is based on the governing package-management command. For example, the following command includes all packages currently installed on a Red Hat system:

```
# rpm -qa
```

Look at the output. It includes version numbers for each package. If saved to a file, you could use it in the future to identify changes in the list of installed packages over time.

A similar list of packages is available in Ubuntu systems, using the following command:

```
# dpkg -l
```



NOTE

Red Hat systems include a list of packages in the installation, in the /root/install.log file. Ubuntu does not have an equivalent file, although it's relatively easy to create one with a command like `dpkg -l > install.log`.

Although the output of the noted `dpkg` command includes a bit more information, that output can also serve as a database. If saved to a file, you could use that database to derive the differences in lists of installed packages at different dates.

Of course, if you have started with a gold baseline and installed a service such as the Samba Web server, you would expect the differences to include related packages. Any additional packages that are not the result of a dependency, an update, or a security issue are and should be suspect, however.

Compare Files, Permissions, and Ownership

One thing that integrity scanners do is check the status of files compared to a baseline. On any Linux system, some files are expected to change. Some change all the time. That is normal. Changes to key files such as the Linux kernel, modules, configuration files, and service scripts, however, may be a problem. Generally, you can divide files in these categories into stable and unstable directories. Even so, some changes to files in stable directories can be explained. Administrators may change configuration files. Security updates can change the kernel. The installation of new services can add files to stable directories.

Integrity scanners such as **Tripwire** compare the state of stable directories. If there are changes, Tripwire can help detect them. (If Tripwire is not already installed on a local system, you will see how to do so later in this chapter.) One measure of stable and unstable directories is the standard Tripwire policy file, twpol.txt, in the /etc/tripwire/ directory. This file includes several useful directives for different kinds of files and directories, as described in [Table 12-5](#). The categories and labels shown in [Table 12-5](#) provide one measure for files that should be tracked relative to a baseline configuration. The files and directories in each of these categories may vary depending on the type of system that is tracked.

TABLE 12-5 Tripwire configuration variables.

VARIABLE	DESCRIPTION
SEC_CRIT	These are critical files that cannot change, primarily in the /boot/, /lib/modules/, and /root/ directories.
SEC_BIN	These are files that generally should not change except when a system is updated. This applies mostly to binary files but includes configuration files in the /etc/ directory.
SEC_CONFIG	These are configuration files that are accessed frequently. Although such files change infrequently, access times are also recorded.
SEC_LOG	Log files may grow, but a change of ownership may indicate a malicious user who is substituting false information.
SEC_INVARIANT	These are directories that should always have the same user and group owner.
SIG_LOW	This is the label for files that are not critical, such as those in most user home directories.

SIG_MED	This is the label for files with some security impact, such as the user authentication database, configuration files, some binary files, and some library files.
SIG_HI	This is the label for files of high security importance, such as those in the /boot/, /bin/, /sbin/, and /lib/ directories.

Define the Baseline Network Configuration

One element that changes with each system is the baseline network configuration. By definition, different systems must have different IP addresses, hostnames, and network card hardware addresses. If you have configured a baseline system, there should be common elements in the network configuration such as the network device filename. One example is /dev/eth0. If all these systems are on the same network, they should generally have the same routing table.

Additional network configuration relates to open ports. There are three ways to measure open network ports, based on the following commands:

- **netstat -atun**—This command listens to all (-a) network sockets for connections associated with the TCP (-t) and UDP (-u) protocols, with output in numeric (-n) format, to avoid load on Domain Name Server (DNS) services.
- **nmap ip_address**—This command scans for open ports on the given IP address. Run this command using the same IP address on the local system and on a remote system. Any differences may reveal the effectiveness of a firewall.
- **iptables -L**—This command lists current firewall rules based on the **iptables** command.

This is not a complete view of the current network configuration, as individual services frequently have their own firewalls. It's a starting point, however.

Collect Runtime Information

One more element of a baseline configuration is how well it runs on a local system. A good baseline configuration will have some room to spare with regard to disk space, RAM, central processing unit (CPU) usage, and so on. A system that has been taken over by a malicious user may no longer have that room to spare. It may also have unexpected processes running, such as the **ping** command on a system being used in a denial of service (DoS) attack. Runtime information varies, however, so be flexible with the information you collect. A slight variance should not be important. Excessive use of swap space may be evidence of a memory leak as opposed to evidence of a Trojan horse.

You can collect runtime information with the following commands:

- **top**—This displays currently running Linux processes along with runtime, CPU load, RAM, and swap-space usage.
- **free**—This notes the current amount of free and used memory with respect to RAM and swap space.
- **vmstat**—This lists free RAM, memory allocated to buffers, memory swapped in and out, system interrupts, and CPU usage.
- **sar**—This collects custom system activity reports. It may be configured as a cron job.

Perhaps the most useful of these commands is **sar** because it enables you to collect system activity reports like log files. You can then review system activity that exceeds certain levels for further issues. It's available on both the Red Hat and Ubuntu distributions from the sysstat package.

Checking for Changes with Integrity Scanners

An integrity scanner is a system that can detect unwanted software on a computer. For the purposes of this section, it's essentially the same thing as an IDS. Software such as Tripwire and the **Advanced Intrusion Detection Environment (AIDE)** provide a number of options that can help you check the integrity of local Linux systems.

The right time to install an integrity scanner is just after you have set up a baseline configuration, before any such systems are made operational on a network. Unless a malicious user has penetrated the repositories for your selected Linux distribution, that integrity scanner should be able to establish baseline settings for your standard configuration.

The noted IDSs are just two of those available from the Ubuntu universe repositories. Although Tripwire has been included in older Red Hat releases, the only IDS available from Red Hat repositories is AIDE.

Tripwire

Perhaps the best known of the Linux IDS systems is Tripwire. Developed in 1992 by Gene Kim, Tripwire pioneered many of the techniques now associated with intrusion detection.

Similar to the sendmail and Sendmail e-mail services, there are open source and commercial versions of Tripwire. Although both were originally based on open source code, the development path of the open source and commercial versions of this software began to diverge in 1999. This section focuses on the open source version of Tripwire, documented at <http://sourceforge.net/projects/tripwire/>. If you need a commercial solution, there are several excellent Tripwire options described at <http://www.tripwire.com/>.

Because it's included in the Ubuntu universe repositories, Tripwire is somewhat easy to install on such systems. If you are installing Tripwire on a Red Hat system, you will first have to download the source code from the SourceForge Web site. You will also need the GNU C Language compiler along with C++ language support.

Tripwire requires a site and local passphrase, which supports extra levels of security. Once this is configured, you can run the following command on a regular basis to detect deviations from the baseline:

```
# tripwire --check
```

Figure 12-5 illustrates a small section of a Tripwire report, using the twpol.txt file policies in the /etc/tripwire/directory. The output is more alarming than is required, as the default twpol.txt settings set a high severity level to changes to files in the /proc/ directory. As any experienced Linux user should know, files in this directory are dynamic.

Rule Name	Severity Level	Added	Removed	Modified
Invariant Directories	66	0	0	0
* Tripwire Data Files	100	1	0	0
* Other binaries	66	3	0	1
Tripwire Binaries	100	0	0	0
* Other libraries	66	4	0	1
Root file-system executables	100	0	0	0
* System boot changes	100	1	0	1
Root file-system libraries (/lib)	100	0	0	0
Critical system boot files	100	0	0	0
* Other configuration files (/etc)	66	1	0	4
Boot Scripts	100	0	0	0
Security Control	66	0	0	0
Root config files	100	0	0	0
* Devices & Kernel information	100	522	348	0
Total objects scanned:	30575			
Total violations found:	887			
		56 , 1		3%

FIGURE 12-5

A Tripwire check.

They change frequently, with the load on a system. The report includes more than 1,000 lines of information, with data on changes detected to files on configured directories. You can analyze the file to review files that have been added or otherwise modified to see if there is a real problem.

Advanced Intrusion Detection Environment

One feature enabled with AIDE is flexibility with respect to the default configuration database. Depending on the distribution, you can configure AIDE in the aide.conf file in either the /etc/ or the /etc/aide/ directory. The Ubuntu release includes a cron job in the aide file in the /etc/cron.daily/ directory. To configure AIDE, open the aide.conf file.

As shown with the following database directives, the AIDE configuration database is normally set up in the /var/lib/aide/ directory. You can configure that on any location—even a remote networked directory. When properly configured, AIDE can send intrusion detection reports to a central server—perhaps a central logging server.

After you configure the aide.conf file, be sure to modify the aide cron job file in the /etc/aide/ directory to conform to the location for the baseline database.

To test it out, you will need to run the following command to configure that first database. The --config/etc/aide/aide.conf switch is not required on the Red Hat version of AIDE.

```
# aide --init --config /etc/aide/aide.conf
```

Once configured, the noted Ubuntu cron job should rerun the AIDE IDS on a daily basis. On Red Hat systems, you should be able to create a cron job based on the aide --check command. That command uses the files identified in the database and database_out variables. You will then be able to inspect those reports for possible intrusions.

For more information and the latest releases for AIDE, navigate to <http://aide.sourceforge.net>.

Best Practices: Building and Maintaining a Secure Baseline

Both Ubuntu and Red Hat Enterprise Linux will support the creation of a server baseline with a minimal installation. This will give you the smallest number of packages to start with and a good foundation on which you can then build other services, like a Web server, a firewall, or a log server.

Additional options include read-only or live-bootable versions of Linux. Several directories may be suited to configuration as a read-only filesystem, subject to the changes required from functional and security-related updates. Alternatively, specialized mount options such as `nosuid`, `nodev`, and `noexec` can minimize risks to files in key directories. A number of excellent Linux distributions are available in live CD format.

You can use the baseline installation to easily create other systems. It should be protected to avoid tampering. It should also include processes that will help to keep the baseline up to date with the latest security updates from the distribution vendor. You should also implement any configuration changes to the baseline in any systems that were created from the baseline. You will need processes and procedures to make that happen. This will enable you to have consistency in the base operating system across all of your server installations.

You can monitor the performance of a system through its logs. Local logs in Linux are in transition from the system and kernel logging services to the rsyslog service. The services in question share the same basic log facilities and message priorities. Although the standard log files for Ubuntu and Red Hat systems vary, they use the same basic log facilities and priorities. Both distributions give way when a service has its own log configuration in a subdirectory of `/var/log/`. The benefit of rsyslog is the ability to configure centralized log servers, with log files transmitted over secure connections.

A baseline system state starts with a list of installed packages. It continues with IDSs such as Tripwire, which can classify current files in different categories of importance. It includes information on the baseline network configuration and standard runtime values, based on commands like `sar`. Linux includes a number of options for IDSs, including Tripwire and AIDE. The time to install and set up an IDS is before a gold baseline is configured with a specific service and connected to a network. After a database of current files and directories with ownership and permissions is configured, you can run an IDS on a regular basis to make sure no significant changes have been made to that gold baseline.

FYI

You can use a number of products and services to help keep systems up to date and to deploy new systems. Red Hat offers Kickstart for deploying new systems, but there are also vendor-neutral options like Ansible. Ansible allows for easier automation of tasks. Learning scripting languages like Python, Perl, and Ruby can also help you to automate these tasks and to ensure that all your systems are kept as close to your specified standard as possible.



CHAPTER SUMMARY

A secure baseline is fundamental to a secure network. With a secure baseline, it's easier to configure servers with confidence. You can start with the basic Red Hat and Ubuntu server installation programs. When coupled with the right mount options, that baseline can be as good as gold.

You can measure the quality of the baseline with logs, configured locally or configured to send information to a central logging server. With that and related runtime information, you can identify a baseline system state. In addition, IDS tools such as Tripwire and AIDE can help you protect the integrity of that baseline system.



KEY CONCEPTS AND TERMS

alert

Advanced Intrusion Detection Environment (AIDE)

Anaconda
debug
emerg
err
info
notice
rsyslog
Tripwire
warn



CHAPTER 12 ASSESSMENT

1. Which of the following options supports remote updates from a Web-based interface?
 - A. Minimal installation
 - B. No automatic updates
 - C. Install security updates automatically
 - D. Manage system with Landscape

2. Which of the following package groups is included in a default RHEL 7 installation?
 - A. Automatic updates
 - B. KDE
 - C. GNOME
 - D. Secure Shell server

3. What is the mount option that disables executable binaries in an /etc/fstab configuration file?

4. Which of the following directories is normally *not* appropriate as a read-only filesystem?
 - A. /boot/
 - B. /home/
 - C. /root/
 - D. /sbin/

5. Which of the following directories is a standard location for packages downloaded from an Ubuntu repository?
 - A. /var/cache/apt/
 - B. /var/cache/yum/
 - C. /tmp/
 - D. /root/

6. Which of the following is *not* a reason to test updates before installing them on a gold baseline?
 - A. Potential effects on compiled software
 - B. Support issues with third-party software
 - C. Source code is unverified
 - D. Potential interactions with other software

7. Which of the following log priorities provides the most important messages?
 - A. **debug**
 - B. **err**
 - C. **info**
 - D. **notice**

8. In a Samba log file, which of the following is associated with the %m variable?

- A. Username
 - B. Hostname
 - C. Service version
 - D. User profile
- 9.** What option in the /etc/syslog.conf configuration file includes mail messages of only the `info` priority? Use the *facility.priority* format.
- 10.** Which of the following modules is associated with system logging in an rsyslog configuration file?
- A. imuxsock
 - B. imklog
 - C. imudp
 - D. imtcp
- 11.** Which of the following symbols in an rsyslog configuration file is associated with UDP connections?
- A. !
 - B. @
 - C. @@
 - D. #
- 12.** What is the simplest command that includes all packages on an Ubuntu system?
- 13.** Which of the following commands can best collect information on the activity on a system?
- A. `top`
 - B. `sar`
 - C. `vmstat`
 - D. `free`
- 14.** Which of the following configuration files includes Tripwire configuration policies in a human-readable format?
- A. `twcfg.txt`
 - B. `tw.cfg`
 - C. `twpol.txt`
 - D. `twpol.enc`
- 15.** What command switch inspects the current configuration, comparing it with a previously derived baseline configuration? This switch works with both the `tripwire` and `aide` commands.
- A. `--inspect`
 - B. `--check`
 - C. `--compare`
 - D. `--review`

CHAPTER

13 Testing and Reporting

ALTHOUGH YOU MAY TAKE every conceivable step to secure local systems and networks, that is not enough. Tests come in real time when attackers try to break through those security measures. Before an attacker can break in, you must test systems and networks for known security issues and measures.

Any test should apply to all parts of a layered defense, from an external firewall to the passwords selected by local users. You can monitor those ports that must be open to enable network communication. But if other ports are also open, attackers can use them to penetrate local systems without your knowledge. In this chapter, you will see what ports should be open and what you can do to keep unnecessary ports closed. The challenges have increased with the proliferation of virtual systems.

If such defenses prevent legitimate users from connecting, that would defeat the purpose of security. If an attacker does break in without your knowledge, however, that person may change files on local systems. Standard commands and integrity checkers can detect such changes. There are excellent open source and commercial tools that you can use to detect changes, potential security flaws, intrusions, and more.

Chapter 13 Topics

This chapter covers the following topics and concepts:

- Why you should test every component of a layered defense
- How to check for open network ports
- How to run integrity checks of installed files and executables
- How to ensure that security does not prevent legitimate access
- How to monitor virtualized hardware
- What open source security testing tools exist for Linux
- What vulnerability scanners exist for Linux
- Where to install security testing tools
- What best practices are for testing and reporting

Chapter 13 Goals

When you complete this chapter, you will be able to:

- Use open source testing tools to identify potential flaws in a security configuration
- Take advantage of tools on bootable read-only Linux distributions
- Manage security issues on physical and virtual systems
- Review systems for changes in key files

Testing Every Component of a Layered Defense

A typical network includes a number of different systems that provide detection, protection, logging, and maybe prevention. When combined, the resulting layered defense can be effective in preventing attacks. No security system is perfect, however. If there are weaknesses, you must know what they are. Although some security issues may be unavoidable in a functional system, consequent risks can be limited.

The layers of security on a network could include a series of firewalls. You can configure firewalls based on the `iptables` command on external routers, internal routers, and individual systems. You can also configure firewalls on many individual services.

If an attacker breaks through these defenses, there is still the matter of protecting user accounts. Even if a user account is compromised, mandatory access control systems can further protect internal systems.

This section provides an overview of the sections that follow so you can view the big picture of what is required to test every component of a layered defense.

Testing a Firewall

Implicit in these tests is the issue of open network ports. As noted in the opening comments of the `/etc/services` file, there are 65,536 ports available. Network communication normally proceeds through one of three protocols: the Transport Control Protocol (TCP), the User Datagram Protocol (UDP), or the Internet Control Message Protocol (ICMP).

FYI

nmap is short for Network Mapper and is the best-known port scanner. You use a port scanner to determine the port status on target systems. With a port scanner, you can determine whether a port is open, closed, or filtered. A system behind a firewall may respond differently to a connection request than one not behind a firewall.

Any communication, legitimate or otherwise, must proceed through these ports using one of the noted protocols. Commands that check for open network ports, such as `nmap` and `nc`, are discussed later in this chapter. Because firewall commands may be difficult to decipher, output like the following excerpt from the `nmap` command may be easier to read:

```
Interesting ports on 192.168.1.23: Not shown: 1679 closed ports
PORT STATE SERVICE
22/tcp open ssh
Nmap finished: 1 IP address (1 host up) scanned in 1.383 seconds
```

Just be sure to apply the command from both inside and outside the firewall. The results may differ depending on active services and ports addressed by firewall rules.

For testing defenses, after you have applied commands such as `nmap` and `nc`, temporarily turn off any firewall on a local system. One quick way to do so is with the `iptables -F` command. (You will learn how to reactivate firewalls at the end of the next section.)

Testing Various Services

It's possible for an administrator to make a mistake with a firewall. It's also possible for an attacker to use a command like the `nc` command to redirect traffic through an open port in a firewall. If you want systems to be able to communicate on the Internet, you cannot close all network ports on the firewall.

The easiest way to test the protection afforded various services is with a disabled `iptables` firewall. Only then can you test the effectiveness of internal protections on various services. You can then further protect such services with TCP Wrapper along with configuration rules unique to certain services. These additional configurations may include access control lists (ACLs) managed by the application.

As you read through the tests for various services, you will note the reliance on IP address-based restrictions. Yes, attackers can spoof IP addresses, including those normally found on private IP address networks. But that is one item that `iptables` rules such as the following can address—a benefit of a layered defense:

```
iptables -A RH-Firewall-1-INPUT -i eth0 -s 10.0.0.0/8
→ -j LOG --log-prefix "Dropped private class A address"
```

This command appends (-A) a rule to the chain named RH-Firewall-1-INPUT. The rule is applied to network packets coming in through the network interface (-i) on device eth0. The rule is applied to packets with a source (-s) address in the 10.0.0.0/8 network. If an incoming packet meets these conditions, the incidence of the packet is logged with the noted prefix in the log file. This just gives you an idea of the kinds of addresses being used by attackers, however. You should go one step farther with the following rule:

```
iptables -A RH-Firewall-1-INPUT -i eth0 -s 10.0.0.0/8 -j DROP
```

If you would rather have an error message sent to such users, substitute REJECT for DROP.

Testing TCP Wrapper Services

You can apply one general test to any services otherwise protected by TCP Wrapper in the /etc/hosts.allow and /etc/hosts.deny configuration files. For example, the following entry in the /etc/hosts.allow file limits access to the Secure Shell (SSH) service by username and IP address. This directive limits access to user michael on the noted IP address.

```
sshd : michael@10.22.33.44
```

You should be able to test this rule by trying to log in as user michael from a different IP address as well as by trying to log in as a different user from the specified IP address.

Testing Apache

Several services feature their own configuration checkers. For example, the Apache Web server includes a syntax checker in its basic control command. Depending on the distribution, the control command may be `httpd` or `apache2ctl`. The switches for these commands are the same. For example, the following command checks the syntax of Apache configuration files on an Ubuntu system:

```
# apache2ctl -t
```

In contrast, if you configure virtual hosts on an Apache Web server on a Red Hat system, the following command checks the syntax on each individually configured virtual host:

```
# httpd -S
```

Syntax checks are a basic part of the configuration of any service. In addition, a syntax check can help you identify services that may have been modified without your knowledge.

Of course, security checks on Apache go beyond syntax. If you have configured a secure Web site, you should check the status of secure certificates—especially if they were purchased from a certificate authority. For that purpose, you can look for an Online Certificate Status Protocol (OCSP) server.

Normally, if you have configured password security on an Apache-based Web site, it will be based on usernames and passwords configured with the `htpasswd` command. Unless such databases have been handed off to a Lightweight Directory Access Protocol (LDAP) server, an Apache authentication database is typically kept in or near the same directory as Apache configuration files.

Of course, you will want to test such password-protected directories. Can users without entries in the Apache authentication database access such directories? If you navigate directly to the protected directory in question, the Apache server should automatically prompt for a username and password.

One strong recommendation is to disable the use of .htaccess files. As hidden files, they do not appear in the output of a casual search. Therefore, if an attacker were to include an .htaccess file in an Apache configured directory, he or she could include directives that override any configuration you have set up. You can avoid such problems with the following directive in any **Directory** container:

```
AllowOverride none
```

Testing Samba

As with Apache, one issue with respect to Samba is the syntax. If there is a problem, Samba may not be available to your users, violating that tenet of the C-I-A triad of confidentiality, integrity, and availability. The **testparm** command provides an excellent syntax check for a Samba configuration file. The output, shown in [Figure 13-1](#), is straightforward.

If there is a problem with basic Samba security options, this output should reveal it. For example, the following option in the [homes] stanza reveals the existence of user home directories to any user who connects to that service:

```
browsable = yes
```

With a list of usernames, an attacker can get to work, especially if those users work primarily on Microsoft Windows machines.

```

Load smb config files from /etc/samba/smb.conf
Processing section "[homes]"
Processing section "[printers]"
Loaded services file OK.
Server role: ROLE_DOMAIN_MEMBER
Press enter to see a dump of your service definitions

[global]
    workgroup = MYGROUP
    server string = Samba Server Version %v
    security = ADS
    cups options = raw

[homes]
    comment = Home Directories
    read only = No
    browseable = No

[printers]
    comment = All Printers
    path = /var/spool/samba
    printable = Yes
    browseable = No
[root@RHELserver ~]#

```

FIGURE 13-1

Testing Samba syntax with `testparm`.

Some Samba directories may be further limited with configuration options such as `hosts allow`, which limits access to the local Samba server to systems on the localhost and the noted private IP address networks:

```
hosts allow = 127. 192.168.12. 192.168.13.
```

Of course, Samba security extends to connected user-authentication databases. If it's a Windows NT-style database stored locally, you can configure the Samba server as a Primary Domain Controller (PDC). Security options are associated with that database.

That database may be stored locally in a file in the `/etc/samba/` directory. Related options such as user and machine profiles are in essence Linux directories that hold Microsoft configuration files.

Testing Other Services

Apache and Samba are just two of the many Linux-based services available. The steps taken to test the security of those services should provide a model for testing the security of other network services that you have configured on a Linux system.

After Service Tests Are Complete

When you are finished testing various services, you should be able to restore the firewall with an appropriate script. Because the `iptables` command includes its own service on Red Hat systems, that command is straightforward:

```
# /etc/init.d/iptables restart
```

Of course, that applies only to Internet Protocol version 4 (IPv4) networking. For Internet Protocol version 6 (IPv6) networking, the corresponding command and script is `ip6tables`. Although the same scripts are not available on Ubuntu systems, tools such as the Uncomplicated Firewall front-end provide a script that can activate and deactivate iptables and ip6tables firewall rules.

Testing Passwords

As for passwords, a variety of open source testing tools are available. Of the top 10 password crackers at <http://sectools.org/crackers.html>, tools like **John the Ripper**, **Hydra**, and **RainbowCrack** can quickly decrypt and identify simple passwords from files like `/etc/shadow` and `/etc/samba/smbpasswd`.

In general, you should be able to take the files with local authentication databases and test them with the noted tools. Given enough time, processing power, and direct access to such files, all such passwords can be cracked. However, the process can take days, months, or even years, depending on the tool, the hardware, and the strength of the password. If an attacker did break into a local system and could get access to local authentication databases, how long would it take for that person to crack such passwords? That and related questions are explored later in this chapter.

Testing Mandatory Access Control Systems

If an attacker can break into a user or a service account, the next line of defense is a mandatory access control system, such as Security Enhanced Linux (SELinux) and Application Armor (AppArmor). If network services are properly protected, even a compromise in a service account would not lead to a compromise in a service.

For example, what happens if there is a compromise of the service account associated with the Apache Web server? Would an attacker, masquerading as a legitimate user, be able to write to key files? Generally, if SELinux were set to enforcing mode with an `httpd_sys_content_rw_t` context, the answer would be yes. But is that context necessary for the files in question? Could the service still function appropriately with a context of `httpd_sys_content_t`? These are the tests and questions associated with mandatory access control systems such as SELinux and AppArmor.

Checking for Open Network Ports

This section focuses on basic open source commands related to network ports—specifically the `nmap`, `lsof`, `telnet`, and `netstat` commands. Many of the tools discussed in this chapter take advantage of these commands and are in effect front-ends, which frequently include some graphical interface illustrating the output to these commands.

Related to these commands is netcat, the `nc` command, which is discussed later in this chapter. More extensive tools such as System Administrator's Integrated Network Tool (SAINT) and Nessus are also covered later in this chapter.

The `telnet` Command

One reason the `telnet` command still exists on the systems of many security administrators is its utility. It's an easy way to check for open ports over a network. For example, suppose a colleague has set up a Web site but nobody can connect to it. She says it's up and running on her system. It's easy to test a connection to such a Web site with the following command, substituting the actual domain name for `domain.com`:

```
$ telnet domain.com 80
```

If ports between your systems are open and unblocked, you should see the following output:

```
Trying some_ip_address...
Connected to domain.com (some_ip_address). Escape character is '^]'
```

At that point, you should be able to press **Ctrl+]** to get the **^]** character and press Enter to get to a **telnet>** prompt. From there, type the **quit** command to return to the command-line interface.

Similar commands should work with the port number for most standard services online. For example, if you run the **telnet your_mail_server.com 25** command (substituting the name of the target e-mail server for **your_mail_server.com**) and get a “Connection refused” message, a bad port number or perhaps a firewall rule may be blocking the connection to that outgoing e-mail server. If there is no response at all, then the **telnet** command is likely getting through to the target server but not getting a response. That suggests nothing is listening on port 25.

One of the limitations of the **telnet** command is that it can connect only to systems that use TCP ports. Later, you will look at the **nc** command, which can connect to systems that use TCP and UDP ports.

If some server is listening on port 25, you will see it in the output of a **netstat** command with appropriate switches. It can also work with systems that communicate on both TCP and UDP ports.

The **netstat** Command

The **netstat** command is a powerful tool. With a variety of different switches, it can help you get into and better understand the networking subsystem.

Knowledgeable Linux users should already know that the **netstat -r** command displays the routing table that applies to the current system. With the **-n** switch, the **netstat** command works in numeric format. In other words, it does not require access to Domain Name Service (DNS) or the **/etc/services** file.

The **netstat** command can tell you more. The **-t** switch specifies TCP ports. The **-u** switch specifies UDP ports. With the help of the **-a** switch, **netstat** can tell you about all connections on the local system. So put those switches together. [Figure 13-2](#) displays the start of the output of the **netstat -atun** command.

Active Internet connections (servers and established)				State	
Proto	Recv-Q	Send-Q	Local Address		
tcp	0	0	127.0.0.1:2208	0.0.0.0:*	LISTEN
tcp	0	0	0.0.0.0:902	0.0.0.0:*	LISTEN
tcp	0	0	0.0.0.0:742	0.0.0.0:*	LISTEN
tcp	0	0	0.0.0.0:139	0.0.0.0:*	LISTEN
tcp	0	0	127.0.0.1:5900	0.0.0.0:*	LISTEN
tcp	0	0	127.0.0.1:5901	0.0.0.0:*	LISTEN
tcp	0	0	127.0.0.1:5902	0.0.0.0:*	LISTEN
tcp	0	0	0.0.0.0:111	0.0.0.0:*	LISTEN
tcp	0	0	127.0.0.1:30003	0.0.0.0:*	LISTEN
tcp	0	0	192.168.122.1:53	0.0.0.0:*	LISTEN
:					

FIGURE 13-2

Identify what your system listens to with `netstat -atun`.

This is a lot of output, indicative of a system that is probably listening for too many connections. [Table 13-1](#) describes the output, with columns listed from left to right. To see the actual column headers, you will have to pipe the output to the `less` command pager, and then the views of the output would be limited. Alternatively, if a bastion host is configured on the local system, a view of all such listen connections can be seen in a normal output, and you will see an example of such later in this section.

With [Table 13-1](#) in mind, take another look at the output from the `netstat -atun` command shown in [Figure 13-2](#). Review the first couple of lines of output, one at a time:

```
tcp 0 0 127.0.0.1:2208 0.0.0.0:.* LISTEN
```



TIP

When you see the IPv4 address 0.0.0.0, it is commonly meant to denote all possible addresses. In the case of `netstat` output, it's meant to indicate that the service is listening on all possible interfaces.

The first line is a TCP connection associated with port 2208, listening for requests from systems with any IP address (0.0.0.0). An Internet search might send you into a panic, as there are entries that suggest that port is a common backdoor for attackers. But it's also used by the Hewlett-Packard Linux Imaging and Printing Project and may be necessary for communication with such printers. In addition, because the local address is set to the loopback address of 127.0.0.1, the system might be listening just to the local loopback adapter.

To confirm, you can run the variation on the `telnet` command described earlier, as it would apply to port 2208:

```
$ telnet localhost 2208
```

TABLE 13-1 Output from `netstat -atun`.

COLUMN	DESCRIPTION
Proto	The protocol, typically TCP or UDP. On some systems, protocols associated with IPv6 communication may be shown as TCP6 and UDP6.
Recv-Q	Received communication in the queue, in bytes, usually 0.
Send-Q	Transmitted communication in the queue that is not yet acknowledged, in bytes, usually 0.
Local Address	Local IP address and port number.
Foreign Address	Remote IP address and port number. Unless there is an established connection, this is normally set to IPv4 address and port number <code>0.0.0.0:*</code> , which is listening for connections from any IPv4 address. It may also be set to IPv6 address and port number <code>:::*</code> .
State	Connection state, normally LISTEN or ESTABLISHED. A network service that is in the LISTEN state has not yet made a connection.

Because the localhost address is normally associated with 127.0.0.1 in the /etc/hosts file, that connection should work. Try it again with the IP address of the local adapter.

If the `netstat` status is `true`, the connection should be refused. The same command, when run from a remote system, may lead to a message that suggests that the connection was refused, typical of a block by an iptables-based firewall.

Now take the second line of output, which listens for TCP traffic from any address on port 902:

```
tcp 0 0 0.0.0.0:902 0.0.0.0:* LISTEN
```

In this case, the /etc/services file is a bit misleading, because port 902 is the standard for VMware systems that are controlled over a network. That is confirmed by the output to the `telnet remote_ip_address 902` command, as shown here:

```
220 VMware Authentication Daemon Version 1.10:  
→ SSL Required, MKSDisplayProtocol:VNC
```

In this case, the port is open, even when the associated VMware service is stopped.

The lines in the screen shown in [Figure 13-2](#) go on for some time. That quantity of output suggests that the associated system may not be very secure. In contrast, take a look at [Figure 13-3](#), where the `netstat -atun` command is applied to a more secure system.

The displayed output is easier to explain. The first line, which specifies port 22, is associated with an SSH server. The second line, associated with the localhost address and port 25, illustrates a Simple Mail Transfer Protocol (SMTP) server such as Postfix or sendmail that is not yet listening to anything from a regular network address. Take a closer look at the third line, associated with an ESTABLISHED connection:

```
tcp 0 0 192.168.122.19:22 192.168.122.1:56112 ESTABLISHED
```

It's a TCP connection on port 22, from the noted 192.168.122.1 IP address. Communication is received back on that client on port 56112.

The next two lines are associated with TCP networking over an IPv6 connection, as indicated by the `tcp6` label. They are the IPv6 settings that correspond to the first two lines.

The final line specifies an open UDP port 68, normally associated with the **Bootstrap Protocol (BOOTP)**. (BOOTP is a protocol of the Transmission Control Protocol/Internet Protocol [TCP/IP] suite associated with the automatic assignment of IP addresses.

```
michael@ubuntulucidsrv2:~$ netstat -atun
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address          Foreign Address        State
tcp      0      0 0.0.0.0:22              0.0.0.0:*
tcp      0      0 127.0.0.1:25             0.0.0.0:*
tcp      0      0 192.168.122.19:22       192.168.122.1:56112 ESTABLISHED
tcp6     0      0 :::22                  ::::*
tcp6     0      0 ::1:25                 ::::*
udp      0      0 0.0.0.0:68              0.0.0.0:*
michael@ubuntulucidsrv2:~$
```

FIGURE 13-3

The **netstat -atun** command on a bastion server.

It may also be used for the automatic acquisition of IP addresses from a Dynamic Host Configuration Protocol [DHCP] server on a remote network. It's associated with UDP port 68.) If this system can get IP address information from the local system or does not need to use DHCP, you may want to disable or even uninstall associated software.

Now take the command a step farther with the **-p** switch to identify the process in question. For the UDP BOOTP port just described, that specifies the dhclient process, which confirms the association between that port and the DHCP client.

The **lsof** Command

The **lsof** command lists open files. It can also be focused on network-related processes with the **-i** switch. Add the **-n** switch to keep IP addresses in numeric format, and you can quickly get a view of network-related processes on the local system:

```
# lsof -ni
```

If you have a lot of active network services, the output could be substantial. [Figure 13-4](#) shows one example. Compare this with the output to the **netstat -atun** command described earlier. Not only does the **lsof -ni** command organize connections by service, it also includes user and process identifier (PID) information.

COMMAND	PID	USER	FD	TYPE	DEVICE	SIZE	NODE	NAME
ssh	3037	michael	3u	IPv4	11425157		TCP 10.168.0.8:36277->10.168.0.60:ssh (ESTABLISHED)	
racoon	5144	root	6u	IPv4	17509		UDP 127.0.0.1:isakmp	
racoon	5144	root	8u	IPv4	17510		UDP 10.168.0.8:isakmp	
racoon	5144	root	11u	IPv6	17513		UDP [::1]:isakmp	
racoon	5144	root	12u	IPv6	17514		UDP [fe80::2ff:17ff:fe6b:c680]:isakmp	
portmap	5163	daemon	3u	IPv4	13293		UDP *:sunrpc	
portmap	5163	daemon	4u	IPv4	13294		TCP *:sunrpc (LISTEN)	
rpc.statd	5182	statd	5u	IPv4	13324		UDP *:694	
rpc.statd	5182	statd	7u	IPv4	13332		UDP *:56646	
rpc.statd	5182	statd	8u	IPv4	13335		TCP *:42673 (LISTEN)	
avahi-dae	5896	avahi	14u	IPv4	14154		UDP *:mdns	
avahi-dae	5896	avahi	15u	IPv4	14155		UDP *:41511	
mysqld	6007	mysql	10u	IPv4	14265		TCP 127.0.0.1:mysql (LISTEN)	
cupsd	6204	root	2u	IPv6	14532		TCP *:ipp (LISTEN)	
cupsd	6204	root	3u	IPv4	14533		TCP *:ipp (LISTEN)	
cupsd	6204	root	5u	IPv4	14536		UDP *:ipp	
sshd	6234	root	3u	IPv6	10064987		TCP *:ssh (LISTEN)	
hddtemp	6254	root	0u	IPv4	14572		TCP 127.0.0.1:7634 (LISTEN)	
master	6440	root	11u	IPv4	14952		TCP *:smtp (LISTEN)	
nmbd	6472	root	6u	IPv4	15109		UDP *:netbios-ns	
nmbd	6472	root	7u	IPv4	15110		UDP *:netbios-dgm	
nmbd	6472	root	8u	IPv4	15112		UDP 10.168.0.8:netbios-ns	
nmbd	6472	root	9u	IPv4	15113		UDP 10.168.0.8:netbios-dgm	
nmbd	6472	root	15u	IPv4	61862		UDP 172.16.105.1:netbios-ns	
nmbd	6472	root	16u	IPv4	61863		UDP 172.16.105.1:netbios-dgm	
smbd	6474	root	19u	IPv4	15139		TCP *:microsoft-ds (LISTEN)	
smbd	6474	root	20u	IPv4	15140		TCP *:netbios-ssn (LISTEN)	
vsftpd	6598	root	3u	IPv4	15230		TCP *:ftp (LISTEN)	
xinetd	6655	root	5u	IPv4	15340		TCP *:vmware-authd (LISTEN)	
xinetd	6655	root	6u	IPv4	15341		TCP *:swat (LISTEN)	
xinetd	6655	root	8u	IPv4	15342		TCP *:kshell (LISTEN)	
xinetd	6655	root	9u	IPv4	15343		TCP *:eklogin (LISTEN)	
xinetd	6655	root	10u	IPv4	15344		TCP *:telnet (LISTEN)	
asterisk	6737	asterisk	7u	IPv4	15443		TCP 127.0.0.1:5038 (LISTEN)	

FIGURE 13-4

The `lsof -ni` command on a multipurpose system.

The nmap Command

The `nmap` command is an incredible tool. Several books have been written about this network exploration tool and port scanner. To get a taste of what it can do, run `nmap` by itself. Then run it on the system of your choice using an IP address or hostname. If a firewall has been configured on a system, it's interesting to note the differences from inside and outside the firewall.

The following sections address some of the options described in the `nmap` output. Those commands that require root administrative privileges are preceded by the standard root prompt, #.

`nmap` can take several minutes to run over a network. If you are just learning `nmap`, it may be faster to run the commands described in the following subsections against an IP address on your local network. Of course, if you are testing a system behind a firewall, you cannot measure the effectiveness of that firewall unless you run `nmap` on a system on the other side of that firewall.

WARNING

As noted by the developers of `nmap` at <http://nmap.org/book/legal-issues.html>, improper use of the command "can (in rare cases) get you sued, fired, expelled, jailed, or banned by your ISP." So don't use `nmap` against a system or network unless you are specifically allowed to do so.

In general, if you want more information in the output, the verbose (-v) switch can help. If you need still more information, the -vv switch adds even more verbosity. The description in this chapter is far from complete; there are several books written solely on the intricacies of `nmap`.

Target Specification

You can target **nmap** at a group of systems, an IP address, a hostname, or a fully qualified domain name. For example, the following command reviews the systems with IP addresses 192.168.0.1 through 192.168.0.3:

```
$ nmap 192.168.0.1-3
```

The following command reviews all systems on the noted network, in Classless Inter-Domain Routing (CIDR) notation:

```
$ nmap 192.168.0.0/24
```

Host Discovery

An attacker with access to a private network might start with a ping scan. In the following command, the **-sP** switch scans and the **-P** switch applies the **ping** command to the systems on the target addresses:

```
$ nmap -sP 10.0.0.0/8
```

It just takes a few seconds to display a list of active systems on that network. One other useful switch from this section is the **-n** switch, which avoids the use of DNS servers for name resolution. If there is an excessive delay when running other **nmap** commands, the **-n** switch may help.

Although such a command saves time, not all hosts respond to requests made with the **ping** command. The following command scans all hosts in the noted network:

```
$ nmap -PN 192.168.0.0/24
```

The command is time consuming. It could easily take several hours to scan all the ports on all the IP addresses on the noted network. That is just for the 254 allowable IP addresses on that network. That process would take a lot longer for the 16,000,000 IP addresses on the class A private network, 10.0.0.0/8.

Scan Techniques

Although **nmap** focuses on TCP packets, network communication also uses UDP packets. To scan a system for open UDP ports on a given IP address, run the following command:

```
# nmap -sU ip_address
```

Of course, network communication can proceed through more than just the TCP and UDP protocols. To find the full list of protocols enabled for an IP address, run the following command:

```
# nmap -sO ip_address
```

Port Specification and Scan Order

One of the idiosyncrasies of **nmap** is that it scans only most commonly used ports. The number of ports checked varies depending on the version of **nmap** used. For example, version 5.00 checks 1,000 ports by default. For some, however, that is an incomplete scan. The **-p** switch can help; the following scan checks all 65,536 TCP/IP ports:

```
$ nmap -p 0-65535 ip_address
```

Of course, because that scan just applies to TCP ports, you may also be interested in the following command, which builds on the switches described in a previous section to check UDP ports:

```
$ nmap -sU -p 0-65535 ip_address
```

Service Version Detection

With the `-sV` switch, `nmap` searches for the version number of all services communicating over open ports.

[Figure 13-5](#) shows the output of the `nmap -sV -p 0-65535 10.168.0.1` command. Note how it lists all open services and their version numbers. An attacker who looks at this output and is aware of weaknesses of older versions of certain servers may pounce on this information.

Of course, as a security professional, you can use this information to make sure the given services are up to date, with the latest security updates. This information may also reveal ports that you may close, perhaps by uninstalling or disabling a service. Alternatively, you could close these ports with appropriate firewall commands.

PORT	STATE	SERVICE	VERSION
21/tcp	open	ftp	vsftpd 2.0.6
22/tcp	open	ssh	OpenSSH 4.7p1 Debian 8ubuntu1.2 (protocol 2.0)
23/tcp	open	tcpwrapped	
25/tcp	open	smtp	Postfix smtpd
80/tcp	open	http	Apache httpd
111/tcp	open	rpcbind	2 (rpc #100000)
139/tcp	open	netbios-ssn	Samba smbd 3.X (workgroup: WORKGROUP)
443/tcp	open	http	Apache httpd
445/tcp	open	netbios-ssn	Samba smbd 3.X (workgroup: WORKGROUP)
544/tcp	open	kshell	Solaris kerberised rsh
631/tcp	open	ipp	CUPS 1.2
877/tcp	open	ypbind	1-2 (rpc #100007)
901/tcp	open	http	Samba SWAT administration server
902/tcp	open	ssl/vmware-auth	VMware GSX Authentication Daemon 1.10 (Uses VNC)
1021/tcp	open	rquotad	1-2 (rpc #100011)
2000/tcp	open	callbook?	
2105/tcp	open	klogin	Kerberized rlogin
3306/tcp	open	mysql	MySQL 5.0.51a-3ubuntu5.5
5038/tcp	open	asterisk	Asterisk Call Manager 1.0
7634/tcp	open	hddtemp	hddtemp hard drive info server
8834/tcp	open	unknown	
37737/tcp	open	unknown	
57817/tcp	open	status	1 (rpc #100024)

FIGURE 13-5

A list of open services with version information.

Operating System Detection

The operating system detection option, `-O`, may reveal the operating system associated with the target machine. As of this writing, it's not very precise. For example, it provides this output for the operating system along with a list of open TCP ports. Fortunately or otherwise, the noted operating system includes a Linux kernel version 2.6.24.

```
Device type: general purpose Running: Linux 3.X
OS details: Linux 3.7 - 3.15
```

The scripts as listed fall into several categories. Some scripts may fall under multiple categories. Because new scripts and even new switches are under development, assume this is a partial list. There may be a dozen or more scripts available in several of these areas.

Scripts can be bundled together. For example, a command like the following applies all scripts associated with a scan of the Hypertext Transfer Protocol (HTTP) service:

```
# nmap --script=http-* 192.168.100.1
```

If the given script is misspelled or is not otherwise available, `nmap` should return an error with the following message related to the script:

```
NSE: failed to initialize the script engine:  
'filename' did not match a category, filename, or directory
```

Some of the scripts described appear only in beta versions of `nmap`. Although the use of beta software is normally discouraged in production software, it can give you more tools to detect security issues as they are created by attackers.

Authentication. `nmap` authentication scripts test usernames, passwords, and more on target systems, possibly through a brute-force attack. You can use authentication scripts for networked services associated with a number of protocols. These protocols range from the Apple Filing Protocol (AFP) to Telnet. For example, the following command attempts a brute-force attack to guess usernames and passwords on the FTP server on the noted IP address:

```
# nmap --script=ftp-brute 192.168.100.1
```

Default. The default scripts associated with `nmap` find default information for a wide variety of services, from DHCP to DNS to shared directories on Microsoft networks. For example, the following command checks the target system for the Conficker.C worm:

```
# nmap --script=p2p-conficker 192.168.100.1
```

Discovery. Discovery scripts for `nmap` can find information associated with a variety of network services, from banners to shared Samba and NFS directories. Discovery scripts include WHOIS information, Secure Sockets Layer (SSL) certificates, and more. For example, the following command checks the target system for a list of directories shared over the Samba file service:

```
# nmap --script=smb-enum-shares 192.168.100.1
```

`nmap` provides more than just share information. It also includes the version number of the Samba release, the path to the shared directory, anonymous access if allowed, connected users, and more.

Denial of service and exploit. As of this writing, there is one script for both the denial of service (DoS) and exploit categories. The following script reviews a target Samba server for a number of vulnerabilities beyond DoS:

```
# nmap --script=smb-check-vulns 192.168.100.1
```

External. The scripts in the external category are associated with the way servers look to external systems. Many of the scripts in this category can expose risks. For example, the following command determines whether an e-mail server on a remote system is an open relay that can be used by spammers to send messages through your systems:

```
# nmap --script=smtp-open-relay 192.168.100.1
```

The following output is associated with a SMTP server on port 25:

```
25/tcp open smtp  
|_smtp-open-relay: Server is an open relay (16/16 tests)
```

Fuzzer. Fuzz attacks on various servers test responses to problems like buffer overflows. So far, `nmap` includes only one script in this area, on DNS. For example, the following command uses a script to launch a DNS fuzzing attack against an associated server:

```
# nmap --script=dns-fuzz 192.168.100.1
```

Intrusive. There are many **nmap** command scripts that qualify in both the intrusive and authentication categories. Any attempt to guess usernames and passwords by brute force is an attempt to intrude and a test of authentication. The following command goes farther to see whether the network card associated with the IP address is running in promiscuous mode. If so, an attacker will be able to use a protocol analyzer on the local network:

```
# nmap --script=sniffer-detect 192.168.100.1
```

If the network interface is running in promiscuous mode, which would be the case if you were running a packet-capture program, you may see the following output:

```
Host script results:
|_sniffer-detect: Likely in promiscuous mode
```

Malware. Perhaps this should be a growing category of scripts. For the moment, three have been written for this topic: one each for authentication, Web servers, and SMTP servers. The following command checks a Web server for known compromises:

```
# nmap --script=http-malware-host 192.168.100.1
```

Safe. Scripts in the safe category test the relative safety of named services. As befits this category, it includes several dozen scripts testing the safety of services from the Internet Mail Access Protocol (IMAP) to the LDAP. While a script that tests the capabilities of a Post Office Protocol version 3 (POP3) server such as Dovecot may not seem to fit the bill, it can reveal features that are unneeded or unused. Such features are frequently ripe for the picking by attackers. The noted script is shown here:

```
# nmap --script=pop3-capabilities 192.168.100.1
```

The following output excerpt suggests a lot of POP3 features that should be carefully reviewed:

```
110/tcp open pop3
|_pop3-capabilities: USER CAPA UIDL TOP OK(K)
  ↘ RESP-CODES PIPELINING STLS SASL(PLAIN)
```

Version. The scripts in this category detect more than just the version number of specific services. Not many scripts exist in this category because the **nmap -sV ip_address** command does an excellent job detecting the version number and service associated with open ports. However, the following command does provide more information about existing Point-to-Point Tunneling Protocol (PPTP) services. In Linux, this is also known as Internet Protocol Security (IPSec).

```
# nmap --script=pptp-version 192.168.100.1
```

Vulnerability. The scripts in the vulnerability category are all about testing for known vulnerabilities. Expect the number of scripts in this category to grow. The following example tries a known SQL injection on a target HTTP server:

```
# nmap --script=mysql-injection 192.168.100.1
```

The scripts as listed fall into several categories. Some scripts may fall under multiple categories. Because new scripts and even new switches are under development, assume this is a partial list. There may be a dozen or more scripts available in several of these areas. Scripts can be bundled together. For example, a command like the following applies all scripts associated with a scan of the HTTP service:

```
# nmap --script=http-* 192.168.100.1
```

If the given script is misspelled or is not otherwise available, **nmap** should return an error with the following message related to the script:

```
did not match a category, filename, or directory
```

Running Integrity Checks of Installed Files and Executables

When you have a production system, it's important to monitor that system for changes. Of course, you should not be alarmed every time a new log is added to the /var/log/ directory. However, changes to critical files such as those in the /boot/ and /sbin/ directories may indicate a problem.

To detect such changes, you need to know how to check the integrity of target systems. Sure, you could just set up a checksum on potentially vulnerable partitions or volumes, but that would be imprecise. If there is a problem, the checksum will not specify the files that have changed.

The first method to check the integrity of a system relates to the integrity of installed packages. To that end, commands based on package-management systems such as the Red Hat Package Manager (**rpm**) and the Debian package-management program (**dpkg**) can help.

The default configuration based on a package installation is not precise, however. It highlights problems with normal changes to standard configuration files. Sure, you could keep a list of those changes, but if an attacker were to get into a system and change those files further, there would be no way to tell just using package-management commands. This is the reason behind integrity checkers such as Tripwire and the Advanced Intrusion Detection Environment (AIDE). In general, you will want to use such integrity checkers to log the configuration associated with a gold baseline before putting such systems online. Then, you can run the integrity checker on a regular basis. When properly configured, they can highlight any unintended changes to the baseline configuration.

Verifying a Package

The first method for verifying the integrity of a system is relative to its installed state. With the correct command, you can compare the current list of installed packages with an associated list of installed files. The basic commands vary depending on whether the distribution is built on packages associated with Red Hat or Debian systems.

Note that the use of package-management commands may not provide a complete measure of what is installed. Many enterprising users and administrators install packages more directly from source code, typically collected in a compressed tarball archive. (A *tarball* is what you call a file that has files or directories collected into a single file by the tar program.)

Verifying Ubuntu/Debian Packages

Because Ubuntu is built on Debian Linux, Ubuntu packages are built on the Debian package system. As such, it relies on commands like **dpkg** and **debsums**. To obtain a list of currently installed packages on these systems, run the **dpkg -l** command. Prepare to be overwhelmed as the list of hundreds or even thousands of packages whizzes by. Of course, the value of such a list is as a database, which you can pipe to a text file or search with the **grep** command.

One variation on **dpkg -l** that provides a more straightforward list of installed packages is as follows:

```
$ dpkg --get-selections
```

That is just a list of installed packages. When such packages are installed, they include a list of checksums, classified by package, separated by each installed file in the /var/lib/dpkg/info/directory. To take advantage of that list, run the **debsums** command. The **debsums** command uses Message Digest 5 (MD5) checksums to see if changes have been made to files. The output from the command is impressive. Some desktop systems include a couple of hundred thousand packages or more.

Not all packages include checksum information, however. In addition, some files are configured from a combination of packages, such as the initial RAM disk file in the /boot/directory. For those files associated with checksums, the only indication of a problem is a **FAILED** message after the filename.

Verifying Red Hat Packages

The Red Hat Package Management system is more precise when it checks the integrity of installed packages. It's a straightforward command, given the name of a target package. For example, the following command and output is typical for the Postfix e-mail server:

```
$ rpm -V postfix
..5....T c /etc/postfix/main.cf
```

TABLE 13-2 Failure codes for `rpm -V`.

CODE	DESCRIPTION
5	Checksum associated with the MD5 algorithm
S	File size
L	Symbolic link
T	File modification time
D	Device file
U	User owner
G	Group owner
M	File mode

If there were no changes to the files relative to the original package, there would be no output. Eight tests are performed on each file, as described in [Table 13-2](#). The codes would appear from left to right. A dot in place of the code means that the associated test has passed.

If a C appears at the end of the failure-code list, the noted file is a configuration file. The output from the `rpm -V packagename` command can tell you more about the change to the file.

Of course, with hundreds and sometimes thousands of packages on a system, it's not enough to verify the files associated with one package. Fortunately, the `-a` switch helps the `rpm -ga` command check all installed packages. Although changed files are not directly associated with the package, it's easy to identify the package that owns a file. For example, the following command identifies the package that owns the noted Samba configuration file:

```
$ rpm -qf /etc/samba/smb.conf
```

As with the `debsums` command, such checks apply only to packages installed in the rpm package-manager format.

Performing a Tripwire Check

Tripwire is an intrusion detection system (IDS), with configuration files in the /etc/tripwire/ directory. Briefly, Tripwire can collect information on key files on a computer based on how Tripwire is configured in the twcfg.txt file and the reporting policies documented in the twpol.txt file. When the configuration and policy are set, the following command creates a database of digital checksums for such files in the /var/lib/tripwire/ directory:

```
# tripwire --init
```

Next, you should run the following command to review associated policies. Standard policies for the Tripwire package that you have installed may check on files that do not exist on local systems. Therefore, you should run the following command right away, to modify configuration options and policies to minimize the number of false positives:

```
# tripwire --check
```

For example, the standard Tripwire configuration reports changes in files in the /proc/ and /dev/ directories. Current Linux systems change the contents or access time of numerous files in these directories every time a system is rebooted. Current Linux systems may also change files in the /etc/ directory if the system acquires network address information using DHCP. You can update Tripwire policies with the following command:

```
# tripwire --update
```

Then the standard Tripwire package is configured to check the local system on a daily basis, courtesy of the `tripwire` script in the /etc/cron.daily/ directory.

Testing with the Advanced Intrusion Detection Environment

Like Tripwire, AIDE supports periodic checks of installed and configured files. This section goes into a bit more depth on how AIDE works based on information in the aide.conf configuration file. The configuration for Ubuntu and Red Hat systems can vary somewhat. Of course, the details are subject to change, especially if the developers behind one distribution decide that the approach of the other distribution is better. In either case, AIDE rules are defined with attributes as described in [Table 13-3](#).

Some developers consider AIDE to be superior to Tripwire. In fact, AIDE packages are available from the main Ubuntu repository and supported Red Hat Enterprise Linux (RHEL) repositories. Therefore, the installation of AIDE is supported at least in some ways by developers for both distributions.

Ubuntu AIDE

When installed on Ubuntu systems, AIDE is configured in files in the /etc/aide/ directory. In addition, per the /etc/default/aide file, configuration options for individual services are defined in files in the /etc/aide/aide.conf.d/ directory. Although the basic database files are similar, Ubuntu's version includes directives with combinations of default codes in the main /etc/aide/aide.conf file. For example, the following directive suggests that AIDE can use a variety of checksums for every file. Each of the options listed is described in [Table 13-3](#).

```
Checksums = md5+sha1+rmd160+haval+gost+crc32+tiger+whirlpool
```

TABLE 13-3 AIDE rule codes.

CODE	DESCRIPTION
p	Permissions: read, write, execute
i	Inode: labeled location on a volume or partition
n	Number of links
u	User
g	Group
s	Size

b	Block count
m	Modification time: the last time ownership or permissions were changed
a	Access time: the last time the file was read or otherwise accessed
c	Change time: the last time the file was changed
S	Label for files expected to change in size, such as log files
acl	Access control list (ACL) labels
selinux	Security Enhanced Linux (SELinux) context
xattr	Extended file attributes
md5	MD5 checksum
sha1	Secure Hash Algorithm (SHA) version 1 hash function; may also refer to sha256 for a 256-bit hash function
rmd160	A hash function based on a Java checksum algorithm based on a Message Digest 4 (MD4) hash; the rmd acronym is a shortened form that refers to the RACE Integrity Primitives Evaluation Message Digest
haval	A cryptographic hash function with variable length output
gost	A cryptographic hash function originally developed in the former Soviet Union
crc32	A cryptographic hash function based on a cyclic redundancy check (CRC)
whirlpool	Another cryptographic hash function

The **OwnerMode** directive that follows calculates the permissions, user owner, and group owner for all files so configured:

```
OwnerMode = p+u+g
```

The **Size** directive is applied to files where a constant size can help determine whether there has been a change with respect to the size and block count of a file:

```
Size = s+b
```

Additional directives in the aide.conf file describe other directives that are applied in files in the /etc/aide/aide.conf.d/ directory.

Red Hat AIDE

The Red Hat AIDE configuration is somewhat more self-contained in the /etc/ aide.conf file. The directives defined are somewhat different and are in some ways cumulative. For example, the R directive shown here defines a group of codes, as defined in [Table 13-3](#).

```
R: p+i+n+u+g+s+m+c+acl+selinux+xatrs+md5
```

This group is inserted in the value of the **NORMAL** directive shown here, which incorporates a couple of checksum hash functions.

```
NORMAL = R+rmd160+sha256
```

The directive that follows essentially creates a DIR directive equivalent to the R group of codes:

```
DIR = p+i+n+u+g+acl+selinux+xattrs
```

You should check some files for access controls with the PERMS directive defined here. This directive essentially is a check of regular discretionary access permissions, ownership, ACLs, and SELinux contexts:

```
PERMS = p+i+u+g+acl+selinux
```

The excerpt of /etc/aide.conf shown in [Figure 13-6](#) illustrates default checks on key directories. Several files have the ! character in front. This is described as the *bang*, which means “anything but.” In other words, in this default configuration, AIDE does not check files in the /usr/src and /usr/tmp/ directories or certain files in the /etc/ directory.

```
# Next decide what directories/files you want in the database.
```

```
/boot    NORMAL
/bin     NORMAL
/sbin    NORMAL
/lib     NORMAL
/lib64   NORMAL
/opt     NORMAL
/usr     NORMAL
/root    NORMAL
# These are too volatile
!/usr/src
!/usr/tmp

# Check only permissions, inode, user and group for /etc, but
# cover some important files closely.
/etc     PERMS
!/etc/mtab
# Ignore backup files
!/etc/.*~
/etc/exports  NORMAL
/etc/fstab    NORMAL
/etc/passwd   NORMAL
```

FIGURE 13-6

Excerpt from the default Red Hat version of aide.conf.

Ensuring that Security Does Not Prevent Legitimate Access

It's possible to configure too much security on a system. In the area of passwords, if the requirements are too strict, users are more likely to forget their passwords. Users will then either overload administrators with password-reset requests or write their passwords next to their workstations. Neither situation promotes true computer security. In the following sections, you will look at reasonable password policies, how you can

implement them in the shadow password suite with the `chage` command, and how you can regulate access with pluggable authentication modules (PAMs).

Reasonable Password Policies

In an ideal world, users would frequently change their passwords. After all, if the lifetime of a password were less than the time it takes for an attacker to decipher the password, then you could keep password-authentication databases out in the open! It's not, however, an ideal world.



NOTE

Of course, you never want to publicize a password-authentication database. Attackers can get access to some powerful systems. Some companies sell rainbow tables, which can speed the process of decrypting a password.

In addition to needing to change passwords to prevent them from being cracked, you should change them so that if a password *is* cracked, the attacker has access to the account for only a limited time. This is less relevant if the account has the ability to create users or access can be escalated to create users. This is because an attacker may simply create his or her own account rather than continuing to use a compromised account. Of course, a new account can be noticed, so it's not a guarantee that a new account will be created.



Considering the long-known weakness of passwords, many organizations are moving to multi-factor authentication. Additional technologies like biometrics have become much cheaper and more reliable, making multi-factor authentication a security mechanism that is within reach of many more organizations than in the past.

The best passwords are passphrases, which can be complete sentences. Many authentication systems do not support options such as spaces in a password, however. Other than a passphrase, the best password is a combination of uppercase and lowercase characters, along with numbers and punctuation. You want users to be able to remember their passwords without having to write them down, however. One U.S. governmental guideline is based on the Federal Desktop Core Configuration (FDCC), as mandated by the U.S. Office of Management and Budget (OMB). While the password guidelines apply only to Microsoft Windows systems, the listed policy suggests that passwords:



TIP

One way to help users remember such passwords is based on a sentence. For example, a password such as "OT,ls3ba1e!" could stand for "On Tuesday, I shot 3 birdies and 1 eagle!"

- Have a minimum of 12 characters
- Be changed every 60 days

Although these policies do not apply to users on Linux systems, these are reasonable minimum requirements. On the other hand, in organizations where security is important, such as intelligence agencies, users generally have a greater understanding of the need for password security. You should be able to set more rigorous limits in such cases. The challenge is to have a balanced password policy. If you make the policy too restrictive with regular changes and complexity requirements, people may end up writing their passwords down. If users are writing their passwords down just to keep up with draconian policies, the password policy will not be protecting systems

or the information stored on them. It's important to keep the goal of a policy in mind. The goal should be to protect unauthorized access to your systems.

You can configure account policies for a specific account with the `chage` command. The current settings are defined in the output to the `chage -l username` command. [Table 13-4](#) shows the output and associated `chage` command options. You would not use `chage` to set the actual password policies, which would include complexity requirements and length. That would be configured with the PAM settings.

When applicable, the date can be set in YYYY-MM-DD format. The password expiration date is based on the value of the “maximum number of days between password change” setting associated with the `-M days` switch shown in [Table 13-4](#).

In addition, it's possible to set requirements for password complexity with the help of PAMs. The key file is `/etc/pam.d/passwd`. Password complexity requires the use of the `pam_cracklib` and `pam_unix` modules. On Red Hat systems, you can find the key directive either in the `system-auth` or `passwd-auth` file in the `/etc/pam.d/` directory. As suggested by the last directive on the line, three login attempts are allowed.

```
password requisite pam_cracklib.so try_first_pass retry=3
```

On Ubuntu systems, you can find the key directive in the `common-password` file in the `/etc/pam.d/` directory:

```
password required pam_cracklib.so retry=3 minlen=6 difok=3
```

As suggested by the directives in the line, the minimum password length is six alphanumeric characters. The `difok=3` option means that new passwords must have at least three characters different from the current password. [Table 13-5](#) lists options that support password complexity.

TABLE 13-4 Command options for `chage`.

OUTPUT	COMMAND OPTION
Last password change	<code>-d YYYY-MM-DD</code>
Password expires	N/A
Password inactive	<code>-I YYYY-MM-DD</code>
Account expires	<code>-E YYYY-MM-DD</code>
Minimum number of days between password change	<code>-m days</code>
Maximum number of days between password change	<code>-M days</code>
Number of days of warning before password expires	<code>-W days</code>

TABLE 13-5 Password complexity options for `pam_cracklib.so` and `pam_unix.so`.

OPTION	DESCRIPTION
<code>minlen</code>	Minimum password length, in alphanumeric characters.
<code>dcredit</code>	Minimum number of digits in a password, in negative numbers.
<code>lcredit</code>	Minimum number of lowercase letters in a password, in negative numbers.
<code>ocredit</code>	Minimum number of other characters in a new password, in negative numbers.
<code>retry</code>	Number of tries allowed to confirm when changing a password.

nullok	Blank password allowed.
remember=n	<i>n</i> previous passwords are remembered; a repeat of from that list of passwords is not allowed.
obscure	A given password is compared to a previous password to make sure it's not a palindrome, different only with respect to case, or otherwise too similar.

Allowing Access from Legitimate Systems

You can regulate access by adding an appropriate module to the /etc/pam.d login configuration file. Specifically, the following account module supports access limits through the /etc/security/access.conf file:

Account required pam_access.so

The /etc/security/access.conf file includes examples of access limits by the root administrative user, by group, and by regular user. You can configure such limits by console, by domain name, and by IP address.

Monitoring Virtualized Hardware

Virtual hardware is sort of a contradiction in terms. By definition, it's some sort of software configured to emulate hardware. It may in fact be configured to connect to physical hardware. For example, virtual universal serial bus (USB) ports are frequently configured to connect to actual USB hardware. Virtual machine hardware can represent anything you might see in real hardware. A number of solutions for virtual machines can simulate everything required for a computer within a software framework. Although there are many commercial options for virtual machines, there are also open source options.

Virtual Machine Hardware

Of course, the hardware in a virtual machine can represent just about anything that might be seen on a physical system. The hardware does not even have to exist on the local system. For example, many virtual machine systems can simulate Small Computer Systems Interface (SCSI) hard drives on regular PCs. In some cases, it's even possible to set up 64-bit virtual systems on 32-bit hosts.

There are a number of advantages to using virtual machines. Everything runs through the host CPU, which manages the access to virtual memory. Meanwhile, the hypervisor, or the software that manages the guest operating systems, takes care of presenting the virtual machine with virtual hardware for access to services like a network interface or a disk drive. Virtual machines are very space efficient because you can create a large hard drive for the guest operating system but the space will not actually be used on the physical disk until it gets filled in the guest. If you were to create a 500-gigabyte (GB) virtual hard drive, for example, on a 2-terabyte (TB) physical disk but the initial installation of the OS only takes 20 GB, the space used on the 2 TB physical disk will only be around the 20 GB mark. You can also use virtual machines to isolate network access. If you are going to have a lot of virtual machines residing on a virtual machine host, however, you need to have some ability to manage and monitor them.

Virtual Machine Options

Options for virtual machines are wide and varied. You can install software-assisted virtualization solutions on just about any modern PC, even many netbooks. For administrator convenience, you can use several of these options to create and administer virtual machines remotely. A few of the major options for virtual machines include the following:

- **Kernel-based Virtual Machine (KVM)**—This is a Linux kernel module with a virtual machine monitor. It's the default open source solution for both Ubuntu and RHEL.
- **Xen**—This is a virtual machine monitor for Linux systems that often requires a specialized kernel. Red Hat support for Xen ended with RHEL 5. It's owned by Citrix.
- **VMware**—A **VMware** virtual machine includes a range of virtual machine options, from VMware Player for software-based virtual machines to vSphere as a bare-metal solution.
- **VirtualBox open source edition**—This software-based virtualization package is now part of the Oracle family of virtualization products.
- **Microsoft Virtual Server**—This is a solution that can run Linux systems only as guests.



NOTE

Because KVM and Xen require different kinds of changes to the kernel, you cannot use them simultaneously on the same system.

This is just a sampling of the virtual machine options that are available. Many options not included in this list are also excellent. However, because KVM is the preferred open source virtualization solution for both Ubuntu and RHEL, it's the primary virtualization solution covered here.

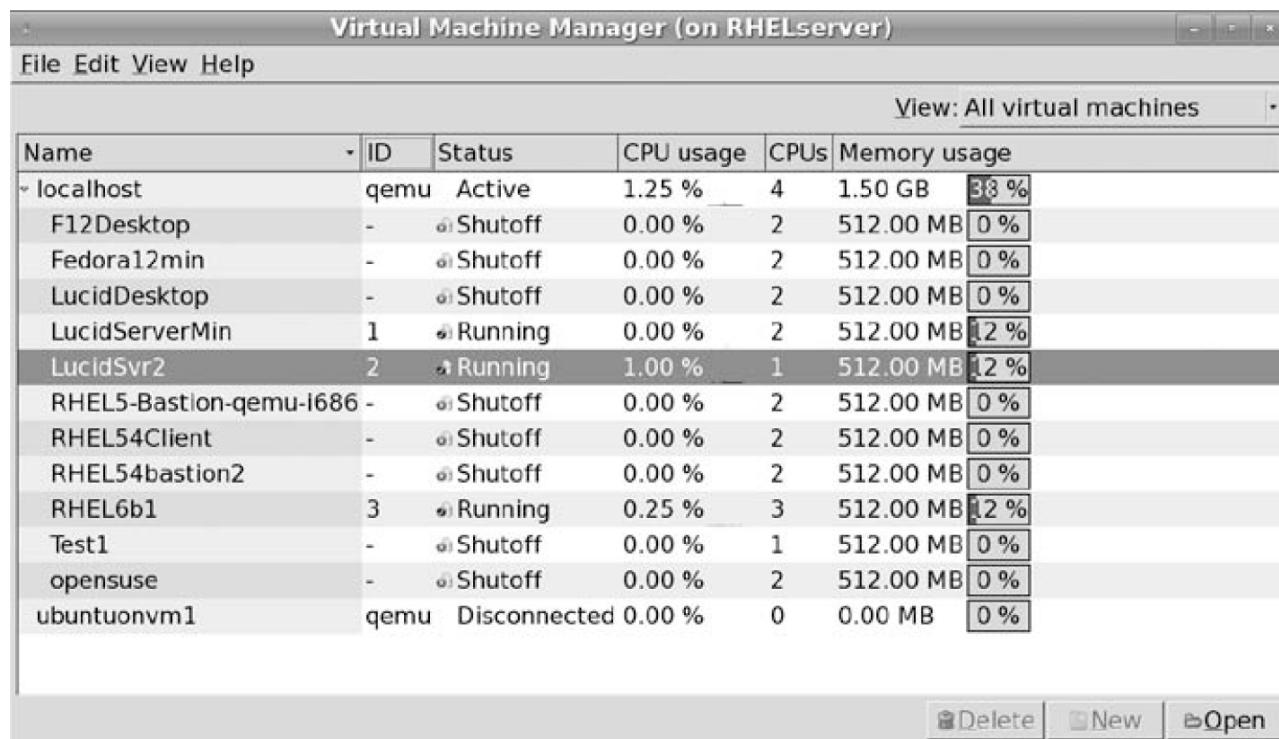


FIGURE 13-7

The Virtual Machine Manager.

Monitoring the Kernel-Based Virtual Machine (KVM)

The standard solution developed by Red Hat for managing virtual machines is the Virtual Machine Manager. The graphical interface available through the `virt-manager` command is fairly trivial and straightforward, as shown in [Figure 13-7](#). Note the Status, CPU Usage, and Memory Usage columns. These offer an easy way to

monitor the status of a group of virtual machine systems. If one system suddenly acts up, it may be a symptom of legitimate heavy use—or it may be a symptom of a system under attack.

Creating a new virtual machine is also a straightforward process. Just right-click the name of a virtual host and click New. A Virtual Machine Creation window appears.

If you are working from the command-line interface, you may prefer to use a command-line tool. The following command prompts for basic information associated with the virtual machine:

```
# virt-install --prompt
```

As with most virtual machine systems, the data of the simulated hard disks is stored as large files in dedicated directories. The contents of such disks change frequently, even if only to reflect log file changes for standard cron jobs. Therefore, any monitoring of a gold baseline virtual machine must be done with intrusion detection tools such as Tripwire or AIDE within that machine.

With the proliferation of virtual machines, it's important to have a tool to monitor what is happening on each machine on a virtual host. For this, you use a command-line monitoring tool, the `virsh` command. Unless you run it with administrative privileges, you will not see the full capabilities of that tool. Rather, you will see a virtualized interactive terminal with a `virsh #` prompt. [Table 13-6](#) lists some important commands at this prompt. For a complete list, run the `help` command.

TABLE 13-6 Virtual machine management commands at the `virsh #` prompt.

COMMAND	DESCRIPTION
list	Lists active domains
attach-disk	Supports the inclusion of physical devices, virtual images as drives, and ISO files as CDs/DVDs
attach-interface	Connects a virtual machine to network interfaces
dominfo	Provides status information on a domain
domblkstat	Notes read and write information of a hard-drive device on a domain
migrate	Moves a domain to a different host
reboot	Reboots a currently running domain
shutdown	Shuts down a currently running domain
start	Starts a currently inactive domain
vcpuinfo	Lists current CPU information for a domain, including numbers of CPUs
vncdisplay	Uses the Virtual Network Computing (VNC) system to open a display

Standard Open Source Security-Testing Tools

There are many excellent open source security-testing tools. This section only scratches the surface of some of the tools that are available. There are a number of categories and a number of tools in each category. Some common security testing tool categories are port scanning, intrusion detection, and vulnerability scanning.

As of this writing, nine of the top 10 network security tools can be configured on Linux. The top open security tools that can be configured on Linux include the following:

- **Wireshark**—This supports the analysis of network packets.
- **Snort**—An IDS, Snort has modes for packet analysis, logging, and interaction with iptables firewall rules. Associated rules are updated on a regular basis.
- **Netcat**—One utility that can read both TCP and UDP packets is **netcat**.
- **Metasploit Framework**—This is a platform for testing for code designed to take advantage of known vulnerabilities.
- **Hping3**—This is a command-line packet analyzer that sends TCP and UDP packets based on the features of the **ping** command.

Take care if you use any of these tools on remote systems. Make sure you have the permission of responsible administrators before applying any of these tools. Some of the tools require administrative privileges in order to function. Running any tool with administrative privileges can be dangerous, but it's especially dangerous to do so with tools that are designed to poke and prod remote systems.

This chapter focuses on the use of Snort and netcat.

Snort

Snort is a flexible IDS. It's actively maintained. With a set of rules that is regularly updated, Snort can help you keep local systems secure. It's readily available from the Ubuntu universe repository. RPM packages customized for Fedora and RHEL systems are available from the main Snort Web site at <http://www.snort.org/>.

When installing Snort on Ubuntu systems, the standard installation prompts you to configure a value for the **HOME_NET** directive, normally an IP address network in CIDR notation such as 192.168.0.0/24. That and related startup information is stored in the **snort.debian.conf** file, which is configured with the **snort.conf** file. Similar information for Red Hat systems can be configured in the **/etc/sysconfig/snort** file.

As noted in the *Snort Users Manual*, you can configure Snort to run in four different modes:

- **Sniffer mode**—This reads the contents of each packet, similar to Wireshark.
- **Packet-logger mode**—This stores the contents of packets.
- **Network IDS mode**—This supports packet analysis against a set of rules, typically downloaded from the Snort Web site.
- **Inline mode**—This works with iptables-based firewall rules to configure additional actions on certain network packets.

These modes are cumulative. For example, the lessons learned from running Snort in sniffer mode apply to all other modes.

Sniffer Mode

Sniffer is a commonly used colloquial term for a protocol analyzer. As such, Snort can look inside the contents of packets as they are sent across a network.

Sniffer mode may seem overwhelming, especially if you run Snort in verbose mode, with the **snort -v** command. If you do so and are overwhelmed by the packets that appear, you can stop the reading by pressing Ctrl+C.

The **snort -d** and **snort -e** commands go further. The **-d** switch dumps Application Layer data. The **-e** switch includes headers at the Data Link Layer, associated with communication at the hardware level.

Try these commands on their own. Review the rules as they apply to different services. If there is an error message related to missing rules, you may have to download them. The latest versions are available from the Snort Web site.

Packet-Logger Mode

If you have tried any of the commands in sniffer mode, you should realize the importance of being able to send the data to a log file. To avoid transmission problems associated with verbose mode, Snort can send information to a log file in compressed binary format.

As suggested in the *Snort Users Guide*, the following command sends information to a log file in a local log/ subdirectory:

```
# snort -vde -l ./log
```

The command works only if a local log/ subdirectory exists. You can create one as needed. Of course, you can modify the command to send logging information to a more standard directory:

```
# snort -vde -l /var/log/
```

The **snort** command by itself may not know what network to listen to, however. You can address that with the **-h** switch. For example, the following command listens for packets on the noted IP address network:

```
# snort -vde -l /var/log/ -h 192.168.0.0/24
```

After you collect logging information, you can analyze it with another protocol analyzer such as Wireshark.

Network IDS Mode

Network IDS mode depends on an appropriate Snort configuration file, starting with **snort.conf** in the **/etc/snort/** directory. That file should take information from rules as defined in the **/etc/snort/rules/** directory. For example, the following command uses the noted **snort.conf** file as a configuration file:

```
# snort -vde -l /var/log/ -h 192.168.0.0/24-c /etc/snort/snort.conf
```

You can use the **-c** switch on the configuration file of your choice. In other words, you can test network IDS mode with different Snort configurations.

Inline Mode

Snort inline mode requires the installation of the **libipq** development library to enable packet queuing. In brief, it enables a **QUEUE** mode for the **-j** switch associated with the **iptables** command. In other words, with the **libipq** development library, you can configure **iptables** to accept (**ACCEPT**), reject (**REJECT**), drop (**DROP**), or queue (**QUEUE**) a packet. You can link queued packets to Snort.



NOTE

Because neither Ubuntu nor RHEL include the **libipq** development library in their repositories, you should consider such software carefully before installing it.

Netcat and the **nc** Command

The **nc** command is short for netcat, a service that reads and writes over network connections. It's versatile. It can act like a **telnet** command to check for access over an available port. Or, it can act like an **nmap** command to scan a range of ports.

First, to demonstrate one simple capability, run the **nc** command on a target port on a supposedly secure Apache-based Web site on the Hypertext Transfer Protocol, Secure (HTTPS) port 443:

```
$ nc -v 192.168.0.1 443
```

When the following message appears, type **GET / HTTP** at the next line:

```
Connection to 192.168.0.1 443 port [tcp/https] succeeded!
```

You may see too much information about the Apache server on the noted IP address. You will want to deny attackers as much information as possible in this area. You can change the **ServerTokens** directive in the main Apache configuration file to limit the amount of information provided by the server. You can also use the **Header unset Etag** and **File ETag none** directives in the Apache configuration to better control files that are cached on the client systems.

Sometimes, connections with a simple response provide an attacker with a bit too much information for comfort. For example, the following command confirms the availability of port 111, typically used for the Sun Remote Procedure Call (RPC) protocol:

```
$ nc -v 192.168.0.1 111
Connection to 192.168.0.1 111 port [tcp/sunrpc] succeeded!
```

This is an important hint to an attacker. Port 111 and the Sun RPC protocol are frequently associated with relatively unsecure services such as the Network File System (NFS) and the Network Information Service (NIS). If an attacker receives such a message, that person will likely wonder whether he or she has hit the jackpot and will want to go further with respect to checking a group of port numbers. For example, the following command would quickly scan every port between 1 and 1024 on the noted IP address, randomly (-r), in verbose mode (-v), and just scanning for daemons that are listening on the noted ports (-z):

```
$ nc -rvz 192.168.0.1 1-1024
```

Try it out on a system and watch how quickly the information is actually scanned. Given the value of the **nc** command, it's worth some trouble to analyze the switches described in [Table 13-7](#).

TABLE 13-7 Command switches for **nc**.

SWITCH	DESCRIPTION
-e	This executes a given command or script from a server. On a client, it pipes requests to the noted server.
-g	This specifies a source route, effectively substituting a false source IP address in the packet when received by the destination.
-l	This binds to a given port if and when it's coupled with -p, to await incoming connections.
-n	This avoids hostname lookups. It's faster because it does not wait for responses from DNS servers.
-P	This associates with a local port number for a connection to a remote system. It may be used to spoof a port number or route connections through an unexpected and therefore possibly more secure route.
-r	This allows the nc command to use random ports.
-s	This allows the use of an IP address other than that of a network card.
-u	This specifies the use of UDP communication.
-v	This configures verbose mode.
-z	This scans for daemons that are listening.

You can use a number of tools to perform vulnerability scans from and against Linux systems. The earliest vulnerability scanner was developed to run on Unix systems. It was called the **System Administrator Tool for Auditing Networks (SATAN)**. Developed in 1995, it went on to spawn two other vulnerability scanners, including one commercial offering named **System Administrator's Integrated Network Tool (SAINT)**. In the intervening decades, a number of companies have entered the vulnerability scanning and management space. There are a number of good commercial vulnerability scanners, including Qualys and ISS.

Although vulnerability scanning started on Unix systems, some of the predominant scanners in the market today do not run on Linux. This section focuses on tools that run on Linux systems.

Nessus

Even though the code for **Nessus**, by Tenable Security, is no longer released under an open source license, it's still near the top of the list of security tools at <http://sectools.org/>. Nessus can perform remote vulnerability scans by checking open ports and banners. It can also perform local scans if it's provided with login credentials for the target system.

Although there used to be a standalone client for Nessus—and there still is for OpenVAS, which was forked from the open source version of Nessus—you access the Nessus UI through a Web browser. Just navigate to the hostname or IP address of the machine with Nessus installed using HTTPS and port 8834. For example, on a system with IP address 10.192.168.10, you would navigate to <https://10.192.168.10:8834/>.

After filters are set, a system is analyzed, and a report is created through the Web-based interface, the report can be analyzed by hostname or IP address along with analyzed protocol. For example, the Web browser view shown in [Figure 13-8](#) lists an issue of medium severity for port 25, using TCP communication. You should already know that port and protocol are associated with SMTP services such as sendmail and Postfix.

Over time, Nessus has become more commercial, targeting the needs of its enterprise customers. This can be seen through the introduction of scan configurations that are specifically geared toward satisfying auditing requirements. The Payment Card Industry (PCI), an alliance of companies responsible for the Data Security Standard (DSS) to which any company handling payment cards is required to adhere, has an audit requirement.

<input type="checkbox"/> Severity ▾	Plugin Name	Plugin Family	Count
<input type="checkbox"/> HIGH	PHP 5.6.x < 5.6.10 Multiple Vulnerabilities	CGI abuses	1
<input type="checkbox"/> HIGH	PHP 5.6.x < 5.6.9 Multiple Vulnerabilities	CGI abuses	1
<input type="checkbox"/> MEDIUM	HTTP TRACE / TRACK Methods Allowed	Web Servers	1
<input type="checkbox"/> MEDIUM	OpenSSL 1.0.1 < 1.0.1m Multiple Vulnerabilities	Web Servers	1
<input type="checkbox"/> MEDIUM	OpenSSL 1.0.1 < 1.0.1n Multiple Vulnerabilities	Web Servers	1
<input type="checkbox"/> MEDIUM	SSL RC4 Cipher Suites Supported	General	1
<input type="checkbox"/> MEDIUM	SSL Version 2 and 3 Protocol Detection	Service detection	1
<input type="checkbox"/> INFO	Nessus SYN scanner	Port scanners	2
<input type="checkbox"/> INFO	Service Detection	Service detection	2
<input type="checkbox"/> INFO	Common Platform Enumeration (CPE)	General	1

FIGURE 13-8

An excerpt from a Nessus report.

Companies that handle any payment card like a credit card have to perform regular audits against their infrastructure. Unlike the full, fast, or deep sorts of configurations that you may find in other vulnerability scanners, Nessus includes scan configurations that can be targeted at the PCI requirements. If you have auditing requirements for PCI or other regulations, having a preconfigured scan will save you a lot of time. You will not have to read through the PCI DSS and develop a scan configuration that will address all the requirements. Tenable Security has already done that. For more information on Nessus, navigate to <http://www.nessus.org/>.



NOTE

One reason Nessus closed its source was that the developers felt they were not getting adequate assistance from the community in the development and maturation of the program itself and, more importantly, in the plug-ins used to perform tests. Without the constant addition of fresh plug-ins, a vulnerability scanner is of little use.

OpenVAS

When Nessus closed source and became a commercial product, there was a void when it came to open source vulnerability scanners at a time when they were gaining in popularity and importance. System administrators who could not afford large payments for commercial scanners were left with an old version of Nessus that did not have signatures that were being updated. Alternatively, they could use SATAN, which had not been updated in years.

After Nessus closed its source code, a small group created a new project from the open Nessus source, calling it Open Vulnerability Assessment System (OpenVAS). Over time, OpenVAS has become a standalone project in its

own right with very little left of the original design or architecture of Nessus. OpenVAS also comes with a Web interface and a desktop client called the Greenbone Security Desktop. You can use either of these interfaces to generate a scan. One feature of OpenVAS is that you can perform a quick scan without going through the process of entering targets and selecting scan configurations. Most scanners have a multi-step process that is required before you can get a scan started. With OpenVAS, you can enter a target address and let OpenVAS go to work. [Figure 13-9](#) shows the Web interface for OpenVAS with a box in the middle of the screen where you can enter your target.

This does not preclude you from entering multiple targets and selecting a specific scan configuration. It is, though, a way to get a quick idea of the security posture of your systems. You can then follow up with more targeted scans in areas where you have additional concerns.

Nexpose

Nexpose, from Rapid 7, has several offerings depending on your needs and the number of systems you have to scan. Although it may not be strictly open source in a traditional sense, there is a community offering for Nexpose that allows you a free license for the software. It does, though, restrict you to 32 hosts. If you want to scan more than 32 hosts, you have to scan them 32 at a time, store the report, and then remove the systems from the database of hosts that Nexpose keeps track of. This is doable, though perhaps tedious. If you need more, you can pay for the ability to scan more at a time. The more IP addresses you need to scan, the more you pay for licenses.

The screenshot shows the OpenVAS Greenbone Security Assistant web interface. At the top, there's a navigation bar with links for Scan Management, Asset Management, SecInfo Management, Configuration, Extras, Administration, and Help. The user is logged in as Admin admin | Logout, and the date is Sat Jul 4 21:07:24 2015 UTC.

The main area displays a table of scan tasks:

Name	Status	Reports		Severity	Trend	Actions
		Total	Last			
Immediate scan of IP 172.30.42.20	Stopped at 10 %	0 (1)				
Immediate scan of IP 172.30.42.20	Done	1 (1)	May 31 2015	5.0 (Medium)		

(Applied filter: apply_overrides=1 rows=10 permission=any owner=any first=1 sort=name)

Below the table, there's a welcome message from a cartoon character:

Welcome dear new user!
To explore this powerful application and to have a quick start for doing things the first time, I am here to assist you with some hints and short-cuts.

I will appear automatically in areas where you have created no or only a few objects. And disappear when you have more than 3 objects. You can call me with this icon any time later on.

If you want help creating new scan tasks but also more options, you can select "Advanced Task Wizard" from the wizard selection menu at the top of this window where it currently says "Task Wizard" marked with a small arrow.

For more detailed information on functionality, please try the integrated help system. It is always available as a context sensitive link as icon .

Quick start: Immediately scan an IP address
IP address or hostname:

For this short-cut I will do the following for you:

1. Create a new Target with default Port List
2. Create a new Task using this target with default Scan Configuration
3. Start this scan task right away
4. Switch the view to reload every 30 seconds so you can lean back and watch the scan progress

In fact, you must not lean back. As soon as the scan progress is beyond 1%, you can already jump into the scan report via the link in the Reports Total column and review the results collected so far.

By clicking the New Task icon you can also create a new Task yourself. However, you will need a Target first, which you can create by going to the Targets page found in the Configuration menu using the New icon there.

FIGURE 13-9

OpenVAS Greenbone Security Assistant.

One important reason to bring up Nmap is that Rapid 7 also maintains Metasploit, the exploit framework. Since both are maintained by the same company, there is integration between the two products. If you have both Nmap and Metasploit, and Nmap knows about your Metasploit installation, Nmap will make use of Metasploit. If Nmap has found a vulnerability for which Metasploit has an exploit module, all you need to do is click to exploit. Nmap will call the correct module with the right parameters in Metasploit. It will then open a Metasploit console to show you the exploit and give you a shell if that is the outcome of the exploit. You may not get a shell if the exploit was not successful or if, for example, the point of the exploit was not to get a shell but instead to obtain a list of information from the system.

This can be an important tool for a security tester. You will save a lot of time by not having to open another tool, search for the right module, enter the correct parameters, and run the exploit. The outcome will be the same, but it saves a lot of time and effort. Considering that security testing is generally very limited on time, any time you can save will allow you to find more vulnerabilities to remediate. In the end, this will be beneficial to the security posture of your network and systems.

Where to Install Security-Testing Tools

If a system has been compromised, it's too late to install security-testing tools on that machine. Issues such as backdoors and Trojan horses could provide opportunities for attackers who might want to mislead you about what they are doing.

You generally do not want to install testing tools, particularly scanners, on every system. First, it's not necessary. Second, these tools can be used against you if a system is compromised. While this sentiment may have been expressed as a conditional, using the word *if*, you should always plan for your systems to be compromised. This means limiting what an attacker can do and what that person can get access to as well as giving yourself a lot of audit trails so you can determine what happened.

The number of security-testing tools that are available on live media such as CDs, DVDs, and USB keys suggests the excellence of such media as a secure location for security-testing tools.

Hint: Not Where Attackers Can Use Them Against You

Ideally, you should install security-testing tools only on well-hardened systems that are isolated from production networks. That leads to a bit of a loop, however. How do you know if a system is secure until security tools have been used to check a system? As strange as it sounds, there is a small element of faith involved. You can start with a reputable Linux distribution released by a company like Red Hat or Canonical. When those systems are up to date, you have a reasonable starting point to create a testing system. Of course, you can also use a distribution that is specifically targeted at performing security testing, like Kali Linux.

There is risk involved. Attackers could have somehow included their code in the packages released for a distribution. They may have penetrated the mirror servers where so many Linux distributions are downloaded and updated. They may take advantage of security issues before you can update a default system with fixes. There are safeguards, however. Open source developers test their software in the public eye. The source code is publicly available. If there are security issues, a great majority are revealed in the public testing process. Any issues that get by this process are publicized as soon as they are known by the community. This is not a perfect barrier, as made evident by some long-standing vulnerabilities that escaped countless developers. One example is Shell Shock, which is a vulnerability that existed in the Bourne Again shell for a couple of decades before it was discovered in 2014. However, it's a start.

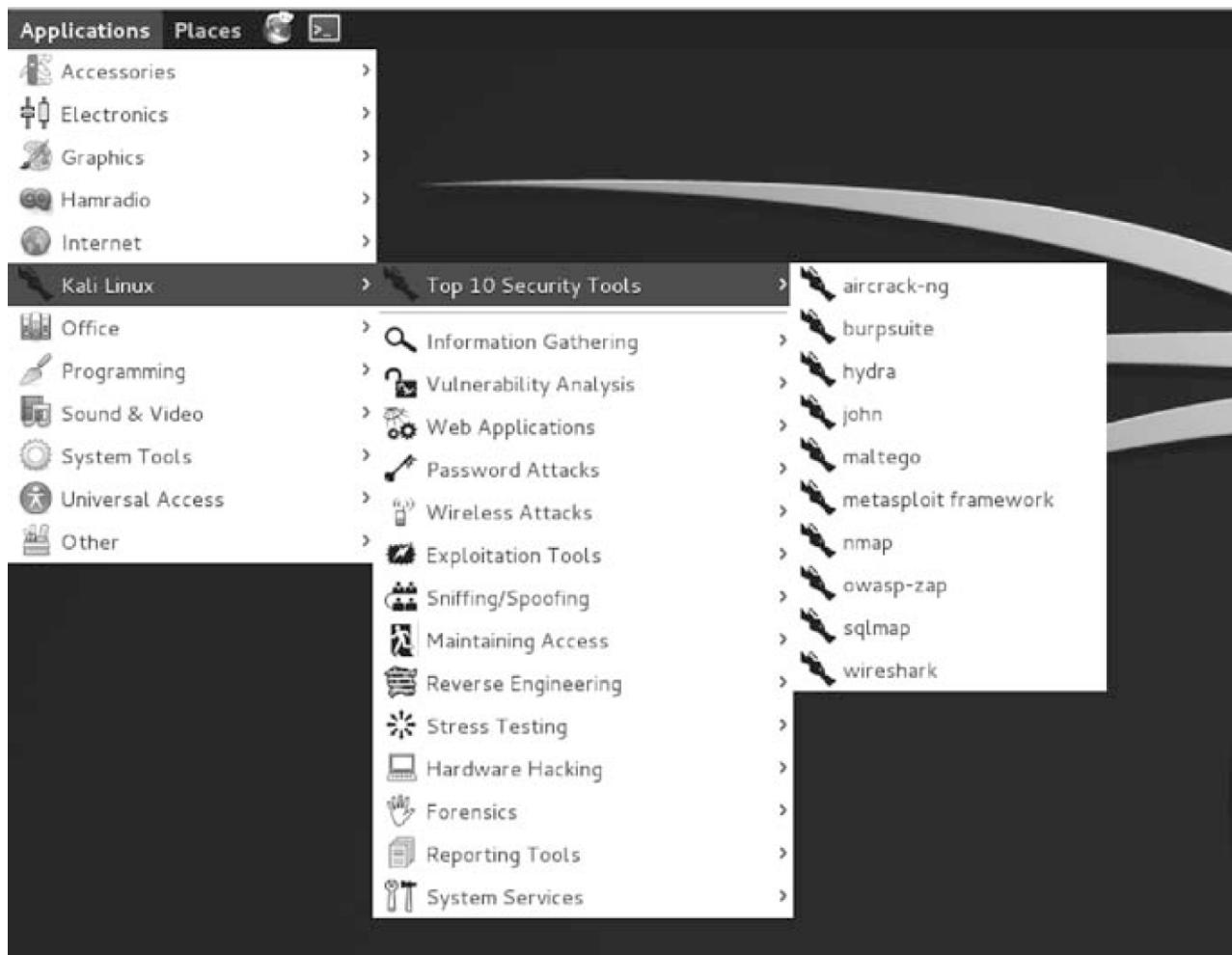
There are safeguards on the mirror servers that carry Linux distributions. Downloads with commands like `yum` and `apt -get` check GNU Privacy Guard (GPG) keys to ensure the integrity of a downloaded package. If you trust certain groups of developers, one option is to use security tools already included with live boot media.

Some Tools Are Already Available on Live CDs

One excellent option for security tools is live boot media, such as CDs, DVDs, and USB keys. Perhaps the model for live media for Linux systems is the **Knoppix** distribution. This live operating system, which can be booted directly from a CD/DVD drive or even a downloaded ISO file, is impressive. With the tools on the Knoppix distribution, you can back up or recover troubled media. You can reconfigure partitions and volumes.

Although there have been a number of Linux distributions over the years targeted at helping to perform penetration testing, including WHAX (previously Whoppix), Pentoo, and DEFT, perhaps the most predominant of these is **Kali Linux**. Kali, previously named BackTrack, has been updated after a period during which it lay dormant. A new group has taken responsibility for updating and maintaining it. The new group, Offensive Security, has been very active in keeping Kali up to date with as many security-testing tools as are available and at least reasonably stable.

Kali uses Ubuntu as its foundation distribution and adds a number of security-testing tools on top of it. In place of the newer Unity desktop, Kali uses an older implementation of GNOME. After all, the point of Kali is not to give a feature-rich desktop but to provide quick access to the tools needed by a penetration or security tester. [Figure 13-10](#) shows the menu system that Kali implements to provide reasonably quick access to its hundreds of testing tools.

**FIGURE 13-10**

The Top 10 Security Tools menu option in Kali Linux.

Kali includes all the top security-testing tools that are available under Linux. Additionally, it breaks out the tools into the following categories:

- **Information gathering**—During most security testing, an attacker would gather information about IP addresses, open ports, and other information that would be important to make plans. This includes using tools like nmap, dnsmap, hping3, sslstrip, and many others.
- **Vulnerability analysis**—There are a number of tools useful for identifying vulnerabilities, including system and network vulnerabilities. These tools include Cisco Auditing Tool, OpenVAS, and sqlmap.
- **Web applications**—Web applications are a common target. Burp Suite, sqlninja, DirBuster, and w3af are all tools you would find in this set of menus.
- **Password attacks**—When an attacker has the passwords to a system, he or she can maintain and maybe even extend access. As a result, password attacks are common. This set of tools includes John the Ripper and RainbowCrack, among many others.
- **Wireless attacks**—You may need to test 802.11 networks or Bluetooth access. As a result, redfang, bluesnarf, aircrack-ng, and many others are available in Kali.
- **Exploitation tools**—After you have gathered your information, you will want to make use of it. Metasploit is not the only exploitation tool available, but it's one of the most important and widely used tools in this category.

- **Sniffing/spoofing**—Attacks may require you to pretend to be another system. To do that, you need spoofing tools like Ettercap.
- **Maintaining access**—Attackers like to maintain access to systems they have compromised. Tools like HTTP Tunnel and Cryptcat serve that purpose and are available here.
- **Reverse engineering**—This set of tools will allow you to determine how different programs operate. This would include decompilers like Jad and debuggers like OllyDbg.
- **Stress testing**—As a security tester, you want to make sure your systems can stand up to load. FunkLoad, termineter, and others serve this purpose.
- **Hardware hacking**—You can use Kali to perform testing on devices like Android phones and tablets. Android SDK and APKtool will help perform testing against Android devices.
- **Forensics**—In addition to performing security testing, you can use Kali as an incident response tool. If systems have been compromised, you can use a Kali live boot disk to perform analysis to help determine the cause. To that end, you will find tools like Volatility and RegRipper here.
- **Reporting tools**—Ideally, when you are finished testing, you will want to generate a report on what you have found. Tools like dradis will help you store information in a useful manner so you can generate a report when you are finished.

Best Practices: Testing and Reporting

When testing a Linux system, it's important to test every part of a layered defense. Because some tests may require the temporarily disabling some defenses, it's appropriate to withdraw a system from production before running some tests. Open ports on a system should appear different inside a firewall from outside a firewall. Tests of TCP Wrapper depend on the rules configured in the /etc/hosts.allow and /etc/hosts.deny files. Tests of some services may be specific to the limitations configured within the configuration files for that service. Tools such as John the Ripper can help identify passwords that are too simple. Be sure to re-enable any disabled services after tests are complete.

Services are not accessible without open network ports. It's important to know what network ports are open on local systems to make sure that the only ports that are open are those ports required for local services.

Commands that can help check for open network ports include `telnet`, `netstat`, `lsof`, and `nmap`. `nmap` is especially versatile because it can scan different ports, protocols, and operating systems on a variety of target machines.

Several tools can help you check for unauthorized changes to key files. The `rpm -V` and `debsums` commands use checksums to inspect installed files. Because even normal changes to configuration files are highlighted, however, more precise tools are needed. Two options are Tripwire and AIDE.

Some systems have so many security rules, they are too difficult for people to use. This can lead to security problems. For example, when password policies are too strict, users who have trouble remembering their passwords may write them down at their workstations. You can manage password policies with the `chage` command. Access can be regulated with the help of PAM files.

Several options are available for virtual machine hardware, from VMware to VirtualBox. The emerging standard for an open source virtual machine manager is KVM. Although such virtual machines use large files that simulate hard disks, the contents of those files change frequently. Therefore, intrusion detection tools are important to monitor the integrity of such systems.

A variety of open source tools are available for security testing. Some of the major open source tools include Wireshark, Snort, and netcat. Snort includes modes for packet sniffing, packet logging, IDS, and inline interaction with the `iptables` command. The netcat command, `nc`, can identify a lot of information about open ports and the servers behind them, which highlights configuration issues with different services.

A variety of commercial security testing tools are also available. A number of open source options exist. You can use Nessus, OpenVAS, or Nexpose to perform vulnerability scans. All of these tools include Web-based interfaces, with reports that can identify security issues by service, port, and more.

You can use live media to boot to security-related Linux distributions like Kali, Pentoo, and others. A number of different developer groups have created dedicated live CDs/DVDs based on popular Linux distributions. If you trust these developers, you know that such systems will not be changed by attackers once burned to physical media. Of course, custom groups of security tools can also be configured on systems based on a restricted baseline and customized for local networks.



CHAPTER SUMMARY

In this chapter, you examined different tools for checking firewalls, inspecting network ports, and testing the integrity of files on local systems. Various methods for testing layered defenses are more likely to identify flaws and better ensure the integrity of a system. Be aware that security policies that are too severe can lead to other problems. Although it's important to test the strength of passwords with tools like John the Ripper, it's also important to make sure passwords do not have to be too complex to ensure user cooperation.

In this chapter, you also looked at a variety of options for open source and commercial penetration-testing tools. They vary from regular commands such as `nc` to open source options such as Snort. A number of open source tools have been collected together in live media distributions such as Kali and Pentoo.



KEY CONCEPTS AND TERMS

Bootstrap Protocol (BOOTP)

`debsums`

Hydra

John the Ripper

Kali Linux

Knoppix

`lsof`

`nc`

Nessus

Netcat

`netstat`

`nmap`

RainbowCrack

Security Administrator Tool for Analyzing Networks (SATAN)

System Administrator's Integrated Network Tool (SAINT)

VMware



CHAPTER 13 ASSESSMENT

1. Which of the following commands, when used inside and outside a firewall, can best test the effectiveness of that firewall?
 - A. `iptables`
 - B. `telnet`
 - C. `nmap`
 - D. `lsof`
2. Which of the following is *not* a password-cracking tool?
 - A. The `nmap` command
 - B. John the Ripper
 - C. Hydra
 - D. RainbowCrack
3. What is the `telnet` command that would connect to an open port 25 and an active server on a system with an IP address of 10.12.14.16?
 - A. `telnet 10.12.14.16`
 - B. `telnet 25 10.12.14.16`
 - C. `telnet 10.12.14.16 25`
 - D. `telnet 25:10.12.14.16`
4. Which of the following commands includes port information for TCP and UDP communication in numeric format?
 - A. `netstat -aunp`
 - B. `netstat -atnp`
 - C. `netstat -aund`
 - D. `netstat -atunp`
5. Which of the following commands can help you discover the active hosts on the 192.168.0.0/24 network?
 - A. `nmap -sP 192.168.0.0/24`
 - B. `nmap -sH 192.168.0.0/24`
 - C. `nmap -sh 192.168.0.0/24`
 - D. `nmap -sK 192.168.0.0/24`
6. You can install AIDE from supported repositories for both Red Hat Enterprise Linux and Ubuntu.
 - A. True
 - B. False
7. Which of the following commands sets a last password change date of April 1, 2015, for user michael?
 - A. `chage -c 2015-04-01`
 - B. `chage -d 2015-04-01`
 - C. `chage -E 2015-04-01`
 - D. `chage -I 2015-04-01`
8. What is the full path to the PAM configuration file that regulates password settings on a Linux system?
 - A. `/etc/init.d/pam/conf`
 - B. `/etc/common-auth`
 - C. `/etc/pam.d/common-password`
 - D. `/etc/passwd`
9. Which of the following virtual machine options is the default open source solution for the latest Red Hat and Ubuntu distributions?
 - A. KVM
 - B. Xen
 - C. VirtualBox, open source edition

- D. Hyper-V
- 10.** Which of the following commands opens a graphical tool that depicts the current CPU and memory load for virtual machines on the target host system?
- A. `virt-viewer`
 - B. `virsh`
 - C. `virt-manager`
 - D. `virt-top`
- 11.** Nessus is still released under an open source license.
- A. True
 - B. False
- 12.** Which of the following is not an option for bootable Linux systems with security testing tools?
- A. Kali
 - B. Pentoo
 - C. Netcat
 - D. WHAX

CHAPTER

14 Detecting and Responding to Security Breaches

Y

OU SPEND A LOT OF TIME protecting your systems from attack. You should keep one axiom in mind, however: Your system will be attacked, and there is a good chance it will be breached. This may come by accident or it may be deliberate. It may come from inside or it may come from outside. If you expect to be breached, you are more likely to be ready for an incident when it does happen. Think about all that you can and should do to help protect your system. Think of all of the ways these strategies can fail.

With that in mind, you should audit a system to identify baseline performance parameters. Deviations may be causes for concern. You will be able to identify user access through commands and log files. In addition, sensible policies can keep users within secure limits. Not all users are highly knowledgeable about computers, however, so you should monitor Linux user accounts for potentially risky behavior.

You must be ready if there is a problem. With appropriate forensic analysis tools, you may need to recover data from RAM while a compromised system is still running. To that end, different kinds of portable media such as live CDs are available. Some are suitable for compromised systems, others for test systems. Care is required, especially if such data may be used as evidence in official investigations.

After dynamic data is recovered and a system is powered down, you should handle static data. There are appropriate ways to copy hard drives without changing data on those hard drives. Investigations can then proceed with the copied drives. When investigations are complete, you should consider sharing the results with the open source community. In general, you can preserve corporate secrets and get the credit for finding a security flaw if you share appropriate results from any investigation.

Chapter 14 Topics

This chapter covers the following topics and concepts:

- How to perform regular performance audits
- How to make sure users stay within secure limits
- How to log access into the network
- How to monitor account behavior for security issues
- How to create an incident response plan
- What kinds of live CDs to have ready for forensics purposes
- What to do when to put your response plan into action
- What tools are appropriate for secure backups and recovery
- How to save compromised data as evidence
- How to recover from a security breach
- How and when to share with the open source community
- What best practices are for security breach detection and response

Chapter 14 Goals

When you complete this chapter, you will be able to:

- Understand typical system performance with appropriate user behavior
- Monitor user behavior through logins
- Know what to do when there is a security breach
- Understand when you should share information on a problem

Performing Regular Performance Audits

To determine whether there is a problem, you must have a baseline for a system—not only in terms of files but also in terms of behavior while the system is running. You should implement tools like Tripwire and the Advanced Intrusion Detection Environment (AIDE), but the abilities of those tools are by and large limited to the static characteristics of a system. You also need to know what happens dynamically. To that end, Linux provides some basic tools, such as the **ps** and **top** commands. Linux also allows the tracking of system status with the sysstat package. If you suspect or detect a problem, additional commands are available, such as **strace**, **ldd**, and **lsof**.

The Basic Tools: **ps** and **top**

The fundamental audit command for a running system is the **ps** command. As suggested by its man page, it provides a snapshot of currently running processes.

When run by itself, the **ps** command identifies the processes associated with the current login shell. That's not of much use, however. A better approach is to run the command with the **-u** switch and a username. In this way, users can audit the processes running under their own accounts. For example, the following command lists all processes associated with user michael:

```
$ ps -u michael
```

As an administrator, you must take a bigger-picture view of a system. To that end, the following command lists all processes (**a**) associated with users (**-u**), including those not associated with a terminal (**x**).

```
$ ps aux
```

The output is a database that can be saved as a file. It's sorted by user and process ID (PID) number. Perhaps just as important is the load on the central processing unit (CPU) and random access memory (RAM). Just be aware, occasional cron jobs scheduled in **/etc/crontab** or by individual users may skew the results. Because of this, you may want to run the **ps aux** command to determine a dynamic process baseline at a specific time. In addition, the load on certain processes, such as those related to the Apache Web server, may vary with the number of users who request information from that server.



NOTE

The **ps** command is one of those rare Linux commands for which not all switches require or even use a dash. Even though the **-u** switch normally includes a dash, the **ps** command shown here is the correct syntax of a command that lists all currently running processes.

As long as you are aware of the limits, it may be helpful to compare the output of the **ps aux** command at different times. Because the output is already sorted by PID number, an appropriate comparison of process names could help reveal suspicious processes. However, that is subject to processes that are constantly changing, such as hardware detection and cron jobs.

The **top** command provides a different kind of sorting of active processes. Take a look at the output in [Figure 14-1](#). By default, it organizes processes first by their load on the CPU and then on RAM. If you want to change the sort field, the directional keys (< and >) above the comma and period on a standard U.S. keyboard can help. Because **top** is a dynamic command, it's difficult to store the associated information in a file. This can be addressed in three ways. First, you can capture a screenshot of **top** command output from a GUI desktop. Second, you can list and store general memory information related to RAM and swap space with the **free** command. Third, you can list and store the load on the CPU and RAM with the **sysstat** package.

```
top - 17:33:44 up 3 days, 22:35, 5 users, load average: 1.20, 0.98, 0.78
Tasks: 173 total, 2 running, 171 sleeping, 0 stopped, 0 zombie
Cpu(s): 27.5%us, 3.7%sy, 0.0%ni, 67.6%id, 0.0%wa, 1.0%hi, 0.3%si, 0.0%st
Mem: 3366868k total, 2771344k used, 595524k free, 92972k buffers
Swap: 1959888k total, 40188k used, 1919700k free, 1031672k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
30621	michael	20	0	633m	375m	36m	R	32	11.4	316:59.01	firefox
7136	root	20	0	415m	74m	13m	S	10	2.3	152:17.59	Xorg
7927	michael	20	0	90748	45m	15m	S	6	1.4	9:42.12	gnome-panel
6763	asterisk	-11	0	29744	9724	5848	S	3	0.3	0:46.18	asterisk
7880	michael	20	0	13816	5032	3796	S	3	0.1	27:16.02	at-spi-registry
27340	michael	20	0	48864	16m	9564	S	3	0.5	0:00.74	screenshot
7926	michael	20	0	27364	18m	9016	S	2	0.6	13:13.02	metacity
7913	michael	20	0	17000	6036	4452	S	1	0.2	3:08.45	gnome-screensav
8077	michael	20	0	689m	387m	94m	S	1	11.8	128:50.33	soffice.bin
15521	michael	20	0	337m	205m	36m	S	1	6.2	63:26.20	acroread
2494	michael	20	0	45936	15m	8964	S	1	0.5	0:07.52	notification-da
8098	michael	20	0	138m	105m	8384	S	1	3.2	11:36.88	gnome-netstatus
7042	root	20	0	3304	1036	908	S	0	0.0	1:01.11	haldd-addon-stor
7933	michael	20	0	83628	28m	11m	S	0	0.9	1:40.52	gnome-terminal
28897	michael	20	0	6324	3812	1952	S	0	0.1	1:35.80	ssh
1	root	20	0	2844	1688	544	S	0	0.1	0:01.40	init
2	root	15	-5	0	0	0	S	0	0.0	0:00.00	kthreadd

FIGURE 14-1

The **top** command provides a view of active processes.

The System Status Package

The system status package, **sysstat**, includes a number of commands that can report system status in a regular log. It includes the basic information associated with the **top** command, reported on a regular basis. When the **sysstat** package is loaded and started, you can list the system status for the current day with the following command:

```
# sar -A
```

Unfortunately, the Ubuntu version of the system status package can collect only one month's worth of daily load reports. If you are paying attention, however, that should be enough. If you want to pull a report from, say, the 13th of the month, run the following command:

```
# sar -f /var/log/sa/sa13
```

Although the system status package for Red Hat Enterprise Linux (RHEL) 7 collects data for only seven days by default, you can configure it for as many days as needed in the `/etc/sysconfig/sysstat` configuration file.

For Additional Analysis

If you see a suspicious file or process, further investigation may be warranted. The **lsof**, **ldd**, and **strace** commands can help in that investigation.

Analyzing Processes with **lsof**

The **lsof** command lists all files opened by currently running processes. When the **lsof** command is run with the full path to a daemon, it can tell you more about the processes associated with that daemon. For example, the following command should tell you about the multiple Samba server processes running on the local system. Note that **lsof** requires administrative privileges and the full path to the daemon.

```
# lsof /usr/sbin/smbd
```

Identifying Associated Libraries with **ldd**

The **ldd** command lists the libraries associated with another command. It's useful with regular commands such as **ls** as well as daemons such as Samba. On your own systems, take a look at the output of these commands:

```
$ ldd /usr/sbin/smbd
$ ldd /bin/ls
```

Tracing System Calls with **strace**

The **strace** command goes further, tracing the system calls associated with a command. Many of these calls should go through the libraries just listed from the **ldd** command. If a target command such as **ls** is not performing its intended function, the **strace** command can tell you more. For example, if you suspect that the **ls** command has been replaced by malware, the following command traces the system calls associated with that command:

```
$ strace ls
```

The last few calls before the write commands in the output are normally key. If the suspect command is not running appropriate system calls when compared to what is seen from a gold baseline system, you should investigate further. Because there is a lot of output from the **strace** command, the **-o** option to write the output to a text file can be especially helpful:

```
$ strace -o analysis.txt ls
```

Making Sure Users Stay Within Secure Limits

User security starts with the password. As reported on ZDNet, a survey by the organizers of Infosecurity Europe 2004 offered a group of commuters a chocolate bar in exchange for their network-access password. Over two-thirds of the commuters agreed to the request. Of course, there is no evidence that the passwords revealed were in fact the commuters' actual passwords, as they were never verified. Even so, far too many users are not aware of the importance of account security. And although this particular survey took place many years ago, similar patterns of behavior remain common today. For this and many other reasons, Linux distributions include drivers that can connect to smartcard readers and biometric-verification tools.

Considering how many tools are installed on Linux by default on a standard installation, not to mention the administrative access to which Linux users have access, a Linux user can cause a lot of security issues. Because of this, education is important. In an enterprise situation, it's essential. In addition to education, though, there are other controls that can be implemented.

Appropriate Policies

Although Linux users are generally more knowledgeable, it's still best to keep policies as simple as possible. Generally, the policies that apply to regular users can be divided into five areas:

- **Physical access**—Security starts with access limits on computers. For regular workstations, this can be consistent with access to sensitive company documents.
- **Internet access**—Be clear with your users so they understand limits on access. If you monitor where users go online, tell them. Make sure users understand what is considered unauthorized Internet access and what penalties may be associated with such access.
- **E-mail access**—Users must know what is or is not allowed in e-mail communications. If you monitor their e-mail communications, tell them. Have a clear policy on the storage of old e-mails.
- **Remote access**—Anyone allowed remote access to systems has special responsibilities. Limits on access times may be appropriate, depending on the availability of connections. If remote access is available only by telephone modem, limits can increase availability to more users.
- **Passwords**—Password policies may be a challenge. Users must know policies with respect to sharing passwords or posting them in plain sight. They also must adhere to any complexity and rotation requirements that are in place.

Additional policies may be required for other communication systems, such as for instant messaging inside and outside the company. These are just policies for regular users, however. Policies for administrators should go further. Physical access policies for administrators extend to any facility with servers or other secure systems. Remote access policies should support rigorous access via Secure Shell (SSH), such as the use of passphrases. Backup policies should also ensure the security of sensitive information stored on backup media, an often-overlooked concern.

Education

As more and more people use computers, and more and more users move toward Linux as a desktop operating system, there are concerns about usability. The moment systems become more usable, they become open to attack due to the introduction of more software, functionality, and complexity. As a result, it's essential to make sure users are educated about the risks that are introduced the moment they start using a computer.

Even if there is no budget for formal classes, it helps to make educational materials available. Many Linux users believe their systems are invincible. They may be less concerned about Linux as a malware carrier. Although it's true that most malware from Internet attack sites or e-mail attachments may not affect a Linux system, some malware may still use Linux as a carrier, infecting other operating systems—for example, through shared folders.

User Installation of Problematic Services

Linux users may experiment with their systems. Such experiments may be as simple as installing a package from the Ubuntu Universe repository. These experiments may go further when users download and compile a third-party package. Such experiments are an excellent way to learn more about Linux. However, they may also pose risks to your corporate networks.

If users need applications not currently in your baselines, they should be able to request them. If such applications are supported by an organization such as Red Hat or Canonical, you should be able to provide them for your users. Of course, such requests should be subject to security reviews. If a user is asking for the installation of a Telnet server, you should be ready to educate that user on the benefits of using the SSH service instead.

One way to address this sort of issue is to put local repositories on your corporate network. This minimizes the risk associated with giving users full access to all packages in Internet repositories. It's not a perfect solution because there are ways to add other repositories, but it's one way to prevent your enterprise systems from having risky software packages installed.

Logging Access into the Network

There are two basic ways to see who has logged into a system and a network. One is to use a command to review users who are currently logged in. Another is to review logins in log files. Log files also reveal direct and indirect access to the root administrative account. System authentication logs can do more, depending on how logging services have been configured.

Identifying Users Who Have Logged In

Two commands identify users who are currently logged into a system: **w** and **who**. As shown here, the **who** command displays currently logged in users along with remote login locations, if applicable:

```
$ who
Michael    tty1      2010-05-07 08:26
Michael    pts/10     2010-05-06 22:46 (192.168.122.177)
Michael    pts/11     2010-05-07 07:25 (10.168.0.1)
```

The output shows three logins for user michael. The first is on the first local console, tty1. The second is from a known virtual machine with an Internet Protocol (IP) address on the associated private network. The third is from the IP address of a router, which suggests a remote external login forwarded to the local system. Unless user michael is in two places at one time, that external login may be a cause for concern. For more information, try the **w** command, as shown:

```
$ w
michael  tty1      -          08:26  39:01 0.01s  0.01s  -bash
michael  pts/10     192.168.122.177  22:46  10:19m0.02s  0.02s  -bash
michael  pts/11     192.168.0.1      07:25  0.00s        0.05s  0.01s
➥ trojan_horse
```

The **w** command includes information on the process currently being executed by the logged-in user. The output related to the terminal associated with the remote external login suggests that this user is currently running a program called *trojan_horse*. Although it's unlikely that a true attacker would run a program with such an obviously suspicious name, that output would warrant further investigation.

The **last** command provides a more complete history of logins to the local system. That history can help in an investigation to look for other suspicious logins. Just keep in mind that the **last** command sometimes provides misleading output. If you run the **last** command on an older system, it may lead to false positives for remote access. For details, search for bug 82540 in the **Red Hat Bugzilla** database at <https://bugzilla.redhat.com/>. (The Red Hat Bugzilla is a system for bug reports on Red Hat distributions.) In any case, the **last** command accesses a login database in the /var/log/wtmp file.

On Red Hat systems, the **utmpdump /var/log/wtmp** command includes the last process run by the logged-in user, by PID number. If you need to investigate the actions associated with a login session, that PID number—when coupled with system status log information described earlier in this chapter—can help you determine what that user was running.

System Authentication Logs

Suspicious logins as regular users should cause concern. An example of this might be if you see a user log in after regular business hours. Of more concern, likely, is suspicious use of the root administrative account along with related commands such as **su** and **sudo**. Authentication logs are associated with the **auth** and **authpriv**

facilities as configured in the system logging or the rsyslog services. The log files specified in the associated system-logging or rsyslog service configuration should contain information on all login attempts, including the following:

- **Remote access**—Remote access from any system is logged by IP address, service, and port number.
- **Administrative access**—Local access with the `su` or `sudo` command is logged either with the root administrative user ID of 0 or the associated `USER = root` environment variable.
- **Administrative jobs**—Access by services that require administrative privileges, such as a cron job or an e-mail transmission from Postfix, may also be logged.

Monitoring Account Behavior for Security Issues

On a corporate network, you may need to look for signs of inappropriate behavior in user home directories. If a user has downloaded source code on his or her system, that user may have already compiled that code into a binary program of some sort. In that case, there will be files somewhere in the tree of the user's home directory with executable privileges. Of course, such checks are subject to organizational policies. In general, it's best if users know about such checks before they are performed so they are aware of the expectations with respect to their behavior. Note, too, that such checks may not be reasonable if users are in fact Linux developers or testers.

Downloaded Packages and Source Code

A simple way to monitor a system for source code is to look for tarballs. These are normally provided in one or two compressed file archives, normally with a `.tar.gz` or `.tar.bz2` extension. You can find them in user home directories with the `find` command. Specifically, the following command starts with the `/home/` directory and looks in all subdirectories for files with the `.tar.gz` extension:

```
# find /home/ -name *.tar.gz
```

Such commands can be run on a regular basis as a cron job. For example, to set up a daily cron job with such searches, add a command like the one shown to a file in the `/etc/cron.daily/` directory. Make sure the output of the command is redirected to an appropriate file, perhaps in the `/var/log/` directory, suitable for later analysis.

Of course, Linux software comes in other formats. Software suitable for Red Hat distributions is downloaded as a package file with an `.rpm` extension. In contrast, software packages for Debian-based distributions such as Ubuntu are available as package files with `.deb` extensions.

Executable Files

Naturally, users may set up their own executable files in their home directories, with the `chmod` command. Although a few executable files are necessary for certain applications, such files should be kept to a minimum. The `find` command can also be helpful in this area. For example, the following command searches for all files (`-type f`) with user, group, and other executable permissions, starting with the `/home/` directory:

```
# find /home/ -type f -perm /u=x,g=x,o=x
```

It's possible that files other than executables may have the executable bit set. Some files may require the executable bit to function well. Of course, directories also require the executable bit, but by indicating a type of `f`, meaning file, you remove directories from the potential output.

Perhaps as dangerous as executable files are those with set user ID (SUID), set group ID (SGID), and sticky-bit permissions. Once created by a user, an attacker may arrange to have them run by a user associated with a service, potentially infecting that service with malware of some sort. You can use the `find` command to search for such files as follows:

```
# find /home/ -type f -perm /u=s,g=s,o=t
```

Although changes in file-executable permissions are tracked by tools such as Tripwire and AIDE in most of a system, they might not be tracked in user home directories. There are legitimate uses for executable scripts. For example, engineers might have big database jobs that they want to run at night through the cron daemon. Therefore, the tracking of executable files in user directories is often left to customized searches with the `find` command.

Creating an Incident Response Plan

Plan to be breached. Do not assume that Linux is invulnerable to attack just because some Linux users say so. It's true that more users employ other operating systems, such as Windows, so the volume of attacks and the number of potential vulnerabilities on that operating system are higher. However, Linux systems are still vulnerable. This is especially true when you put Linux in the hands of someone who does not take the responsibilities of a system administrator seriously. Any system that is poorly administered can be breached. And the system of any user who does not take security seriously and is not careful about the Web sites he or she visits and the e-mail messages he or she opens may be subject to attack.

Some Linux software has a long history of being vulnerable to exploitation. These software packages tend to be ones that are exposed to the outside world. None of this is to condemn either Linux or users, however. When it comes to security issues, it's often necessary to be blunt. You should never assume that any operating system, in and of itself, will protect you against attacks. You must prepare yourself for how to respond when you are attacked or, worse, breached.

With an appropriate incident response plan, you will do what is required to collect information. Correct procedures can help you determine what went wrong and what should be done next. In addition, such procedures can help you preserve information as evidence in case the organization decides to seek legal remedies.



NOTE

There may be legal requirements related to gathering potential evidence and reporting. You should be aware of any laws and regulations in your area with which you may need to comply.

Increased Vigilance

Every organization with valuable data should have computer security policies in place. If you want to pay more attention to user activity, start with the log files. Look at the history of commands run by a specific user. If a user has stuck with the bash shell, that history is normally stored in that user's home directory, in the `bash_history` file. Of course, a careful attacker may substitute a file with benign commands, but not all attackers are so careful. Any access to administrative accounts should be documented in files such as the `secure` and `auth.login` files in the `/var/log/` directory. You can check any changes to systems accessible to suspected users with intrusion detection systems (IDSs) such as Tripwire and AIDE.

If there is a reason to suspect malicious activity from a local user, you may not be able to share that information with other users. If you are able to share concerns with other users, it's more likely they will accept reasonable increased security measures in the areas discussed earlier. To recap, when appropriate, security can be increased in the following areas:

- **Physical access**—This includes limits on users who are allowed to operate certain workstations.
- **Internet access**—This includes limits on access outside corporate intranets.
- **E-mail access**—This includes restrictions on communications outside the company.
- **Remote access**—This includes reduced access to organizational networks.

- **Passwords**—This includes immediate changes to all passwords.

These are just examples of what you can do to increase security, based on less-rigorous policies. Actual policies in standard and higher-alert situations will vary.

Should You Leave the System On?

There are a number of approaches you can take after an attack has been identified. Some of these approaches are discussed here. However, none of these should be considered a foolproof approach to responding to system breaches. Your organization must identify the set of strategies that will work best for you.

FYI

You can choose to simply get back online as quickly and cleanly as possible. However, this will reduce your ability to learn from what happened and implement changes in your systems to protect against future attacks. It will also limit or remove your ability to gather evidence that could be used against an attacker. While this may not be as important if you are dealing with an attacker in another country, you should not assume this is the case. The reality is, the majority of attacks are perpetrated by insiders. If your system is attacked by someone inside your organization, you will likely want to prosecute that person.

The first thing to do is to create a team that will be in charge of incident response. This team should take the needs and policies of the business into consideration and develop an incident response plan. This should include business aspects as well as technical aspects.

One of the first technical challenges your team must address is whether to leave the system on or pull the plug, and whether to leave it connected to the network or remove it. As with every other decision, you should be aware of the consequences. If you leave it on, you run the risk of the attacker doing more damage. If you shut it down, you run the risk of a kill script cleaning out evidence. If you do not want to leave it on, the best approach is to just pull the plug. Yes, you might corrupt sectors on the disks. But a regular shutdown could invoke a script put in place by the attacker that would wipe evidence. A little corruption is preferable to having all relevant evidence erased.

If you decide to leave the system up, are you going to collect evidence? If so, you may need to leave the system connected to the network as it is. However, you should probably surround it with IDSs and firewall rules to make sure the attacker is not using this compromised system to leapfrog onto other systems on your network. The moment you identify this system as being compromised, you should consider it untrustworthy and potentially hostile. The more you can isolate it, the better you will protect the rest of your network.

Be careful here, though. If you are really looking to collect evidence, you do not want to make any big changes on the system. This will tip off your attacker, who may pull out, preventing you from getting the evidence you are looking for. Worse, it may cause the attacker to damage the system or any other system or network equipment he or she can access. This is a risk you have to take if you choose to gather evidence.

WARNING

States have breach notification laws. They vary from one location to another, but your business may fall under such laws. If you are breached, you may be required to notify your customers about it. Make sure you are aware of any laws to which you may be subject.

Acquiring the Memory Contents

Regardless of what you decide with regard to keeping your systems up and running, one thing you need to do while the system is still up is to acquire the contents of the memory. Because data in system memory disappears

when you power a system off, you must leave the system on to collect information about network connections, running processes, and users logged in. This information will give you a good snapshot of what is going on with your system.

To achieve this, you could use a variety of common Linux commands, like `ps`, `who`, and `netstat`. However, you may not be able to trust the binaries on your system. The attacker may have installed a rootkit, which could have replaced any or all of those utilities to protect the access the attacker may have gained. A better approach is to capture the memory of the system. Although there are physical ways to do this, they can be expensive and are not always reliable. Instead, you can use software to do it while the system is up and running. This requires a kernel module. Building a kernel module requires build tools and kernel source. This is one area where you can prepare yourself ahead of time.



NOTE

In older versions of Linux, you could access physical memory by accessing `/dev/mem`. However, that access was removed several years ago because it was a security risk. If you have access to memory, you can extract sensitive information that may belong to other users. You can also potentially overwrite sections of memory.

When you have a system baseline, build the kernel module that matches the kernel for all your systems and have it available to load up on systems.

One kernel module that you can use is called `fmem`. This software installs a device in the `/dev` tree that you can use to access physical memory. When you have the `fmem` kernel module loaded, you can acquire a memory capture using `dd`.

```
dd if=/dev/fmem of=memcap.dd bs=1M count=2048
```

The `dd`, or disk dump, command is commonly associated with gathering disk dumps. It supports a full copy of all data on a partition, volume, or drive. However, it can also be used to read or write directly from any device. The input file is the new memory device `/dev/fmem` and the output file is whatever file you choose. You specify a count based on the size of your memory. Make sure you have adequate disk space when you acquire this data. You may also want to consider writing it directly to an external disk like a USB stick or hard drive. This keeps your attacker from discovering it sitting on the drive, even if it's only there temporarily. Another approach is to write it out over the network, which avoids even putting a USB stick on the system. You can use netcat (`nc`) to perform this particular feat.

```
dd if=/dev/fmem bs=1M count=2048 | nc storage.example.com 4500
```

On the receiving end, you will need a netcat listener:

```
nc -l 4500 | dd of=memcap.dd bs=1M
```

After you have your memory capture, you can use a software package like Volatility or Rekall to extract information from the memory image. You will also need to make sure you have a way of verifying the image later on. You can use a cryptographic hash like SHA1 or MD5 to get a value you can compare against later. Although both of these hash algorithms have potential issues when it comes to strong cryptography, for the purposes of what you are doing here, either would work.

Before you start using either Volatility or Rekall, you need do some additional work. You need a specific memory description that relates to your system so the program will know where in the image to locate the information. Each kernel build stores its information in slightly different places. You need the `/boot/System.map` as a starting point for these tools to know where to look for things like network connections. There is a lot of information stored in memory. [Figure 14-2](#) shows a sampling of the plug-ins you can use against a Linux memory image in Volatility to extract data that you can use in your incident investigation.

```
kilroy@milobloom:~$ volatility --info | grep linux
Volatility Foundation Volatility Framework 2.4
linux_apishooks           - Checks for userland apihooks
linux_arp                  - Print the ARP table
linux_banner               - Prints the Linux banner information
linux_bash                 - Recover bash history from bash process memory
linux_bash_env              - Recover bash's environment variables
linux_bash_hash             - Recover bash hash table from bash process memory
linux_check_afinfo          - Verifies the operation function pointers of network protocols
linux_check_creds            - Checks if any processes are sharing credential structures
linux_check_evt_arm          - Checks the Exception Vector Table to look for syscall table hooking
linux_check_fop              - Check file operation structures for rootkit modifications
linux_check_idt              - Checks if the IDT has been altered
linux_check_inline_kernel    - Check for inline kernel hooks
linux_check_modules           - Compares module list to sysfs info, if available
linux_check_syscall          - Checks if the system call table has been altered
linux_check_syscall_arm      - Checks if the system call table has been altered
linux_check_tty               - Checks tty devices for hooks
linux_cpuinfo                - Prints info about each active processor
linux_dentry_cache            - Gather files from the dentry cache
linux_dmesg                  - Gather dmesg buffer
linux_dump_map                - Writes selected memory mappings to disk
linux_elfs                   - Find ELF binaries in process mappings
linux_enumerate_files         - Lists files referenced by the filesystem cache
linux_find_file               - Lists and recovers files from memory
linux_hidden_modules           - Carves memory to find hidden kernel modules
linux_ifconfig                - Gathers active interfaces
linux_info_regs               - It's like 'info registers' in GDB. It prints out all the
                                - Provides output similar to /proc/iomem
                                - Lists files that are opened from within the kernel
                                - Parses the keyboard notifier call chain
                                - Compares the output of proc maps with the list of libraries from libdl
                                - Lists libraries loaded into a process
                                - Dumps shared libraries in process memory to disk
                                - List applications with promiscuous sockets
                                - Gather loaded kernel modules
                                - Lists open files
                                - Looks for suspicious process mappings
                                - Dumps the memory map for linux tasks
                                - Extract loaded kernel modules
                                - Gather mounted fs/devices
                                - Gather mounted fs/devices from kmem_cache
                                - Lists Netfilter hooks
                                - Lists open sockets
                                - Enumerates processes through the PID hash table
                                - Writes per-process packet queues out to disk
                                - Scan ELF binaries' PLT for hooks to non-NEEDED images
                                - Gathers process maps for linux
                                - Gathers process maps for linux through the mappings red-black tree
                                - Dumps a process's executable image to disk
                                - Checks for signs of process hollowing
                                - Gathers processes along with full command line and start time
                                - Gathers processes along with their environment
                                - Gather active tasks by walking the task_struct->task list
linux_lsmod                  - 
linux_lsof                   - 
linux_malfind                - 
linux_memmap                  - 
linux_moddump                - 
linux_mount                  - 
linux_mount_cache             - 
linux_netfilter               - 
linux_netstat                - 
linux_pidhashtable            - 
linux_pkt_queues              - 
linux_plthook                 - 
linux_proc_maps               - 
linux_proc_maps_rb             - 
linux_procdump                - 
linux_process_hollow            - 
linux_psaux                   - 
linux_psev                    - 
linux_pslist                  - 
```

FIGURE 14-2

Plug-ins you can use against a Linux memory image in Volatility to extract data for an investigation.

Although you will commonly acquire memory that will get you all the information you need from the running processes, you will only get the processes that are actually in memory. You will not get processes that have been swapped out of RAM into swap space. In some cases, you may want to set up a full copy of the /proc/ directory on separate media such as a USB key. Linux /proc/ directories normally include subdirectories full of information associated with each running process. (See [Table 14-1](#).) If there are any discrepancies between these

subdirectories, the output of various `ps` commands may provide important clues, such as processes that have been hidden by an attacker.

If you have configured a specialized service on a compromised system, do not forget any dynamic settings associated with that service. In general, such dynamic settings should be available in RAM and saved with the `/proc/kcore` file. This is a file that dynamically represents the contents of RAM on the local system. However, you may choose to run a command like `sync` to make sure information currently in RAM is written to appropriate files.

After the recovery of dynamic data is complete, you should be able to power off the system. The next step will be to remove and replicate hard drive media on the compromised systems.

TABLE 14-1 Important files in the `/proc/` directory for forensic analysis.

FILE	DESCRIPTION
<code>cpuinfo</code>	This contains hardware information focused on the CPU.
<code>domainname</code>	This contains the name of the domain for the local area network (LAN); it's located in the <code>/proc/sys/kernel/</code> directory.
<code>hostname</code>	This includes the name of the local computer. It can be found in the <code>/proc/sys/kernel/</code> directory.
<code>kallsyms</code>	This includes debugging information for the kernel.
<code>modules</code>	This contains the loaded kernel modules.
<code>mounts</code>	This contains mounted filesystems. It can be found in the <code>/proc/self/</code> directory.
<code>partitions</code>	This includes configured partitions.
<code>swaps</code>	This includes configured swap volumes.
<code>uptime</code>	This shows the length of time the current system has been in operation.
<code>version</code>	This shows the currently loaded kernel, with version number and associated compiler.

FYI

There are a few live CD Linux distributions targeted at forensics work. Some can even be used directly on compromised systems to recover dynamic data. These tools include **Digital Evidence & Forensic Toolkit (DEFT)** and **Computer Aided Investigative Environment (CAINE)**. These distributions have the capabilities discussed in this chapter, including the ability to investigate memory dumps. For more information, see <http://www.deftlinux.net/> and <http://www.caine-live.net/>.

Having Live Linux CDs Ready for Forensics Purposes

You should have two different types of CDs (or other similar media) available for forensics purposes. Some are suitable for mounting on a compromised system to help the investigator download and otherwise save dynamic data from areas such as RAM. If you suspect a system has been compromised, you should not trust any of the executables on that system. They may have been replaced with substitutes that will hide the existence of malware. Others are suitable as forensic live media. As such, they can be used for booting diagnostic systems. Once booted, such media can be used to save a copy of the partitions and volumes configured on compromised systems.

Although you can certainly purchase expensive commercial software to help with a forensic investigation, such as EnCase or FTK, you can also go open source. Linux has a number of great utilities built in, like dd, md5sum, and sha1sum. You can use these to acquire and verify potential evidence. There are also other programs you can use to perform digital investigations. You have already explored Volatility and Rekall for accessing data stored in system memory. The other place you will want to look is on the disks. You could use the dd command to mount the disks or disk images as read-only and investigate using tools like `find`, `ls`, and `cat`. This may not give you everything you need, though. You might need to dig deeper. For this, you can use a special set of forensics tools called **The Sleuth Kit (TSK)**. [Table 14-2](#) shows a list of commands included in TSK.

You can certainly create your own incident response distribution, either completely from scratch or based on another distribution. Fortunately, however, you do not have to go to that effort. There are a number of Linux distributions, some of which can be used as a live CD, that are targeted at forensics work.

Helix Live Response

One solution you can use to investigate a compromised Linux systems is Helix. This option no longer exists in the open source sphere, however. Perhaps the popularity of Helix encouraged its developers to commercialize this Linux distribution.

Helix is based on the Knoppix live CD. Although it can be booted like the Knoppix live CD, it can also be mounted directly on a suspect system. Key commands are available directly from the mount point.

TABLE 14-2 Commands in TSK.

COMMAND	DESCRIPTION
<code>blkcalc</code>	This maps output of the <code>blkls</code> command to filesystem data units, also known as a group of sectors.
<code>blkcat</code>	This reads the contents of a data unit.
<code>blkls</code>	This lists details of data units.
<code>blkstat</code>	This specifies the allocation status of a data unit.
<code>ffind</code>	This maps metadata to filenames.
<code>fls</code>	This lists files and directories in a given filesystem.
<code>fsstat</code>	This displays details of a filesystem, with a range of inodes.
<code>hfind</code>	This finds hash values in a database.
<code>icat</code>	
<code>icat</code>	This reads the contents of a file from its metadata address.
<code>ifind</code>	This maps between metadata and data units.
<code>ils</code>	This lists details on a range of metadata structures.
<code>img_cat</code>	This shows the contents of an image file.
<code>img_stat</code>	This displays details of an image file.
<code>istat</code>	This lists details of a specified metadata structure.
<code>jcat</code>	This displays the contents of a journal, for a filesystem like ext3.

jls	This lists records in a filesystem journal.
mactime	This creates a timeline of file activity.
mmcatt	This lists contents of a partition in a logical volume.
mmls	This lists partitions and volumes on a drive.
mmstat	This displays the disk label of a system.
sigfind	This identifies binary signatures.
sorter	This sorts files based on type.

Helix, which is owned by e-fense, currently comes in three different flavors: Pro, Enterprise, and Live Response. The Live Response version comes on a USB stick that can be booted directly on the system under investigation.

SANS Investigative Forensics Toolkit

The SysAdmin Audit Network Security (SANS) Institute is an organization dedicated to training security professionals. The SANS Investigative Forensics Toolkit (SIFT) is a Linux distribution developed by SANS for the purposes of forensic investigations. SIFT comes as a VMware appliance image. That means it cannot be used to live boot on a compromised system. However, isolating an investigation into a virtual machine can help to protect against further compromise on the network as you perform your investigation.

Digital Evidence and Forensics Toolkit

Another option is the Digital Evidence and Forensics Toolkit (DEFT). DEFT is not built on top of any other distribution in particular. Many distributions use GNOME as their desktop environment, but DEFT uses LXDE. DEFT also includes the WINE program, which allows for the execution of Windows programs from inside Linux. This enables forensics investigators to use programs that are designed for Windows and do not otherwise run under Linux. This can be a big advantage because there are a number of tools that are designed exclusively for Windows.

Build Your Own Media

You can certainly build your own media. Using your own particular configuration gives you control over what is included. However, it also means you have to pay attention to all the underlying packages when it comes to keeping up to date with the latest security fixes. You do not want your investigation tool to be compromised. When compiled from source code, you can set up commands on read-only media such as CDs or DVDs with all the libraries required to run those commands reliably. [Table 14-3](#) lists some common commands you will likely need.



NOTE

When downloading packages, many of the associated servers do not organize them by date. The latest version of a package may be somewhere in the middle of the listing.

Detailed instructions for building and installing the media should be available in a file named INSTALL in the directory of unpacked source code. Some customization may be required. For example, it's helpful to configure the source code to redirect tools to a specific directory with a command similar to the following:

```
# ./configure --prefix=/tools
```

After the given tools are written to a CD or DVD, you can mount the media on suspect systems, secure in the knowledge that the commands written to such media will not be changed.

TABLE 14-3 Commands to include in a forensic toolkit.

COMMAND DESCRIPTION	
cat	The <code>cat</code> command displays the contents of a file or concatenates multiple files. It's available from the coreutils package. For more information, see http://www.gnu.org/software/coreutils/ .
date	The <code>date</code> command displays the current date and can be used to modify the system date and time. It's available from the coreutils package.
dd	The <code>dd</code> command is a disk dump utility, used to copy disks and other media. It's available from the coreutils package.
dmesg	The <code>dmesg</code> command displays kernel messages. It's available from the util-linux-ng package, which can be downloaded from the Linux kernel servers at ftp://ftp.kernel.org/pub/linux/utils/util-linux-ng/ .
md5sum	This command is used to generate an MD5 cryptographic hash of any potential evidence.
nc	The <code>nc</code> command is used to open network connections, providing the user with a way to directly interact with listening services. Although it's available from a variety of sources, one option is coupled with the <code>nmap</code> command, available online from http://nmap.org/download.html .



WARNING

Most live CD/DVD distributions include command-line tools that can help with forensic analysis. But be careful. Standard live CD/DVD distributions may use or otherwise overwrite information on a swap partition. When booting some forensic live CD/DVD distributions, you may have to explicitly choose an option to avoid using the swap partition. Many of these distributions include installation options that support a dedicated forensic workstation.

Forensic Live Media

Many groups have created their own live media for forensic data recovery. Although very few of these options support recovery of volatile data such as files in the `/proc` directory, most provide excellent support for the investigator who wants to create a copy of suspect hard drives and analyze the data on those drives in detail.

Several options for forensic live media are listed in [Table 14-4](#). Most of these options can boot into a customized version of Linux, loaded in RAM, with tools that can diagnose and otherwise examine files associated with a Linux or Microsoft system on a local drive. In general, that drive is not mounted when the live CD or DVD is loaded.

TABLE 14-4 Options for forensic live media.

NAME	DESCRIPTION
------	-------------

Kali	This bootable live DVD distribution can be started without disturbing a swap partition, available from http://www.kali.org/ .
CAINE	This software, discussed earlier in this chapter, is available from http://www.caine-live.net/ .
Chaox	Based on Arch Linux, Chaox focuses on penetration testing and forensic tools. For more information, see http://chaox.wordpress.com/ .
DEFT	Built on Ubuntu Linux, DEFT includes a number of live tools for recovering data from live Microsoft operating systems. It's available from http://www.deftlinux.net/ .
FCCU Linux	This distribution, built in Belgium, is based on Debian Linux. It's available from http://www.lnx4n6.be/ .
Network Security Toolkit (NST)	Built on Fedora Linux, NST provides a number of standard network security applications on its live media. It also includes an Internet Protocol Security (IPSec) mode for secure communications. You can acquire a copy at http://www.networksecuritytoolkit.org/ .
SIFT	This Ubuntu-based Linux distribution was developed by SANS for the purposes of forensic investigation. You can get a copy at http://digital-forensics.sans.org/community/downloads .

When You Put Your Plan into Action

An attacker can take a number of paths to compromise your system. Regardless of which path the attacker takes, you will want to chase down the source of the breach or incident. This will be necessary to improve your operations and to prevent further incidents.

Of course, if there are files that should be found only on secure systems, there may be a different sort of security breach. Using a baseline image will help you more quickly get back online after you have gathered the necessary information. Although you may be able to clean the system and get it back online, it may simply be easier and quicker—not to mention safer—to start from a clean baseline, add the necessary services, and restore your data from a known good backup.

Confirming the Breach

To confirm a breach, you must know something about what happened. Has sensitive data been lost? If so, that by itself does not confirm a configuration issue. It may instead be due to lax physical security. Has a user account been compromised? You can get a good sense of that from the commands that account for logins, as described earlier in this chapter. Can you identify malware on a system? You should have an antivirus software package available with up-to-date signatures that you can use to scan your disk image. You can also use the malfind plug-in in Volatility to look for markers in memory suggesting that you have malware.

Until you can identify the specific breach and understand what needs to change on a system, you will want to make sure you have additional protections in place if you bring up a new system while you are performing your investigation. You must ensure your business is not affected, but you must also ensure that your other systems are protected and that an additional compromise does not occur while you are trying to isolate the point of entry.

Identifying Compromised Systems

After you confirm a breach, the next step is to identify any additional compromised systems. Sometimes it's easy to identify a compromised system. You run an antivirus scanner on a system and it identifies malware. You see log files where someone has logged into a service account. You see activity on an e-mail server that far exceeds the demands placed on it by local users. Such events are symptoms of a compromised system.

The reason behind a compromised system may be straightforward. It could be that the problem you have identified matches one addressed by a security update. But administrators do not always install every security update. Sometimes, they avoid such installations for excellent reasons. For example, kernel updates may affect support associated with third-party software. Related updates may require additional work on software that may be compiled during the installation process. The documentation behind some security updates may suggest that the security issue does not apply to systems on your network.

Although there are good reasons to avoid installing a security update, you must also be aware of the risks. If the subject system is compromised, the reason may be the uninstalled security update.

Having Replacement Systems in Place

Unless the problem is with the associated service, you should be able to quickly put a replacement system in place. This could be a warm or cold spare or it could be a quickly assembled system built from your baseline standard.

If you have determined that the problem is an uninstalled security update, have a system in place with that update. Make sure that system has been documented with a tool such as Tripwire or AIDE.

Of course, if the problem with a service has not been determined, users on the network may need to live without the service for a while. If it's a mission-critical service, few people will be happy with you. Users will be more willing to wait if you are able to set up the service in question with reduced functionality to help address the security issues.

Secure Backup and Recovery Tools

You use backup and recovery tools to make copies or images of suspect media, normally hard drives. The standard Linux tool for this is the **dd**, or disk dump, command. It copies every bit of a storage volume in a format that can be copied to an image. The alternative is the **rsync** command, which is well suited to copying files, preserving all relevant data such as links and last access times.

Disk Images for Later Investigation

Although the **dd** command is still the standard, other excellent commands are available for disk imaging. One of the benefits of the **dd** command is that it copies everything, even the universally unique identifier (UUID) number associated with the drive or volume. To that end, the simplest invocation of **dd** copies the contents of one hard drive to a second hard drive as follows:

```
# dd if=/dev/sda of=/dev/sdb
```

Breaking down this command, it takes the hard drive represented by the `/dev/sda` device as the input file (**if**) and copies the content, block by block, to the output file (**of**), represented by the `/dev/sdb` device file.

You can also use the **dd** command to create disk images. For example, the following command saves the contents of that first hard drive, represented by `/dev/sda`, to the noted file, in a directory named `/remote`, which could be mounted from a remote, portable, or shared network directory.

```
# dd if=/dev/sda of=/remote/troubleddisk1.img
```

Sometimes, disks on compromised systems may be troubled. In fact, an attacker may have caused such problems to make copying and investigating more difficult. To that end, options could be added to the command shown here to continue the backup process in case of errors (**noerror**); to disallow truncating (**notrunc**), which could lose important data; and to support the noted block size (**bs=16384**) in bytes, which is more likely to complete the duplication process.

If you see errors during the disk-duplication process, the **dd_rescue** command may be helpful. It reads data from back to front on a specified partition, volume or drive, and is more error-tolerant than **dd**. At the command line, it works as a drop-in replacement for **dd**. However, it's more likely to save data from a crashed filesystem, possibly even from an overused or otherwise damaged hard drive. If a regular **dd_rescue** command does not work, a **dd_rescue -r** command copies data from the end of the specified **if** volume first. One drawback is that **dd_rescue** can take several hours on a typical multi-gigabyte hard drive.



NOTE

The **dd** command copies everything on a drive or partition, including free space. Therefore, if you are using the **dd** command to make a disk image of a 100 GB drive, it'll require a 100 GB file.



TIP

The **dd_rescue** command is especially useful if recovering data from a drive that has not been backed up, such as from a portable system.

For a forensic investigation, the **dcfldd** command is even better. It performs all the functions of **dd** with all the parameters previously mentioned but also generates a cryptographic hash of the image it creates. This cryptographic hash is essential if you want to use the image as evidence. An example of the use of **dcfldd**, based on the preceding example of **dd**, is as follows:

```
# dd if=/dev/sda of=/remote/troubledisk1.img
hash=sha1
```

The **rsync** Command

For pure backups, the **rsync** command may make more sense. Unlike the **dd** command, it does not include the free space in a disk or partition in the backup. It just copies the actual files to the remote system. In addition, the **rsync** command is efficient. The second time it's used to copy a filesystem, all it transmits is anything that changed. That makes the **rsync** command well suited to backing up data over a network.

One problem originally associated with the **rsync** command is that it transmits data in cleartext, using the Remote Shell (RSH) protocol. With the help of the **-e ssh** switch, **rsync** data is encrypted and transmitted over an SSH connection in encrypted format. In fact, one version of the **rsync** command does even more. For example, the following command backs up user home directories to the **/backup/** directory on the noted remote system:

```
# rsync -aHz -e ssh /home/ bkadmin@bkserver:/backup/
```

This command prompts for either the password or the passphrase for the **bkadmin** user, in archive (**-a**) mode, copying hard-linked (**-H**) files, in compressed (**-z**) format. Archive mode incorporates a number of options; see [Table 14-5](#) for a list of **rsync** command options.

Mount Encrypted Filesystems

To work with encrypted filesystems, you will need live CD/DVD media that can handle such filesystems. Perhaps the most complex case is encryption on a logical volume.

If you are working with an Ubuntu system, you will have to add a couple of packages to the live CD operating system after it's booted. For Ubuntu systems, the key packages are **lvm2** and **cryptsetup**. For RHEL systems, you

can work with a Fedora system loaded from a live CD. You will also need to load a couple of packages, and the package names are essentially the same: lvm2 and cryptsetup-luks.

You can then load the cryptographic module, which is part of the cryptsetup* package. You should then be able to run a **cryptsetup** command similar to the following:

```
# cryptsetup luksOpen /dev/device path
```

If you have selected the correct device, the command should prompt for the password or passphrase required to decrypt the volume.

TABLE 14-5 Options for the **rsync** command.

OPTION	DESCRIPTION
-a	Archive mode, equivalent to -rlptgoD
-D	Device files
-e	Transfer over a remote shell option such as RSH or SSH
-g	Group ownership
-H	Hard-linked files
-l	Soft-linked files
-o	User ownership
-p	Permissions
-r	Recursive mode
-z	Compressed data transfer

The Right Way to Save Compromised Data as Evidence

Earlier in this chapter, you saw how to save dynamic data from a compromised system. Here, you will review that data. When appropriate dynamic data is saved, you will want to make a copy of the compromised hard disks. You can then analyze the copied volumes, secure in the knowledge that if there is a mistake, you can repeat the process. Of course, you will want to be sure to document everything, including checksums of any copies that have been created.

Basic Principles for Evidence

The information in this section is not intended to be used as legal guidance. This is not a comprehensive list of requirements when computer data has to be saved and analyzed as evidence. In any case, actual requirements vary by jurisdiction. In general, however, when saving compromised data as evidence, you must have excellent records. You must document where the evidence came from, how that evidence was duplicated, and the methods used to analyze that evidence, and prove that the duplicate is identical to the original. A matching cryptographic hash will be enough to prove the duplicate is identical. That is all part of the chain of custody.

With a good chain of custody, you can demonstrate that evidence has not been tampered with. Additionally, if it is always in the hands of a trusted investigator or administrator, it is safe. An attacker will not have a chance

to tamper with or perhaps erase that incriminating data. To that end, an appropriate chain of custody meets the following requirements:

- Storage is tamper-proof.
- Locations are documented.
- Receipts are given when the responsibility for data is transferred from one person to another.
- A list is kept of all known passwords and checksums.

Remembering the Volatile Data

With the chain of custody in mind, you are ready to use the techniques described earlier in this chapter to save dynamic data on a suspect system. To recap, you want to save any data that might otherwise disappear when a computer is powered down. At minimum, that means all data in volatile memory, which is generally limited to RAM. If you have forensic tools that can also save data from the swap partition, by all means do so. Otherwise, you can start many of the live CD forensic tools without using the swap partition, and can therefore save that data after the computer and/or hard drive media are powered down. In the spirit of the chain of custody, you should also document every step taken to save that data.

Preserving the Hard Disks

One approach is to boot the compromised system and a diagnostic system in a simple network. You should then be able to use the network connection to copy the volumes of the troubled system. Alternatively, you could use a live CD to copy the data from a hard disk directly to an image file. Techniques described earlier involved the use of the `dd` and `dd_rescue` commands.

After you have saved the image file, you can mount it with the loopback device. For example, if the image of the hard disk is saved as the `evidence.img` file, you can mount that file on an empty `/test/` directory with the following command:

```
# mount -o loop evidence.img /test/
```

Alternatively, there are methods available to set up such image files in virtual machines. When such a machine is configured without an external network connection, it's another way to record and/or test the data on such systems in isolation.



TIP

Do not use the original hard disk of the compromised system in any analysis. Commands like `dd` and `dd_rescue` are designed to create a bit-by-bit duplicate of an original disk. Physical write blockers can protect the integrity of the hard disk as evidence. One source for a forensic toolkit is the SANS Institute.

Disaster Recovery from a Security Breach

The response to a security breach depends on what happened. In general, you want to have gold baseline systems ready to go in case of a security issue. When appropriately configured, they can be put in place of compromised systems. Ideally, before you put these into place, you should determine whether there is a flaw in that gold baseline system.

If data is needed from compromised systems, you must know what has been compromised. During forensic analysis of a compromised system, you should be able to review the differences with the original configuration with tools such as AIDE and Tripwire. Reviewing the differences should help you identify what went wrong. It should identify files that have been changed or added by an attacker.

Determining What Happened

An integrity checker such as AIDE or Tripwire can give you a list of files that have changed. It should also tell you about files with different owners. Any suspect files should be investigated further with tools such as rootkit hunters and malware checkers. If you suspect a system has been compromised—especially if you think it has been compromised with malware—you should not trust the binaries on that system. Use trusted tools on an externally mounted drive, like a USB stick. In addition, use the logs associated with mandatory access control systems, specifically Security Enhanced Linux (SELinux) and Application Armor (AppArmor). If the attacker has made a mistake, perhaps by trying to change a file in a protected directory, it should show up in related log files.

If you have created a baseline of processes, any suspect process may stick out like a sore thumb. A dynamic list of processes shown with the `top` command should include those systems that are taking up an unexpected amount of CPU or RAM. That process can then be further investigated with the `lsof`, `strace`, and `ldd` commands, which track such processes to other files. You may also be able to identify compromised services from the output. If that process leads to a suspicious script, you may have hit pay dirt. Look at that script. It may identify the way the attacker is affecting your system.

After you have identified suspect files, you can try removing them. But that is not enough.

Prevention

After you have determined the problem, the next step is prevention. What can you do to keep the attacker from doing the same thing to the next system? First, you should investigate how the attacker got access to your systems. Then you can take steps to prevent the access from occurring again.

If the attacker gained access through the network, check the services associated with open ports. If it was through a compromised user account, make sure passwords are appropriate. If it was through a compromised service, make sure all security updates are installed. If that is not enough, you may need to find an alternative service. In the world of open source, there are alternatives for almost everything.

It's also possible that the attacker got direct access, perhaps by booting a live CD on a compromised system. The steps required to identify that breach are more in the realm of physical security professionals, who may need to review surveillance tapes.

Replacement

After you have taken preventative measures, you are ready to replace a compromised system on a permanent basis. If the problem is based on a physical security breach, a compromised password, or something else beyond the control of an administrator, your work has been validated. A replacement based on a current baseline ensures that any malware that has infected the system can no longer affect other systems. However, if a problem has been identified with the configuration, more work is required. The baseline may not be secure until the configuration is changed. You may need to update service software or even install a substitute service before putting the replacement into production.

How and When to Share with the Open Source Community

In the open source community, security depends on users who share. However, if you are the victim of an attacker, it may be difficult to admit that there was a problem. For example, if you did not install a known security update, the situation might be embarrassing. Or it may be the case that managers hesitate to share incident reports with the community. Nevertheless, open source software depends on participation from the community. If nobody shares their problems in the open source community, open source developers will not know where their software needs improvement.



WARNING

Any release of organizational information on a public forum should be approved by appropriate management and legal counsel.

If the Security Issue Is Known...

Sometimes, administrators avoid security updates for good reasons. If that reason relates to requirements for third-party software, the developers behind that third-party software must know about the problem. If the third-party software is supported, then the developers of that third-party software should be interested in the problem, especially if there is a support contract involved.

Even without a support contract, there is an ethos in the open source community. If software released by a group of open source developers is known to cause harm, those developers should get to work on a solution. In a great majority of cases, once the problem is confirmed, that is exactly what those open source developers will do.

If the Security Issue Has Not Been Reported...

Many open source development teams provide confidential means to report security issues. This is because these developers want a chance to evaluate the report before the information becomes public. Although that seems to work against the philosophy behind open source software, it's sometimes most appropriate. If the security issue is serious, you may wind up with a race between developers who work to quickly plug the security hole and attackers who want to take advantage.

Not all issues are security related, however. For example, the false positive given by older versions of the `last -i` command is just that: a false positive. Such issues have likely been reported through appropriate bug-collection databases, such as Red Hat's Bugzilla at <https://bugzilla.redhat.com/> or Canonical's Launchpad at <https://bugs.launchpad.net/>. If you have reasonable doubts about the issue at hand, search these databases first.

Like you, Linux security professionals are busy. If you report a security issue, developers need some information about your systems. Although the details depend on the suspect service, it's important to provide sufficient information to allow developers to replicate the problem on their systems.

Best Practices: Security Breach Detection and Response

You must plan for a security breach. It's best if you have a gold baseline system ready to go to take the place of at least any critical system on the local network. Of course, unless you know the cause of the breach, blindly replacing a compromised system could prove foolhardy. You must know what happened. If there is a flaw with the configuration of the system, you should update that gold baseline first.

To understand when there might be a problem, regular performance audits are appropriate. Systems that deviate significantly from baseline performance with respect to CPU and RAM usage may be suspect. Because you cannot spend all day on watch, the system status tool can help. If an issue arises, commands like `lsof`, `ldd`, and `strace` can help check suspect processes.

Good security requires the cooperation of users. It helps if security policies are kept as simple as possible. Appropriate policies with respect to users fall into the areas of physical access, Internet browsing, e-mail, and remote access. Even then, it helps to have educational materials available. Although Linux users are generally more aware about computers, they may not realize that Linux systems can be infected and can be carriers of malware for other operating systems.

Good security means you know who is logging into systems on the local network. Such information is available through the use of commands such as `who` and `w`, historically based on the `auth` and `authpriv` directives. In the associated log files, you can track access by administrative and remote users, along with administrative jobs.

As a security professional, you should monitor what is seen from user accounts. Downloaded packages and source code in user home directories may be signs of unauthorized changes from a regular user. Just as serious

may be the existence of executable files in user home directories. However, you may not want to prevent some users from setting up their own scripts.

To plan for a security breach, you need an incident response plan. When there is a suspected breach and the cause is not obvious, you may need policies that support a temporary increased level of vigilance. After a suspect system is identified, do not turn it off until you can copy volatile data such as the contents of RAM.

Part of the plan includes the right tools, including live Linux CDs with forensic commands. There are a number of options that will support access to compromised systems from their CD/DVD drives. After volatile data is recovered, various options are available to duplicate and diagnose issues on compromised hard drives.

When a Linux security issue arises, it's not always related to the configuration of a Linux system. It could be a physical security issue. If there is a confirmed problem, you will want to identify compromised systems and if possible have gold substitutes in place. You can use backup and recovery tools such as the **dd**, **dcfldd**, and **rsync** commands on compromised hard drives, as well as tools and commands associated with reading encrypted filesystems.

Problems with computer security may lead to legal issues. In general, you should document every step taken to save dynamic and static data. All data that is saved should be stored in secure locations, with appropriate passwords and checksums. After a problem is identified, it's time to replace compromised systems. If the cause is unrelated to the system configuration, you can address the problem and set up a replacement from the current gold baseline. Otherwise, you will need to update the baseline and possibly even set up a replacement service.



CHAPTER SUMMARY

In this chapter, you examined some of the tools and issues related to the detection of security breaches on a Linux system. If you have identified a compromised Linux system, it's important to recover any volatile memory before making an image of the hard disks, especially if it's being done for evidence. It's important to document everything as you go through this process. Live CDs can help in several ways.

Security also depends on users. They are more likely to follow simple, sensible policies in the areas of physical, Internet, e-mail, and remote access security. With such policies and appropriate log files, you will be able to track user logins to help catch suspicious activity. With an appropriate response plan, you will have the tools to identify a troubled system in a way that allows you to replace it with a more secure system.



KEY CONCEPTS AND TERMS

Computer Aided Investigative Environment (CAINE)

dd

dd_rescue

Digital Evidence & Forensic Toolkit (DEFT)

l1dd

/proc/kcore

Red Hat Bugzilla

rsync

The Sleuth Kit (TSK)

strace

sysstat

w

who



CHAPTER 14 ASSESSMENT

1. Which of the following commands can display the free memory in RAM and in a swap partition? (Select two.)
 - A. **free**
 - B. **mem**
 - C. **top**
 - D. **swapon**
2. Which of the following commands is used to identify users who have since logged out?
 - A. **who**
 - B. **w**
 - C. **last**
 - D. **sar**
3. Which of the following file extensions is *not* associated with software packages?
 - A. **.odt**
 - B. **.tar.gz**
 - C. **.rpm**
 - D. **.deb**
4. Which of the following files is most likely to change when a system is powered down?
 - A. **/etc/mtab**
 - B. **/etc/fstab**
 - C. **/boot/grub/menu.lst**
 - D. **/etc/crontab**
5. Which of the following commands is least useful for recovering data from a live system?
 - A. **nc**
 - B. **vi**
 - C. **dmesg**
 - D. **cat**
6. What command can be used to duplicate the contents of a partition by its device file?
 - A. **dd**
 - B. **rsync**
 - C. **Copy**
 - D. **backup**
7. Which of the following commands does not pertain to compiling the source code associated with other commands?
 - A. **compile**
 - B. **configure**
 - C. **make install**
 - D. **make**
8. Which of the following commands does not include free space in the duplication process?
 - A. **rsync**
 - B. **dd**

- C. `dd_rescue`
 - D. `icat`
9. Which of the following steps is not appropriate when saving compromised data from a hard drive?
- A. Keeping a compromised system connected to a network during an investigation
 - B. Taking special care to avoid overwriting data in a swap partition
 - C. Booting a live Knoppix CD distribution
 - D. Powering down a compromised system after saving dynamic data
10. Which of the following steps should you take if you have identified a new security problem with open source software?
- A. Share the concern on a standard mailing list for the distribution.
 - B. Share the concern on a standard mailing list for the compromised software.
 - C. Communicate privately with the developers of the compromised software.
 - D. Nothing, as it's important to protect proprietary information in the open source community.

CHAPTER

15 Best Practices and Emerging Technologies

Y

OU MIGHT VIEW SECURITY as a battle between attackers who want to break into computers and security professionals like you who are trying to keep those attackers out. As attackers find new ways to break in, open source developers respond, finding ways to prevent such attacks. It's then up to you to keep systems up to date to protect your systems and the information stored on them. Updates may not always work, however. In some cases, attackers may break into your systems before updates are available. Some people call this type of vulnerability a *zero-day vulnerability*. In other cases, updates may cause problems for hardware. Some updates may even violate service agreements for third-party software. Updates can be especially problematic if they require tools that can also be used by attackers. It's important to test updates before putting them into effect on production systems.

Security also depends on users and administrators. This requires good policies for both groups, as well as compliance with such policies. Audits may be needed, not only to verify compliance with such policies but also to satisfy legal and regulatory requirements.

Security requires the help of everyone in the Linux community. It relies on the developers behind various Linux distributions. It depends on the developers behind major applications. It relies on the developers behind proprietary applications designed for use on Linux. It's even helped by the quality of communication within the Linux community. When you know how to take full advantage of these resources, you will have the best information available on how to secure Linux systems.

Chapter 15 Topics

This chapter covers the following topics and concepts:

- How to maintain a gold baseline
- How redundancy can help ensure availability
- Which varieties of support are available
- Why you should check compliance with security policies
- How to keep Linux up to date
- How to keep distribution-related applications up to date
- How best to manage third-party applications
- Why you should share solutions with the community
- Why you should test new components before using them
- How to keep up with security on your system

Chapter 15 Goals

When you complete this chapter, you will be able to:

- Identify best practices for keeping Linux software up to date
- Take advantage of corporate and community support

- Prepare for any changes that may occur

Maintaining a Gold Baseline

A gold baseline is secure. It stays secure because only copies of that baseline are used in production. Just like any other system, however, it must be kept up to date. By its very nature, the software in a gold baseline is focused on the core commands and libraries associated with a distribution, along with the Linux kernel. It may include critical services associated with all systems, such as the Secure Shell (SSH) and system log services. An attacker who breaks into any of these systems could more easily gain access to all systems associated with every Linux distribution.

Although the diversity of available Linux distributions does help promote overall security, that diversity does not extend to the core bits associated with the kernel or those basic commands and libraries used on every Linux system.

To maintain the gold baseline, you must monitor security news associated with selected distributions. When developers make updates available, it's up to you to make sure those updates work on local systems, conform to requirements associated with other software, and process that update on other systems. In addition, you must make sure the database associated with intrusion detection tools such as the Advanced Intrusion Detection Environment (AIDE) and Tripwire is up to date.

Monitoring Security Reports

The developers behind various Linux distributions maintain security reports for all supported packages on their distributions. In general, when a security report is released, an updated package should be available for installation through standard repositories.

In principle, you should address all security reports. You might choose to avoid installing a security update if it adversely affects a support contract, a critical software application, or important hardware. If you make that choice, the risks will continue until you install that update. If your choice relates to the support level associated with third-party software, you could ask for help from the vendor firm, but the vendor may say no.

If you choose not to install an update, be sure to keep track of the security report. If an affected system is compromised, examine what happened. If the compromise matches the information listed in the security report, you are going to have to reconsider the installation. If the issue relates to third-party software, you can go back to its developers or find an alternative application.

Working Through Updates

Not all updates are security related. Some address bugs that are not related to a security issue. Others may enhance features. Updates may affect individual software packages or they may address the Linux kernel.

If you are responsible for a group of Linux systems, it may be appropriate to set up a dedicated repository. The techniques of Linux patch management support the development of independent repositories, which can be based on a mirror of existing libraries of packages released by the developers of your preferred distribution. Even Red Hat Enterprise Linux (RHEL) supports mirrors of its repositories on remote networks.

If you choose a custom group of updates—say, only for security and bug fixes—you can organize that group of packages into a unique repository.

Recalibrating System Integrity

Every time there is an update, you will need to test the changes functionally. After you have verified the changes on a test system, you should update the gold baseline system. If the baseline works after the update, the next step is to recalibrate the current database created by the integrity-checking tool of your choice, such as AIDE or Tripwire. That database documents the current state of any baseline system that has been configured. You can

find deviations from that baseline in reports from AIDE or Tripwire, which document unexpected changes to critical files.

Ensuring Availability with Redundancy

When problems arise with a system, it's helpful to have a replacement available and ready for use. Having a redundant system available depends on the status of the gold baseline. How you maintain a gold baseline depends on its role. The steps required to configure redundant physical systems are different from those needed to maintain redundant virtual systems. For mission-critical services, it may also be helpful to have service-specific baseline systems available.



NOTE

If the system in question is running a service on which users depend, they will be thankful when you have a replacement ready to go.

A Gold Physical Baseline

In some ways, virtualization has revolutionized the concept of a gold baseline. Before virtualization, gold baselines meant stacks of hard drives, or at least hard drive images ready to be written to other physical devices. Not all systems should be virtualized, however. Gold physical baselines are important, too.

Putting multiple virtual systems on a single physical system can put virtual guests at risk if the physical system is compromised. Because you must connect the host operating system to the network to provide networking to virtual guests, you open the door to potential access. If you have access, even remote access, to the host operating system, it's just like getting physical access to a physical system. This means that to protect very sensitive systems, those systems may need to be physical rather than virtual.

To that end, a gold physical baseline system can still be stored as a virtual image. With the `dd` command, you can create image files from the hard drives of your choice. When new copies of the baseline are required, you can also use the `dd` command to duplicate that image to a new physical drive. For example, the following command writes the `golden.img` file to the second hard drive device on the system:

```
# dd if=golden.img of=/dev/sdb
```

A Gold Virtual Baseline Host

If you are taking advantage of virtual machines, gold baseline systems fall into two categories:

- **Virtual hosts**—Virtual hosts are physical systems, typically configured with multiple and/or multi-core CPUs.
- **Virtual guests**—Multiple virtual guests can run on each virtual host. Each virtual machine system is a guest.

To understand the requirements for a gold baseline virtual host, you need to get into the hardware requirements for different virtual guests.

Hardware Requirements for a Gold Virtual Host

The following is a brief overview of how you can measure the hardware requirements for a virtual host and all virtual guests on that host. This section addresses the basic requirements. For details, refer to the documentation for the virtualization technology of your choice. Despite the range of options, major Linux distributions are moving toward the Kernel-based Virtual Machine (KVM) system.

Conservatively, a gold virtual baseline includes sufficient random access memory (RAM), central processing unit (CPU) speed, and hard drive space for the maximum requirements of each component virtual machine. In practice, multiple virtual machines can share RAM and CPU. In other words, because properly configured systems do not normally use all their allocated RAM and CPU, you can effectively configure multiple virtual machines to overbook CPU and RAM.

However, KVM does have the following limits with respect to virtual hosts and guests:

- **Virtual host**—KVM can manage up to 256 CPU cores and 1 terabyte (TB) of physical RAM.
- **Virtual guest**—KVM can support virtual machines with up to 16 CPU cores and up to 256 gigabytes (GB) of RAM.

In other words, gold baseline virtual hosts may require a large number of CPUs and a lot of RAM. However, such requirements do not necessarily apply to local hard drive storage because you can configure the space required for each virtual guest on remote storage systems including network attached storage (NAS), direct attached storage (DAS), and storage area network (SAN) systems. **Network attached storage (NAS)** is a storage system connected to a network, normally using file-sharing protocols. **Direct attached storage (DAS)** is a storage system attached directly to one or more computer systems using cables. Unlike NAS, DAS systems do not use network protocols to share data. A **storage area network (SAN)** is a storage system that may consist of one or more NAS or DAS systems.

To understand overbooking, consider the following example. Suppose you want to configure 10 virtual machine guests that require 512 MB of RAM each. In addition, suppose the virtual host requires 1 GB of RAM. In the most heavily loaded situation, that system would require about 6 GB of RAM. If the virtual guests are properly configured, however, they will require the maximum amount of RAM only rarely. Therefore, it's quite possible that the noted configuration would work on a virtual host with only 4 GB of physical RAM. The hypervisor, also referred to as the virtual machine monitor, supports the sharing of available RAM and CPU resources by multiple operating systems. In the rare cases where every virtual machine is at maximum capacity, the virtual host would use configured swap space.

The actual allocation of RAM and other resources to different virtual hosts depends on the service. The behavior of such virtual hosts depends on the demands on that service. Therefore, the actual allocation of RAM and other resources to different virtual hosts varies widely.

Once configured, it may be helpful to have a redundant virtual host ready to go in case of hardware failure. After all, if that virtual host is running 10 or more virtual guests, it's an important part of your network!

A Gold Baseline Virtual Guest

As noted, virtual machines on KVM systems are not subject to severe limits. After a gold baseline virtual machine is configured, it's relatively easy to use. The data in virtual guests is contained in just a few files in dedicated directories. To use a gold baseline virtual guest, all you need to do is make a copy of the applicable directory. If that virtual guest has been configured with static networking, you will have to change the IP address, hostname, and other unique information so it does not conflict with other virtual guests, but that is it. All you need to do is install and configure the desired service, and it's ready to go.

One question is whether it's appropriate to keep several gold baseline virtual guests, such as for services that require extra disk space and for desktops. Fortunately, it's easy to adjust the number of CPUs allocated to a virtual guest in major virtual machine systems, including KVM.

Service-Specific Gold Baseline Systems

Some services are more mission critical than others. For example, Web sites associated with electronic commerce are more important than the File Transfer Protocol (FTP) server used to share electronic copies of user manuals. Therefore, you may want to configure some service-specific gold baseline systems. That way, you will be prepared in case a server goes down.

Some gold baseline systems should go beyond a basic operating system. Some services are mission critical to an organization. In such cases, it may be helpful to create a gold baseline, customized for that service.

If there is a security issue that is unrelated to the subject service, some updates may be required, but that should not affect the service. Any associated configuration file should still perform the intended functions. However, the careful administrator will still test the resulting configuration.

If the service was brought down due to a direct security issue, however, additional analysis may be required. For example, Ubuntu Security Notice 612-2 highlighted an issue with the way encryption keys were created from passphrases. Because those encryption keys were vulnerable, administrators of SSH servers had to update their SSH servers and then arrange to regenerate encryption keys for all users.

Identifying Your Support Options

The companies behind Linux distributions have found a way to make money: by selling their expertise. Organizations that purchase this type of expertise can expect some level of support with respect to updates, hardware certification, and certain core applications. Because these are the people who put together and tested these distributions, they are qualified to help you put their software to work. Given the open source nature of this software, others may have similar levels of expertise. In fact, there are other companies that sell similar levels of support.

Some corporate officers are opposed to using software developed by volunteers. Many of these managers recognize that free software requires people with the skills to use such software. If they have not hired such expertise, they can purchase it through corporate support. Canonical and Red Hat are not the only companies that provide support for their Linux distributions. For example, not only does Oracle create its own rebuild of RHEL, but it also sells its own support for its rebuild. Several hardware vendors such as Dell and HP sell servers with different Linux distributions and provide their own support for such. Of course, any number of consulting companies can also provide varying levels of support for your Linux systems.

Because support options are constantly evolving, this section simply summarizes the support options available from Red Hat and Canonical.

Red Hat Support Options

Red Hat support is based on subscriptions. Although reduced-price subscriptions are available—for example, for students and educators—they do not include official Red Hat support. Nevertheless, they do include direct access to downloadable Red Hat installation media along with product updates. All users with subscriptions have access to the Red Hat Network for Web-based administration of their systems.

Other levels of support, which vary, include the following features:

- **Response times**—Red Hat support response times range from one hour to two business days.
- **Hardware certification**—RHEL has been tested and is certified to work with various hardware components and on certain complete systems.
- **Independent software vendor (ISV) support**—RHEL has been tested with a variety of applications.
- **Support for open source server applications**—RHEL supports Apache, Samba, the Network File System (NFS), and more.

Red Hat support may also vary with respect to physical and virtual systems. In response to the litigation that surrounds some open source software, Red Hat includes an Open Source Assurance Program, which promotes the security of Linux systems from potential lawsuits. For more information, see <https://www.redhat.com/details/assurance/>.

Canonical Support Options

Canonical is the corporate sponsor of the Ubuntu distribution. As such, it's an organization with an interest in selling support contracts.

Canonical offers services for the desktop and the server, with a variety of support options. The company offers Landscape as a Web-based administrative tool. On the desktop, the services offered by Canonical include the following:

- **Windows network access**—This includes configuration of an Ubuntu system as a member of a Microsoft Active Directory.
- **Remote desktop configuration**—This supports the sharing of a desktop using remote desktop services.
- **Desktop virtualization**—This helps the user configure Ubuntu and Microsoft operating systems as virtual machines on an Ubuntu host system.

Canonical also offers professional services more directed at the server. Because the literature suggests that Canonical's support is more in the nature of consulting as opposed to a help desk, detailed investigation is left to the reader. For more information on Canonical support options, see <http://www.ubuntu.com/support/services/>.

Canonical encourages users to take advantage of the open source community. Such support is available to anyone who can present their issue and motivate regular users and developers to address that issue. Because this is not paid support, there is no guarantee or warranty associated with any answers the user receives.

Open Source Community Support

There are many skilled Linux users on message boards, mailing lists, chat rooms, and so on who seem to be always online, ready to help. Generally, they will be more motivated if they know you are addressing a problem in the “real world” rather than posing a hypothetical concern. However, because Linux is a hobby to most of these users, their time is limited.

If you treat users on the appropriate Linux forum with dignity and honor, there is no need to fear the occasional flame war that may arise. If you do your homework before posing a question, you are more likely to get a better and more complete response. Before going to the open source community with a question, the following steps are appropriate:

- **Check available documentation**—There is much excellent documentation is available online, including on the Web sites run by Ubuntu and Red Hat.
- **Review error messages**—Online sources are filled with tales of others who have encountered an encyclopedia of errors. It's quite possible that someone has already encountered the problems that you are having and has found a solution.
- **Include relevant hardware and software information**—The behavior of some applications may vary depending on whether they are run on a 32-bit or 64-bit system. Sometimes, the problem may be as simple as a missing software package.

If you have paid for corporate support, use it! However, the same principles apply when working with consultants and help desk technicians. If you have done your homework as described, it will show. And if support is paid for on an hourly basis, the time that is saved translates directly to money.

Checking Compliance with Security Policies

Although it's not necessary to run a full Open Source Security Testing Methodology Manual (OSSTMM) audit every day, it can be helpful to audit various activities on a regular basis. For example, you might regularly run password-cracking tools to verify how easy it would be gain access to the system if an attacker were able to use a brute-force attack. If such tools discover user passwords in a few minutes, those passwords are most likely too simple, and their associated accounts subject to easy access by attackers.

User Security

You should monitor the following areas of user behavior on a regular basis:

- **Physical access**—If users are limited by workstation, check appropriate logs. Are those users the only ones logging in through that workstation?
- **Internet access**—To regulate where users go online, you need software that checks such access. The open source solution for this is Squid. This may be subject to privacy limits, depending on those governmental and regulatory authorities that apply.
- **E-mail access**—As with Internet access, the inspection of user e-mails may be subject to privacy limits, per any laws and requirements of the jurisdiction of the employee and the server (which may differ).
- **Remote access**—Information on all access via logins should be collected on Linux from application and system logs.
- **Passwords**—You can use password-cracking tools to check the complexity of user passwords to highlight those users who may not be in compliance with organizational password complexity policies.

Administrator Security

Security requirements for administrators should go farther than those for regular users. Because administrators set examples for other users, the basic requirements just described for regular users should be applied more rigorously to administrators. Password complexity is especially important, not only for the regular accounts owned by administrative users but also for the root administrative account.

Regular accounts used by administrators are frequently enhanced with the ability to run the `sudo` command. As the importance of the root administrative account password is straightforward, so too are appropriate passwords for the regular account of each administrator. If an attacker can decrypt that password, he or she can use it with the `sudo` command to run administrative commands.

Administrators generally have access to more systems than ordinary users, including server rooms. Even if that access is just through the SSH service, more rigorous access requirements should reflect the importance of data on those servers:

- **Remote access**—Administrative access over the SSH service and subsequent access to root administrative accounts are normally recorded in log files. You should also track direct remote SSH logins to the root administrative account because such passwords could eventually be decrypted.
- **Physical access**—If physical administrator access to server rooms is also tracked, that can and should also be checked on a regular basis.

Keeping the Linux Operating System Up to Date

It's important to keep the Linux operating system up to date, not only for security purposes but also to address any bugs that might arise. The actions taken to address such updates can be classified into three categories, depending on whether they address baseline software, address functional bugs, or add new features. Baseline updates relate to the Linux kernel, along with standard commands and libraries. Updates that address software bugs may or may not affect security. If you or your users want new features, it may be appropriate to install new releases of updated software.

It's important to retain configuration control to make sure you know what is installed on Linux systems on the network. To that end, it's also important to keep related local software repositories up to date. If those repositories are not local, others may install software on their systems without your knowledge. Therefore, it's important to keep local repositories up to date with any new packages that you decide should be installed on local systems.

Because few enterprises look forward to installing new versions of a Linux distribution, the support levels associated with a distribution release are important. That explains some of the appeal of Ubuntu Long Term Support (LTS) Server Edition and RHEL distribution releases. Canonical has committed to providing security

updates for Ubuntu LTS server releases for at least five years. Red Hat has committed to providing similar updates for RHEL releases for at least seven years.

Baseline Updates

Updates of baseline packages are perhaps most important, as they apply to all Linux systems. Any change to baseline packages, especially those related to the Linux kernel, may have wider-ranging effects. For example, it may affect communication between the operating system and local hardware.

If there are third-party packages installed, updates of baseline packages may affect their support status. If such packages were built to the libraries associated with one version of the kernel, they may have to be built again. That process may be more difficult to automate.

Functional Bugs

Bug is another word for software that does not work as it should. Sometimes a bug is purely functional, such as a command that does not work as advertised. Sometimes a bug is related to security. Ideally, all updates that address bugs should be installed. Nevertheless, it's possible to prioritize such updates if you cannot install them all.

Updates that address bugs in third-party packages may be another matter. When released by the developers of a service or application, they may not have been fully tested by the developers of a distribution. Therefore, the risks associated with such updates may sometimes be greater.

New Releases

Many Linux distributions regularly deliver new releases. Ubuntu delivers a new release every six months, in April and October of each year. RHEL gets a new release less frequently.

Some version releases are easy. There are updates to the software packages that are mostly in line with the versions from the previous release if you were keeping up with updates. Other version changes are more significant, like the move from RHEL 6 to RHEL 7, in which there were substantial changes to how the system was managed.

New releases include updated software with new features in a number of areas, starting with a new kernel. Some feature updates to the kernel and most other applications are not possible unless developers start from a more advanced stock kernel. As a result, you may not see the same backports in a version update that you would see in a mere update within the version.

Remember, the kernel is the core of the operating system—the way applications communicate with system hardware. When the kernel goes through major changes, tests are always appropriate. Although developers do their best to retain compatibility with previous versions, especially with respect to hardware, such compatibility is not always assured.

In addition, any update to the kernel normally means changes to any packages compiled against the previous version of the kernel. Required updates may include the tools used to compile such packages.

If You Choose to Upgrade a Distribution

Upgrades to a newer release of a distribution do not necessarily mean a fresh installation. On Ubuntu systems, when the /etc/apt/sources.list file has been modified to point to a new release, an upgrade is then possible with the `apt-get dist-upgrade` command. On Red Hat systems, the installation program associated with the installation CD/DVD detects previous existing versions of a distribution and offers to upgrade that existing system.

Distribution upgrades are not tested as rigorously as new releases. Most beta testers work with new releases on freshly formatted drives. Therefore, although a distribution upgrade may save some existing configuration files, administrators who take that route should apply the same rigor as if they were testing a new installation.

In general, upgrades of more than one major version are not recommended without some understanding of the impact. For example, any third-party applications you may be using on top of Linux may no longer function if the upgrade contains significant changes. You can do the upgrade—although you should usually take intermediate steps—but you need to be aware of the potential for breakage. You may even end up with a system that will not boot properly.

Keeping Distribution-Related Applications Up to Date

Because the major Linux distributions include applications for both the server and the desktop environment, distribution-related applications fall into both categories.

An administrator who is careful about configuration control will limit choice. This can limit risk because it limits the number of potentially compromised software packages that may be installed on a system.

The application releases associated with some distributions may incorporate only security updates and bug fixes. If you or your users need to move to a different version of an application with more functionality, a new release of a distribution may be required.

Server Applications

Linux has made its name as a server operating system. As such, there are excellent groups of developers working on maintaining and improving the whole range of server services.

For example, you might use Samba to configure Linux as a domain controller on a Microsoft Active Directory network. Administrators who adapt such updates may save licensing costs and at the same time make it more difficult for attackers to penetrate such systems. You run the risk of interoperability problems, however. Although you may save the licensing costs, you could end up with much higher support and maintenance costs.

Linux administrators have come to expect similar advances on other major servers, from Apache through Squid, from the Secure Shell through Kerberos. When such advances add features or enhance security, you will want to look at these new applications carefully. Features can expose a system to more vulnerabilities, and security may end up restricting usability. Everything is a trade-off.

Functional advances in server applications are typically tied to different releases of a Linux distribution. As such, if you want a new feature in a server, you will have to make a choice: Install a new distribution release or install and compile a new release of a service from source code. Both options add complexity to your configurations—and complexity makes it more difficult for you to keep track of all the security issues that may arise.

Installing a New Distribution Release

Hesitation is understandable when a new distribution is released. Installing such updates can be a lot of work. However, if the upgrade is of a bastion server with a single service, there is no requirement to install a new distribution release for other bastion servers. Nevertheless, if you install multiple releases of a distribution, that means you will have to monitor appropriate security lists for news on all installed releases. If you have set up a local set of repositories for one release of a distribution, that means you would have to set up additional repositories. The effort associated with installing a new distribution release, even on a limited basis, is not trivial.

Installing the New Release from Source Code

The developers behind open source services make their source code readily available, at least in tarball format. After all, that is a requirement of all basic open source licenses. You can compile and install such source code on most Linux systems. In fact, if you are not ready to upgrade a distribution, that may be the only way to install a new release of a service.

If you choose to compile and install such source code, you will likely forfeit any support available from the companies that support a distribution. In fact, you will have to take care to uninstall any older version of the

subject service because normal updates could easily overwrite key files for the newer version compiled from source code.

If you get the source code directly from the developers of a service, it will be solely your responsibility to monitor the service for security issues. Every time an update is required, you may need to download and recompile the service again. Additional work may be required if the process is tied to a certain version of the kernel.

In some cases, the developers behind a service provide binary packages in formats suitable for Red Hat and Ubuntu systems. You do not need to compile such packages. They are easy to install using standard `rpm` and `dpkg` commands. The packages do show up in databases of installed packages. You can even incorporate them into custom repositories. However, such packages face the same issues as compiled source code with respect to support from the companies behind a distribution and security updates.

Desktop Applications

Sometimes you or users on the network just need that new feature. The issues related to such updates are similar to those just described for server applications. To implement such updates, you must either install a newer release of a Linux distribution or download, compile, and install the new desktop application in question.

The packages of the [OpenOffice.org](#) suite are a special case. Although [OpenOffice.org](#) developers make their packages available in binary format, they are still huge. Installations and updates of such packages can easily run into the hundreds of megabytes for every desktop system. Rather than having every desktop download from the Internet, a local repository or system-management suite can help make the process easier.

Managing Third-Party Applications

Third-party applications are not part of a distribution, for various reasons. Sometimes those issues are related to licensing; sometimes they are related to support. Although many developers behind third-party applications test their work with major distributions, there are rarely warranties with respect to truly open source software.

Licensing Issues

There are two sets of issues associated with licensing. First, not all Linux software is open source. Anyone who does not release source code or does not allow others to modify his or her source code cannot use open source licenses. Second, some controversial software packages may pose different sorts of licensing issues.

Non–Open Source Licenses

Open source software is frequently not the same as freeware. In other words, although freeware is provided to the public without cost, the code behind such software may still be proprietary. In contrast, the code behind open source software must be made freely available. Access to the source code via a public Web site or FTP server is an acceptable means to that end.

Open source licenses allow others to modify the source code and release it, as long as they give credit to the original developers. Although the DNS software released by Daniel J. Bernstein (`djbdns`) is a common alternative on Linux systems, it's not open source because Bernstein's license does not allow for its modification.

Software with Different Legal Issues

One reason the companies behind many Linux distributions do not license certain software components has to do with legalities. For example, many codecs that can be installed on Linux to read DVDs are encrypted with the **Content Scramble System (CSS)**. This is a digital rights management (DRM) scheme used to encrypt commercial DVDs. And although decryption libraries are commonly available for all Linux distributions, they are generally hosted by third parties on servers in nations where the enforcement of legal restrictions such as the **Digital Millennium Copyright Act (DMCA)**, a U.S. copyright law associated with intellectual property, is not as rigorous.

For these reasons, most companies that back Linux distributions do not include such packages on their repositories.

Support Issues

The support issues associated with a Linux distribution are best expressed as a function of the repositories associated with an Ubuntu release. Ubuntu includes universe and multiverse repositories for its releases. These repositories include packages that are not supported by Ubuntu. The difference between the two categories relates to licensing, with open source packages being grouped into the universe repository. In other words, you may be able to find support for packages in the universe repository from its developers.

If research is required, the `apt-cache show` command can help. For example, the following command provides basic information for the ABINIT scientific computing package:

```
$ apt-cache show abinit
```

In many cases, the output specifies the Web site for package developers along with an e-mail address for package maintainers. If such information is not readily available, it's reasonable to wonder whether the package is suitable for a secure environment.

RHEL supports all the packages in its repositories in one form or another. Many of the packages that may otherwise be found in Ubuntu universe and multiverse repositories can be found as part of the Fedora project, the Red Hat–based community Linux distribution, at <http://www.fedoraproject.org/>.

Sharing Problems and Solutions with the Community

Users who share information with the open source community are more likely to get a quick response when they run into problems of their own. Although there is no requirement to share information with developers or other users, the open source community is in many ways like a small-town neighborhood. The users who help their neighbors are more likely to get help in their own times of need.

Three open source communities of interest are developers, users and developers on mailing lists, and users on more generic open source support sites. With so many options, the first question is, where do you turn?

Which Community?

One issue with open source software is the difficulty in identifying a responsible party. Developers monitor mailing lists and boards for regular users, so that is where you should direct questions and information. Interested groups of developers and users include the following:

- **The group behind the application**—You might think the developers of an application would take responsibility for all problems with that application. However, because there are cases where application bugs are limited to one type of Linux distribution, application developers may not be the culprits.
- **The developers of the distribution**—When the company behind a distribution offers a support contract, it frequently includes applications. When such a contract is not available, users frequently rely on the communities associated with the major Linux distributions.
- **The group behind a graphical user interface (GUI) desktop environment**—More interested groups are the developers behind desktop environments such as the GNU Network Object Model Environment (GNOME) and the K Desktop Environment (KDE). Some applications are bundled with different GUIs.

It's certainly easiest if you can just discuss the issue with the developers of a distribution, especially through a support contract. However, not all users can afford that.

**WARNING**

Do not use developer mailing lists unless you are a developer making a direct contribution to a project.

Sharing with Developers

In general, the developers behind open source software want to hear about any problems and successes with their software. In many cases, however, they are overwhelmed with requests. In such cases, developers frequently review messages on mailing lists to get a sense of user response to their work.

In addition, the developers behind many applications maintain online bug-reporting tools. For Red Hat distributions, the Web site is <https://bugzilla.Red Hat.com/>. For Ubuntu distributions, the Web site is <https://bugs.launchpad.net/>. Also, the developers behind many applications and services have their own bug-reporting tools.

In general, developers suggest that you search their databases before reporting a bug. It's quite possible that someone else has encountered the same or a similar problem and has already reported it. If so, you will get a chance to add an e-mail address to the reporting list to help you monitor the status of the bug. Although the reporting systems vary, you will generally need the following information to report a bug:

- **Package name/version number**—To find the package name and version associated with a file, run the `rpm -qf` or `dpkg -S` command with the full path to the file.
- **Applicable hardware**—The behavior of a package may vary with different CPUs.
- **A descriptive summary**—Developers behind many distributions must deal with literally thousands of bug reports. An accurate, descriptive summary can help draw appropriate attention to your problem.
- **Steps to reproduce the bug**—To confirm the problem you are having, developers need to know what happened. If you can document the steps taken to produce the bug and you encounter the condition a second time, developers will know you are serious.
- **Applicable log files**—Error messages frequently appear in log files. It's appropriate to include short excerpts of the main body of a log report. Do not copy huge log files verbatim. If appropriate, you should include them as an uploaded attachment to the report.

Sometimes, the reported bug is just a misunderstanding. Do not be offended if a developer highlights your bug report as a duplicate of another. If it turns out you made a mistake by creating the bug report, it should be no big deal. However, developers and other users frequently notice repeated false bug reports from the same user. In other words, developers—and other users—are less likely to respond to users who continuously “cry wolf.”

Sharing on Mailing Lists

When sharing a problem on a mailing list, follow the same principles as when reporting a bug. Make sure you have done appropriate research. Create a descriptive summary appropriate for a mailing list title. Cite the steps required to reproduce the bug. As applicable, cite the package, version number, and associated hardware. Note, however, that mailing lists are generally not suited for very long log files. Some users may look at the inclusion of all but the most relevant excerpts of a log file as little better than spam.

Be aware, Linux mailing lists may still be a venue for flame wars. A flame war is generally just a heated debate, although it can sometimes wander into the area of abuse. Flame wars are in some ways a tradition that goes back to the first debates between Linus Torvalds and **Andrew Tanenbaum**, the developer of MINIX, on the nature of an appropriate kernel for any operating system. (MINIX is a Unix-like open-source operating system that predates Linux.) However, not all users manage flame wars as well as Torvalds and Tannebaum, so do not be afraid to walk away. Sometimes, that is the best way to preserve the goodwill of the Linux community.

Red Hat and Ubuntu sponsor mailing lists on a substantial number of topics. For more information, see <https://www.redhat.com/mailman/listinfo/> and <https://lists.ubuntu.com/>. They also sponsor a wide variety of message boards.

**NOTE**

The basic principles that apply to sharing information on mailing lists also apply to message boards.

Testing New Components Before Putting Them into Production

Open source developers do their best to test their software with every major combination of hardware and software, within reason. However, they cannot test every possible combination. Mistakes happen sometimes.

You should test every update before putting it into production, particularly on a mission-critical system. It can help others if you document the results. One way to jump ahead in the test cycle is to participate in the beta phase of software, before new features are officially released.

One axiom of testing is that you should always keep a version of the existing software configuration. If the testing fails or if a problem arises soon after you put the new or updated component into production, you will be able to restore the older working configuration.

Testing Updates

Especially in mission-critical applications and services, it's important to test new open source software before putting it into production. The same proviso applies if you are installing a new version of a software package.

With pre-production testing, it's important to make sure the service or application works as intended. You should test the system under simulated production conditions to confirm that the updated software can still handle the demands of your users.

If you have used the application or service before, you may have custom settings or configuration files from the previous version of the software. Although developers normally strive to retain backward compatibility, that does not always happen. Therefore, if you have custom configuration files for older versions of recently updated software, it's appropriate to test those configuration files. If the new version of the software functions at least as well with the older configuration file, you can install that upgrade on a production system with some confidence. You can then incorporate the new features associated with the upgrade at your convenience.

Documenting Results

Whatever happens with the update, document the results. That should help you improve the process associated with future updates. Perhaps more important, documentation of the update process can help future administrators work with systems on the local network.

FYI

If you need a new feature—for example, an easier way to create a chroot jail for an important service—a beta test period may already be too late to get such a change implemented in the next version of an application or a distribution. (A chroot jail is a way of fooling an application into believing that a small subset of the filesystem is the entire system. This prevents access to critical system files if the service becomes compromised.) If you are willing to contribute code to an important application, good developers will generally consider such contributions carefully. Although public participation in open source projects is encouraged—even prized—it's a time-consuming endeavor. However, if you have the time, your contributions will lead to benefits in terms of influencing the future direction of development of the service.

Beta Testing

Beta tests are opportunities for important users like you to influence the development of important software applications. If new features proposed by developers do not work, you get a chance to say so at a time when it's easier for developers to address the problem. If proposed new features make it less likely that you will use an application, a beta test is an excellent time to say so.

In general, you should never put beta versions of software into production. By definition, the testing of such software is not complete. However, if a new feature is needed, that is the opportunity for users like you to provide feedback to developers while they can still implement such changes in upcoming releases.

Keeping Up with Security on Your Systems

The best defense in Linux security—as with any system security, regardless of the operating system—is a layered defense. As such, developers are working toward improvements in every layer of that defense. Three areas in that regard are as follows:

- The development of a new firewall-management system, which is a successor to the `iptables` command
- Additional work on mandatory access control systems, both Security Enhanced Linux (SELinux) and Application Armor (AppArmor)
- Updates to penetration-testing tools, including the open source `nmap` command.

A New Firewall Command

The `iptables` command was introduced in 2001. Although it will help protect Linux systems years into the future, the rules associated with the `iptables` command have become rather complex. RHEL has implemented a new command, `firewalld`, which implements zones, allowing for the quicker implementation of a set of rules depending on where the system is. Other systems will likely implement `firewalld` going forward.



NOTE

Because SELinux and AppArmor do not work together and occupy a similar space, it's possible that one of these systems will become dominant on future releases of Linux distributions.

More Mandatory Access Controls

The two relevant mandatory access control systems are SELinux and AppArmor. Development on both is still quite active. Although both systems provide additional layers of security to a substantial number of services, they do not cover everything.

AppArmor

To paraphrase Mark Twain, rumors related to the demise of AppArmor are greatly exaggerated. AppArmor development did suffer when Novell laid off the developers working on that project. However, AppArmor is still the default mandatory access control system for Ubuntu. AppArmor development remains active in the Canonical system through Launchpad at <https://launchpad.net/apparmor/>.

At this moment, work on AppArmor is primarily based on creating more profiles to be distributed with AppArmor. Each service or application distributed with Ubuntu needs an AppArmor profile to better protect users and data. As you might imagine, with hundreds of software packages, creating all these profiles is a lot of work.

SELinux

The Linux kernel development team is responsible for keeping up with the tasks associated with improving SELinux support. They maintain a list of items that need to be addressed in the source code. Although they may sometimes add new features, at this moment, the list is focused primarily on making SELinux more efficient and offering tighter controls. They are primarily internal items and issues that users will not see. Previously, the SELinux to-do list was maintained by the U.S. National Security Agency (NSA). The current list can be found at <https://github.com/SELinuxProject/selinux/wiki/Kernel-Todo>.

Penetration-Testing Tools

The developers behind vulnerability-testing tools constantly update their databases of security issues. If you subscribe to one of these services and keep up to date with their updates, it will go a long way toward keeping your networks up to date with the latest security issues.

A number of other tools are used for penetration testing besides vulnerability scanners. One is the **nmap** command, which is a port scanner. Scripts associated with the **nmap** command are stored in the `/usr/share/nmap/scripts/` directory. Because new scripts are written frequently, you may want to download them from the associated subversion repository, which is different from standard package repositories. (**Subversion** is a system for version control used on many open source projects.) One method, as described by an nmap developer, David Fifield, is based on the following commands, which download the latest nmap information in the local `nmap/` subdirectory. The current list of scripts can be found in the `nmap/scripts/` subdirectory.

```
$ svn co --username guest --password "" svn://svn.insecure.org/nmap
```

If the **svn** command is not recognized, you will need to install the subversion package. If you prefer to use the latest nmap software instead of the version made available for your distribution release, you can run the `./configure` and `make` commands in the `nmap/` subdirectory. The noted command builds its own version of the **nmap** command in the `nmap/` subdirectory. The version available from the nmap subversion repository may be a beta, so full testing of that release may not be complete.

This may be a case where it makes sense to install a later version of a software package, even a beta, before it's made available for a distribution to help discover security flaws as soon as possible. However, the cautions described earlier in this chapter about third-party software do apply.

Once installed, you can run one of the 100 scripts listed at <http://nmap.org/nsedoc/>, assuming those scripts are available in the aforementioned `/usr/share/nmap/scripts/` directory. For example, the following command checks for recursion on the noted Domain Name Service (DNS) server:

```
# nmap --script=dns-recursion 192.168.100.1
```

Single Sign-On

One more important area for development in Linux security is single sign-on. It's hard enough for regular users to remember one username and password. When they are asked to remember multiple usernames and passwords and then change them frequently, users may revolt.

Linux developers are making progress toward single sign-on in a number of areas. Most straightforward are the drivers that are available for biometric controls. However, Linux also offers a variety of options to provide different levels of privileges. For example, Red Hat continues to develop configuration options associated with pluggable authentication modules (PAMs). Easier implementation of Kerberos can help administrators provide secure tickets to appropriate users.

In a heterogeneous environment of Linux and Microsoft operating systems, single sign-ons have required a common Lightweight Directory Access Protocol (LDAP) authentication database. It's possible for Linux systems to participate in a Microsoft Windows network. This may be as a client or it may be as a server. Making use of a centralized authentication server brings you closer to single sign-on and helps to better maintain password policies across all of your systems.

Incident Response

You should plan to be breached. This statement has two meanings. First, you should expect to be breached. Do not assume you are invulnerable to attack. Linux is not invulnerable. As with any other operating system, if you have an installation that is poorly maintained and managed, it's highly prone to attack. Second, although you should implement as many precautions as possible to protect against it, you should have a plan in place for when you are breached. The number of system breaches, even at companies that spend millions on system maintenance and security, is increasing.

To be ready when you become a target, you must have an incident response plan in place. This should include the formation of an incident response team composed of staff across the business. A serious data incident will affect more than just the IT staff, who have to get the system back online. You may have reporting requirements with which you must comply, so you will need a communications person. You also need someone to be a liaison with executive management. You need technical people to perform an investigation into the cause and you need people to restore systems to full functionality and implement any changes pertaining to the root cause of the breach.

You can and should keep a forensic live CD handy for cases when you do suffer a breach. Practice acquiring evidence from systems and make sure you have a kernel module ready to go that can be used to acquire volatile memory. You will also need a memory profile so you can perform an investigation on the volatile memory you have collected. Make sure anyone who handles these artifacts has been trained in proper evidence handling and that you maintain a chain of custody, documenting everything an artifact like a disk image has gone through and confirming that it has retained its integrity throughout the process.

Incident response is one of the most important aspects of system security. Make sure you have prepared yourself and your organization to respond to any incident that occurs. You should perform drills and walkthroughs and hold lessons-learned meetings so you are ready when you are breached. You may also investigate the potential of systems like Google Rapid Response to acquire data that may be used to develop a timeline and provide evidence in case of a breach. These tools will become more commonplace in the future.

Implementing more capability to gather and store application and system logs for the purposes of incident response will be vital. More movement in intrusion detection and prevention is becoming apparent. Some software packages are dropping off while others are starting development. More work in detection and intelligence gathering is essential.



CHAPTER SUMMARY

Best practices help you keep a network going, even through the disruptions associated with a security breach. Of course, it's helpful to keep such breaches to a minimum. To keep a network going, redundancy with gold baselines can support easy replacement of troubled systems. You must keep such baselines up to date. You can address specific problems through corporate and open source support systems. If you share solutions with the community, it's more likely to respond when you have a problem.

To make sure a network is secure, it's important to have regular security audits, including periodic checks of users and administrators between formal audits. Related challenges include updates of services and applications. More difficult challenges involve third-party software.

As for the future, with the firewalld feature from RHEL 7, Linux firewalls will be simpler and more flexible. Mandatory access control systems such as SELinux and AppArmor will cover more services and devices. Penetration-testing tools will include frequent updates to make sure the tools address the latest security issues. In the open source realm, the `nmap` command will add features through available scripts. You will also find more tools geared toward better documentation of what is happening on your systems so you are prepared when something bad happens.



KEY CONCEPTS AND TERMS

Andrew Tanenbaum

Content Scramble System (CSS)

Direct attached storage (DAS)

Digital Millennium Copyright Act (DMCA)

Network attached storage (NAS)

Storage area network (SAN)

Subversion



CHAPTER 15 ASSESSMENT

1. Which of the following should be considered for inclusion when developing a process for updating packages on your system?
 - A. MySQL
 - B. PostgreSQL
 - C. AIDE
 - D. LDAP
2. The total RAM requirements of virtual guests can exceed what is available from the virtual host.
 - A. True
 - B. False
3. Which command can you use to duplicate a partition by its device filename?
 - A. du
 - B. df
 - C. dd
 - D. db
4. Which of the following features of a virtual host supports resource sharing among virtual guests?
 - A. Multi-core CPUs
 - B. Virtual machine
 - C. Hypervisor
 - D. Networking
5. Which of the following companies provides specific support for the Linux desktop?
 - A. Canonical
 - B. Ubuntu
 - C. Red Hat
 - D. Apache
6. Which of the following distributions normally provides security updates for the longest period of time?
 - A. RHEL
 - B. Fedora
 - C. Ubuntu LTS
 - D. Kubuntu

7. Which of the following commands can provide more information about a package named gimp?
 - A. apt-cache show gimp
 - B. apt-cache info gimp
 - C. apt-cache search gimp
 - D. apt-cache help gimp
8. Which of the following commands can be used to identify the package that owns the /bin/ls file?
 - A. dpkg-qf
 - B. dpkg-i
 - C. dpkg-S
 - D. dpkg-V
9. If you wanted to implement mandatory access controls on your Linux system, which of the following would you install?
 - A. nmap
 - B. SELinux
 - C. Coreutils
 - D. firewalld
10. You have discovered a possible security concern in the Linux kernel. Who should you contact about this?
 - A. Red Hat
 - B. Canonical
 - C. Bugtraq
 - D. Kernel developers

APPENDIX**A Answer Key**

CHAPTER 1 Security Threats to Linux

- 1. D
- 2. B
- 3. C
- 4. A
- 5. D
- 6. C
- 7. GPL
- 8. Process
- 9. D
- 10. C
- 11. A
- 12. C

CHAPTER 2 Basic Components of Linux Security

- 1. C
- 2. B
- 3. A
- 4. C
- 5. D
- 6. A
- 7. C
- 8. B
- 9. A
- 10. C
- 11. C
- 12. D

CHAPTER 3 Starting Off: Getting Up and Running

- 1. C
- 2. B
- 3. A
- 4. C
- 5. D
- 6. A
- 7. C
- 8. B
- 9. A
- 10. C
- 11. C

12. D

CHAPTER 4 User Privileges and Permissions

1. B
2. B
3. D
4. `find / -group audio`
5. C
6. C and D
7. B
8. `visudo`
9. B
10. A
11. D
12. `/lib/security/`
13. B
14. A and B
15. C

CHAPTER 5 Filesystems, Volumes, and Encryption

1. B and D
2. C
3. A
4. A
5. `gpg --list-keys`
6. C
7. D
8. A
9. C
10. A
11. B
12. B
13. B
14. A
15. `getfacl test1`

CHAPTER 6 Securing Services

1. B
2. A
3. D
4. C
5. A
6. B
7. A
8. C
9. D
10. C

CHAPTER 7 Networks, Firewalls, and More

1. 0 to 1023

- 2. A
- 3. C
- 4. B
- 5. B
- 6. D
- 7. **--dport**
- 8. B
- 9. B
- 10. D
- 11. B
- 12. **sealert -b**
- 13. B

CHAPTER 8 Networked Filesystems and Remote Access

- 1. C
- 2. D
- 3. GSSAPI (acceptable: generic security services application program interface)
- 4. B
- 5. B
- 6. A
- 7. C
- 8. /home/donna/.ssh/ (acceptable: /home/donna/.ssh)
- 9. A
- 10. A
- 11. B
- 12. A and D
- 13. A
- 14. C
- 15. D

CHAPTER 9 Networked Application Security

- 1. D
- 2. B
- 3. C
- 4. B
- 5. A
- 6. B
- 7. D
- 8. A
- 9. B
- 10. A
- 11. C
- 12. A
- 13. D
- 14. B
- 15. C

CHAPTER 10 Kernel Security Risk Mitigation

- 1. C
- 2. D

- 3. B
- 4. <http://kernel.org/>
- 5. A
- 6. C
- 7. A
- 8. C
- 9. A
- 10. D
- 11. A and C
- 12. B
- 13. A

CHAPTER 11 Managing Security Alerts and Updates

- 1. D
- 2. A
- 3. A, B, and D
- 4. A
- 5. B
- 6. D
- 7. A
- 8. A
- 9. A
- 10. B
- 11. C
- 12. C

CHAPTER 12 Building and Maintaining a Security Baseline

- 1. D
- 2. C
- 3. **noexec**
- 4. B
- 5. A
- 6. C
- 7. B
- 8. B
- 9. **mail.=info**
- 10. A
- 11. B
- 12. **dpkg -l**
- 13. B
- 14. C
- 15. B

CHAPTER 13 Testing and Reporting

- 1. C
- 2. A
- 3. C
- 4. D
- 5. A
- 6. A

- 7. B
- 8. C
- 9. A
- 10. C
- 11. B
- 12. C

CHAPTER 14 Detecting and Responding to Security Breaches

- 1. A and C
- 2. C
- 3. A
- 4. A
- 5. B
- 6. A
- 7. A
- 8. A
- 9. A
- 10. C

CHAPTER 15 Best Practices and Emerging Technologies

- 1. C
- 2. A
- 3. C
- 4. C
- 5. A
- 6. A
- 7. A
- 8. C
- 9. B
- 10. D

APPENDIX**B Standard Acronyms**

ACD	automatic call distributor
AES	Advanced Encryption Standard
ALE	annual loss expectancy
ANSI	American National Standards Institute
AO	authorizing official
AP	access point
API	application programming interface
APT	advanced persistent threat
ARO	annual rate of occurrence
ATM	asynchronous transfer mode
AUP	acceptable use policy
AV	antivirus
B2B	business to business
B2C	business to consumer
BBB	Better Business Bureau
BC	business continuity
BCP	business continuity plan
BGP4	Border Gateway Protocol 4 for IPv4
BIA	business impact analysis
BYOD	Bring Your Own Device
C2C	consumer to consumer
CA	certificate authority
CAC	Common Access Card
CAN-SPAM	Controlling the Assault of Non-Solicited Pornography and Marketing Act
CAP	Certification and Accreditation Professional
CAUCE	Coalition Against Unsolicited Commercial Email
CBA	cost-benefit analysis
CBF	critical business function
CBK	common body of knowledge
CCC	CERT Coordination Center
CCNA	Cisco Certified Network Associate
CDR	call-detail recording
CERT	Computer Emergency Response Team

CFE	Certified Fraud Examiner
C-I-A	confidentiality, integrity, availability
CIPA	Children's Internet Protection Act
CIR	committed information rate
CIRT	computer incident response team
CISA	Certified Information Systems Auditor
CISM	Certified Information Security Manager
CISSP	Certified Information System Security Professional
CMIP	Common Management Information Protocol
CMMI	Capability Maturity Model Integration
CNA	computer network attack
CND	computer network defense
CNE	computer network exploitation
COPPA	Children's Online Privacy Protection Act
COS	class of service
CRC	cyclic redundancy check
CSA	Cloud Security Alliance
CSF	critical success factor
CSI	Computer Security Institute
CSP	cloud service provider
CTI	Computer Telephony Integration
CVE	Common Vulnerabilities and Exposures
DAC	discretionary access control
DBMS	database management system
DCS	distributed control system
DDoS	distributed denial of service
DEP	data execution prevention
DES	Data Encryption Standard
DHCPv6	Dynamic Host Configuration Protocol v6 for IPv6
DHS	Department of Homeland Security
DIA	Defense Intelligence Agency
DISA	direct inward system access
DMZ	demilitarized zone
DNS	Domain Name Service OR Domain Name System
DoD	Department of Defense
DoS	denial of service
DPI	deep packet inspection
DR	disaster recovery
DRP	disaster recovery plan
DSL	digital subscriber line
DSS	Digital Signature Standard

DSU	data service unit
EDI	Electronic Data Interchange
EIDE	Enhanced IDE
ELINT	electronic intelligence
EPHI	electronic protected health information
EULA	End-User License Agreement
FACTA	Fair and Accurate Credit Transactions Act
FAR	false acceptance rate
FCC	Federal Communications Commission
FDIC	Federal Deposit Insurance Corporation
FEP	front-end processor
FERPA	Family Educational Rights and Privacy Act
FIPS	Federal Information Processing Standard
FISMA	Federal Information Security Management Act
FRCP	Federal Rules of Civil Procedure
FRR	false rejection rate
FTC	Federal Trade Commission
FTP	File Transfer Protocol
GAAP	generally accepted accounting principles
GIAC	Global Information Assurance Certification
GigE	Gigabit Ethernet LAN
GLBA	Gramm-Leach-Bliley Act
HIDS	host-based intrusion detection system
HIPAA	Health Insurance Portability and Accountability Act
HIPS	host-based intrusion prevention system
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
HUMINT	human intelligence
IaaS	Infrastructure as a Service
IAB	Internet Activities Board
ICMP	Internet Control Message Protocol
IDEA	International Data Encryption Algorithm
IDPS	intrusion detection and prevention
IDS	intrusion detection system
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IGP	Interior Gateway Protocol
IMINT	imagery intelligence
InfoSec	information security
IP	intellectual property OR Internet Protocol

IPS	intrusion prevention system
IPSec	Internet Protocol Security
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
IS-IS	intermediate system-to-intermediate system
(ISC)²	International Information System Security Certification Consortium
ISO	International Organization for Standardization
ISP	Internet service provider
ISS	Internet security systems
ITIL	Information Technology Infrastructure Library
ITRC	Identity Theft Resource Center
IVR	interactive voice response
L2TP	Layer 2 Tunneling Protocol
LAN	local area network
MAC	mandatory access control
MAN	metropolitan area network
MAO	maximum acceptable outage
MASINT	measurement and signals intelligence
MD5	Message Digest 5
modem	modulator demodulator
MP-BGP	Multiprotocol Border Gateway Protocol for IPv6
MPLS	multiprotocol label switching
MSTI	Multiple spanning tree instance
MSTP	Multiple Spanning Tree Protocol
NAC	network access control
NAT	network address translation
NFIC	National Fraud Information Center
NIC	network interface card
NIDS	network intrusion detection system
NIPS	network intrusion prevention system
NIST	National Institute of Standards and Technology
NMS	network management system
NOC	network operations center
NSA	National Security Agency
NVD	national vulnerability database
OPSEC	operations security
OS	operating system
OSI	Open Systems Interconnection
OSINT	open source intelligence
OSPFv2	Open Shortest Path First v2 for IPv4
OSPFv3	Open Shortest Path First v3 for IPv6

PaaS	Platform as a Service
PBX	private branch exchange
PCI	Payment Card Industry
PCI DSS	Payment Card Industry Data Security Standard
PGP	Pretty Good Privacy
PII	personally identifiable information
PIN	personal identification number
PKI	public key infrastructure
PLC	programmable logic controller
POAM	plan of action and milestones
PoE	power over Ethernet
POS	point-of-sale
PPTP	Point-to-Point Tunneling Protocol
PSYOPs	psychological operations
RA	registration authority OR risk assessment
RAID	redundant array of independent disks
RAT	remote access Trojan OR remote access tool
RFC	Request for Comments
RIPng	Routing Information Protocol next generation for IPv6
RIPv2	Routing Information Protocol v2 for IPv4
ROI	return on investment
RPO	recovery point objective
RSA	Rivest, Shamir, and Adleman (algorithm)
RSTP	Rapid Spanning Tree Protocol
RTO	recovery time objective
SA	security association
SaaS	Software as a Service
SAN	storage area network
SANCP	Security Analyst Network Connection Profiler
SANS	SysAdmin, Audit, Network, Security
SAP	service access point
SCADA	supervisory control and data acquisition
SCSI	small computer system interface
SDSL	symmetric digital subscriber line
SET	secure electronic transaction
SGC	server-gated cryptography
SHA	secure hash algorithm
S-HTTP	secure HTTP
SIEM	Security Information and Event Management system
SIGINT	signals intelligence
SIP	Session Initiation Protocol

SLA	service level agreement
SLE	single loss expectancy
SMFA	specific management functional area
SNMP	Simple Network Management Protocol
SOX	Sarbanes-Oxley Act of 2002 (also Sarbox)
SPOF	single point of failure
SQL	Structured Query Language
SSA	Social Security Administration
SSCP	Systems Security Certified Practitioner
SSID	service set identifier (name assigned to a Wi-Fi network)
SSL	Secure Sockets Layer
SSL-VPN	Secure Sockets Layer virtual private network
SSO	single system sign-on
STP	shielded twisted pair OR Spanning Tree Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
TCSEC	Trusted Computer System Evaluation Criteria
TFA	two-factor authentication
TFTP	Trivial File Transfer Protocol
TGAR	trunk group access restriction
TNI	Trusted Network Interpretation
TPM	technology protection measure OR trusted platform module
UC	unified communications
UDP	User Datagram Protocol
UPS	uninterruptible power supply
USB	universal serial bus
UTP	unshielded twisted pair
VA	vulnerability assessment
VBAC	view-based access control
VLAN	virtual local area network
VoIP	Voice over Internet Protocol
VPN	virtual private network
W3C	World Wide Web Consortium
WAN	wide area network
WAP	wireless access point
WEP	Wired Equivalent Privacy
Wi-Fi	Wireless Fidelity
WLAN	wireless local area network
WNIC	wireless network interface card
WPA	Wi-Fi Protected Access
WPA2	Wi-Fi Protected Access 2
XML	Extensible Markup Language

XSS

cross-site scripting

Glossary of Key Terms

--frequent | An option of the `ck-history` command that lists users by the frequency of their recent logins.

--last | An option of the `ck-history` command that lists last-logged-in users by user, session, seat, and time.

A

Access control lists (ACLs) | In Linux, access control lists allow authorized users to set the permissions associated with a file or directory. Those permissions can supersede standard discretionary access controls.

Access vectors | Access permissions represented by a bitmap. Commonly stored for SELinux security in an AVC.

Access vector cache (AVC) | An access permission associated with SELinux.

account | A PAM module that can verify account validity based on expiration dates, time limits, or configuration files with restricted users. Also known as account management.

Advanced Encryption Standard (AES) | An encryption standard configured by the National Institute of Standards and Technology (NIST).

Advanced Intrusion Detection Environment (AIDE) | An intrusion detection system. Available in both Red Hat and Ubuntu repositories.

alert | A log priority that specifies problems that require immediate attention.

Anaconda | The Red Hat installation program.

Andrew Filesystem (AFS) | A distributed network file-system. Sometimes associated with Kerberos tokens.

Andrew Tanenbaum | The main developer behind the Minix distribution, an implementation of Unix designed to teach the principles of operating system design. Proponent of a micro kernel, in opposition to Torvalds' use of a modular kernel.

Apache | A foundation responsible for maintaining software including the Web server commonly called Apache or httpd as well as Tomcat, Catalina, and a wide variety of other software. Apache also maintains its own license.

apparmor_status | An AppArmor command that lists the configured profiles of various commands and services.

Application Armor (AppArmor) | A mandatory access control system used to create security profiles for different programs. Uses Linux security modules in the kernel. Not compatible with SELinux.

Application programming interface (API) | An interface that enables programmers to use functionality from outside their own program by specifying a way to call that external functionality from inside their program.

Application server | Software that provides a framework or container for an application to run. The application server manages HTTP connections, passing all the user-related data into the contained application.

aquota.group | The group quota management file.

aquota.user | The user quota management file.

Asterisk | An open source private branch exchange (PBX) telephony service that allows for Voice over IP (VoIP) calling.

at | A service for running administrative jobs on a one-time basis. Protected by the /etc/at.allow and /etc/at.deny files.

Attack canvas | The overall set of threats and vulnerabilities to a system or network.

Attack surface | Another way of saying *attack canvas*.

auth | A PAM module that can verify passwords, group memberships, and even Kerberos tickets. Also known as authentication management.

Authentication Header (AH) protocol | A protocol that guarantees integrity and data-origin authentication of network packets.

Availability | A C-I-A triad concept in which users have access to their data when they want it. When you can reach a system, it is available. Availability is a way of measuring whether you can reach systems and services.

AVG Technologies | A company that creates anti-malware systems for Linux and other operating systems.

B

Backport | A feature from a future version of a software component that is incorporated into an update of an earlier release. Backports are commonly applied to kernels with a certain version number to incorporate additional features without overriding any sort of certification of third-party software related to that kernel.

Backup Domain Controller (BDC) | A backup for a PDC on a Microsoft Windows NT domain. Linux with Samba can be configured as a BDC.

Base64 | A way of converting non-printable characters into a string that can be transmitted using a text-based protocol like HTTP.

Bastion host | A system with every unnecessary piece of software and service removed so it can be used for a specific purpose such as a security gateway between networks with differing protection levels.

Berkeley Internet Name Domain (BIND) | The most common DNS server on the Internet, originally created at the University of California at Berkeley, maintained by the Internet Systems Consortium.

Berkeley Standard Distribution (BSD) | An implementation of Unix, developed at the University of California at Berkeley. BSD uses a different licensing model than Linux does.

Binary kernel | When the source code of a kernel is compiled in an installable package, it is changed from a human-readable format to a binary format readable only by a computer.

Black-hat hacker | A malicious user who is engaged in activities that are likely illegal.

Blacklists | Lists of behaviors you want to prevent. A blacklist can also be a list of systems you don't want to access your system.

Bootstrap Protocol (BOOTP) | A protocol of the TCP/IP suite associated with automatic assignment of IP addresses. May also be used for the automatic acquisition of IP addresses from a DHCP server on a remote network. Associated with UDP port 68.

Bourne Again Shell (Bash) | A command interpreter used to issue text commands.

Brute force attack | A type of attack that uses every possible method without any thought as to the best approach or using any shortcuts to find your way in.

Bugzilla | A Web-based bug-tracking and management tool commonly used on open source projects from Red Hat to the GNOME desktop environment.

C

Cache poisoning | A frequently malicious insertion of non-authoritative DNS data as if it were authoritative. May be used by malicious users to redirect users to malicious Web sites.

CentOS | Short for the Community Enterprise Operating System, CentOS is sometimes known as a rebuild. This is because it is a distribution built by third parties, based on source code released for the Red Hat Enterprise Linux distribution.

Certificate authority (CA) | An entity such as Symantec or GoDaddy that issues digital certificates for use by other parties. Secure Web sites without an official CA return an error message.

chage | A command that can modify password-related information for a user, such as the password's expiration date.

Challenge-Handshake Authentication Protocol (CHAP) | A protocol for validating users before allowing access, which includes a challenge to verify the identity of a user.

Challenge-Response Authentication Mechanism (CRAM) | A group of protocols where a service such as open source sendmail presents a challenge such as a request for a username and password.

chcon | An SELinux command that can be used to change the AVC contexts associated with a file.

chgrou | A command that assigns a new group owner to a file.

chmod | A command that modifies permissions to a file.

chown | A command that assigns a new user owner to a file.

Clam AntiVirus (ClamAV) | A cross-platform antivirus software toolkit developed for and used on Linux, BSD, and derivatives of Unix.

Cloud computing | A computing service offered by a provider via the cloud, or Internet. The service might be a virtual system, an application, or storage.

Cobbler project | An open source project for network-based installations of Linux distributions.

Command injection | A type of attack in which the attacker sends a shell command into a Web application and has it executed by the operating system.

Common Unix Printing System (CUPS) | The default print service for most modern Linux distributions. By default CUPS uses the Internet Print Protocol (IPP) but it can also administer with printers in a number of other protocols.

Common Vulnerabilities and Exposures (CVE) list | A list of operating-system security issues maintained by the MITRE corporation and sponsored by the National Cyber Security Division of the U.S. Department of Homeland Security.

Computer Aided Investigative Environment (CAINE) | A bootable live CD distribution available from <http://www.caine-live.net/>.

Confidentiality | A C-I-A triad concept associated with keeping private information private. It is sometimes required by laws or regulations.

Content Scramble System (CSS) | A DRM scheme used to encrypt commercial DVDs.

Crack | To break into a system or determine a password from a hashed value.

cron | A service for running administrative jobs on a regular basis. Protected by the /etc/cron.allow and /etc/cron.deny files.

cryptsetup | A command used to encrypt entire partitions using the device mapper.

Cyrus | An e-mail server developed at Carnegie-Mellon University, primarily for IMAP version 4 e-mail delivery.

D

Daemons | Standalone programs that run in the background, waiting for requests. A daemon typically has no direct interaction with users, as a user interfaces would.

Daniel J. Bernstein's DNS (djbdns) | A relatively lightweight DNS server alternative to BIND. It is released under a public-domain license, which is not open source.

Dbus | A message bus system for interprocess communication between a variety of applications and devices.

dd | The disk dump command. It supports a full copy of all data on a partition, volume, or drive.

dd_rescue | A variation on the dd command that reads data from back to front on a specified partition, volume, or drive. It is more error tolerant than dd.

Debian | A Linux distribution named after the original developer and his then-girlfriend.

debsums | A command on Ubuntu and Debian systems that uses MD5 checksums to see if changes have been made to files.

debug | The lowest log priority. Also provides the greatest detail.

Default runlevel | The group of services and daemons started by default when Linux is booted. Other services and daemons are started in other runlevels. If Linux is already running, a move to the default runlevel may also stop other services and daemons.

Denial of service (DoS) attack | An attack that involves overloading a network service, denying access to regular users. Not all DoS attacks are malicious. For example, they can be accidental results of certain types of connection attempts. Nevertheless, they should be blocked or slowed down as if they were attacks.

Development tools | Tools such as a compiler and a make utility. These enable you to compile and install software.

Dictionary attack | A type of attack that uses a list of words or strings to try to identify a password.

Digital Evidence & Forensic Toolkit (DEFT) | Built on Ubuntu Linux, DEFT includes a number of live tools for recovering data from live Microsoft operating systems. Available from <http://www.deftlinux.net/>.

Digital Millennium Copyright Act (DMCA) | A U.S. copyright law associated with digital intellectual property. Controversial in part because of fears of prosecution by researchers looking for better forms of encryption. One intent is to prevent the decryption of systems encoded with DRM schemes.

Digital Signature Algorithm (DSA) | An algorithm originally used by the U.S. government for digital signatures.

Direct attached storage (DAS) | A storage system connected directly to one or more computer systems using cables. Unlike NAS, a DAS system does not use network protocols to share data.

Discretionary access control | A security control system that limits access to objects such as files and directories to specified users and groups. Discretionary access control may be modified by authorized users.

Disk encryption subsystem | A system used to encrypt partitions and disks. Part of the device mapper and associated with the loadable dm_crypt module.

Distribution | A collection of software packages and a specific Linux kernel configuration.

Domain Name System (DNS) | A hierarchical database of domain names and IP addresses. Two major DNS services on Linux are BIND and djbdns.

Domain Name System Security Extensions (DNSSEC) | A way of performing validation of the source of information being provided about hostnames and IP addresses.

Dovecot | An open source e-mail service, designed for regular and secure versions of the POP and IMAP protocols.

E

ecryptfs | A command associated with the enterprise cryptographic filesystem.

edquota | A command to check and edit user quotas.

Elgamal | A probabilistic encryption scheme developed by Taher Elgamal (also spelled ElGamel).

emerg | A log priority that specifies very important messages. May also be shown as **panic** or **crit**.

Encapsulating Security Payload (ESP) protocol | An encryption protocol that provides authenticity, integrity, and confidentiality protection of network packets.

Enterprise cryptographic filesystem (eCryptfs) | A system adapted by Ubuntu to encrypt directories. Uses the **ecryptfs** command.

Environment variable | A word used to equate to something else, to obtain information that is relevant to the system or to a particular user.

Eric Raymond | A leader of the open source movement. Author of *The Cathedral and the Bazaar*. Also cofounder of the open source initiative at <http://www.opensource.org/>.

err | A log priority that adds error messages. May also be shown as **error**.

Exim | An open source SMTP server developed by the University of Cambridge. The default MTA for Debian systems.

ext2 | The second Linux extended filesystem.

ext3 | The third Linux extended filesystem. Includes journaling.

ext4 | The fourth Linux extended filesystem. Includes journaling and support for larger files.

F

Fedora | A Linux operating system developed by the Fedora Project and sponsored by Red Hat.

fdisk | A Linux partition configuration tool.

File Transfer Protocol (FTP) | A standard protocol and service for exchanging files.

Filesystem | A protocol for organizing and storing files. Most filesystems require a format. Typical filesystem formats include ext2, ext3, ext4, reiserfs, and xfs.

Filesystem Hierarchy Standard (FHS) | The way files and directories are organized in Linux. The FHS includes a list of standard directories and the types of files normally stored in those directories.

Firewalld | The next generation of Linux firewall.

Firewall | A piece of software or hardware used to inspect and make decisions on network communications.

G

General Public License (GPL) | A copy protection license often used on open source software.

Generic Security Services Application Program Interface (GSSAPI) | An application programming interface that accommodates communication primarily between Kerberos and services such as NFS.

getfacl | A command to review ACL settings for a file.

GNOME | The standard desktop environment in RHEL/CentOS and many other Linux distributions.

GNOME Display Manager (GDM) | A graphical login manager built by the developers of the GNOME desktop environment. May be used to log into graphical desktop environments other than GNOME.

GNU's Not Unix (GNU) | A recursive acronym for the work of the GNU Foundation, including the clones of Unix tools and libraries found in current Linux distributions.

gpg | A command that can encrypt and add digital signatures to a file.

Grand Unified Bootloader (GRUB) | The default boot loader for Ubuntu, Red Hat, and many other Linux distributions. Two versions of GRUB are in common use, with different options for security.

Graphical desktop environment | In Linux, the graphical desktop environment is separate from but requires the use of an X Window System Server. It may also include a window manager to control the placement of windows within that GUI. Two common Linux graphical desktop environments are GNOME and KDE.

Graphical login manager | A service for graphical logins to a Linux GUI. Three standard Linux graphical login managers are GDM, KDM, and XDM.

Gray-hat hacker | Someone who crosses ethical boundaries to locate vulnerabilities to help a company improve its protections.

Greg Kroah-Hartman | The Linux kernel developer responsible for the release of stable kernels.

groupadd | A command that can add a group.

groupdel | A command that can delete a group.

Group ID (GID) | The number associated with a group name in Linux, as defined in /etc/group and /etc/gshadow.

groupmod | A command that can modify the settings of a group in the files of the shadow password suite.

grpquota | The `mount` command option that sets user quotas. Often found in /etc/fstab.

H

Hardening | The process of locking a system down to protect it and any resources that may reside on it.

Honeypot | A computer system designed to appear to be vulnerable to attract the attentions of malicious users. It may include data that appears to be of value. It is carefully monitored and isolated from other systems on the local network. It may be used to gather intelligence about attack patterns.

Hydra | A password tool designed to identify passwords that may be easily cracked.

Hypervisor | Software that manages and runs a virtual machine.

I

inetd | The super server that runs a number of services when those services are called for, based on the port on which a network communication attempt is coming in.

info | A log priority that adds logging notes at the information level.

Init | The master process that starts all services.

Infrastructure as a Service (IaaS) | A cloud service in which the user purchases computing time with an operating system that is hosted within a virtual environment with whatever provider he or she chooses.

Inodes | Data structures associated with a file.

Integrated Services Digital Network (ISDN) | A group of standards for digital transmission of voice and data over the public switched telephone network.

Integrity | A C-I-A triad concept that indicates that data and systems should be able to be verified as original and not tampered with.

Inter-Cache Protocol (ICP) | A protocol used for communications between Web proxy servers such as Squid.

Internet Assigned Numbers Authority (IANA) | The organization responsible for domain names, IP addresses, and TCP/IP protocols on the Internet.

Internet Key Exchange (IKE) protocol | A protocol for key exchange used to set up a security association between different systems.

Internet Message Access Protocol version 4 (IMAP4) | An Application Layer e-mail protocol that supports client access to remote servers.

Internet Protocol Security (IPSec) | A member of the TCP/IP protocol suite that supports encrypted IP connections.

iptables | The Linux packet filtering command for firewalls and masquerading. Although it is the primary command used for packet filtering firewalls, it can also be used in network address translation.

iSeries | An IBM system that uses IBM Performance Optimization With Enhanced RISC (POWER) CPUs.

Itanium | A family of 64-bit CPUs developed by Intel. Red Hat supported Itanium CPUs through Red Hat Enterprise Linux 5, but will not continue support in later releases.

J

Java Cryptography Extension (JCE) | A framework for encryption associated with the Java programming language. May also be used with open source sendmail.

John the Ripper | A password tool designed to identify passwords that may be easily cracked.

Journalized filesystem | A filesystem that keeps track of changes to be written. Recovery tools can then use the journal to quickly find data on files to be written.

K

Kali Linux | A Linux distribution targeted at helping to perform penetration testing.

Kaspersky | A company that creates anti-malware systems for Linux and other operating systems.

KDE Display Manager (KDM) | A graphical login manager built by the developers of KDE. May be used to log into graphical desktop environments other than KDE.

Kerberos | A computer network authentication protocol. Developed at MIT as part of project Athena, it allows clients to prove their identities to each other with secure tickets.

Kerberos principal | An identity associated with Kerberos tickets. It includes the user, the Kerberos administrator, and the realm.

Kerberos realm | Typically, the name of the domain for the LAN or enterprise network, in uppercase letters. It's also an administrative concept that collects systems together for management and authentication purposes.

Kerberos Telnet | A version of the Telnet server that can use Kerberos tickets to enhance security.

Kerberos ticket | The proof on one system that verifies the identity of a second system.

Kernel | Another word for the operating system. This is the software that provides an interface between the hardware and the user-facing software. The kernel manages running processes and memory allocations and stores data on hard drives.

Kernel-space tools | Tools that depend on drivers and modules loaded with the Linux kernel.

Key distribution center (KDC) | Used by Kerberos to verify the credentials of a user before allowing access to various network services. The KDC is used to distribute cryptographic keys to avoid some of the challenges associated with key exchange.

Kickstart | A network-based installation system first created for Red Hat distributions.

Knoppix | A Linux distribution best known for its live CDs and DVDs.

Kubuntu | A release of the Ubuntu distribution that includes the KDE desktop environment as the default GUI.

L

Landscape | A system-management tool available from Canonical for managing and updating clients associated with the Ubuntu distribution.

Launchpad | A platform for bug tracking, open source software development, and more. Developed by Canonical.

ldd | A command that lists libraries used by a specified command. Its use requires the full path to the target command.

Lightweight Directory Access Protocol (LDAP) | A directory service for network-based authentication. LDAP communication can be encrypted.

Line Printer next generation (LPRng) | A server used to send print jobs to or from users or systems.

Linus Torvalds | The developer of the first Linux kernel and the current leader of Linux kernel developers.

Linux | A widely used Unix-based open-source operating system first developed by Linus Torvalds, available in several different distributions.

Linux/Apache/MySQL/P (LAMP) stack | A system in which Linux, Apache, and MySQL are integrated with Perl, Python, or PHP.

Linux distribution | A unified collection of applications, services, drivers, and libraries configured with a Linux kernel.

Linux kernel | The core of the Linux operating system. Different Linux kernels are in effect different operating systems. The Linux kernel includes a monolithic core and modular components.

Linux Kernel Organization | A nonprofit group established to distribute the Linux kernel and other open source software.

Developers of Linux distributions start with this kernel. Linus Torvalds releases new versions of the Linux kernel through the Linux Kernel Organization at <http://kernel.org>.

Linux Loader (LILO) | An alternative Linux boot loader. It is a legacy boot loader for many Linux distributions.

Linux patch management | The system of package updates on the Linux operating system.

Linux security modules (LSMs) | Frameworks for security support within the Linux kernel, associated with mandatory access control. Examples of LSMs include SELinux and AppArmor.

Linux Standard Base (LSB) | A standard specification for what it means to be Linux.

Linux unified key setup (LUKS) | A disk encryption specification that requires the dm_crypt module.

Live CD | A CD or DVD with a bootable operating system. That same data may also be loaded on a USB drive. Several Linux distributions are available in live CD format. When loaded, it provides password-free root administrative access to the system.

Loopback interface | An interface that is connected only to the system on which it resides. Anything sent to a loopback interface comes back to the system.

lsof | A command to list open files. The lsof -ni command lists open files related to networking, in numeric format.

M

Mail delivery agents (MDAs) | Related to servers such as Dovecot that facilitate the delivery of e-mail to user clients.

Mail retrieval agents (MRAs) | Associated with servers that collect e-mail from networks, such as fetchmail.

Mail submission agents (MSAs) | Related to servers that authenticate user connections to e-mail services. Frequently integrated into MTAs such as sendmail and Postfix.

Mail transfer agents (MTAs) | Associated with servers that transmit e-mail, such as sendmail and Postfix.

Mail user agents (MUAs) | Associated with client e-mail applications such as Evolution and Thunderbird.

Main repository | The Ubuntu repositories of supported open source software.

Mandatory access controls | Permissions that have been set administratively. Individual users cannot change them.

Mandriva | A Linux distribution based in France and Brazil.

Mangled | Describes a network packet with modified headers.

Martian packets | Packets with an impossible source or destination address. For example, a packet from the Internet with a source address of a private IP address network is a Martian packet.

Master browser | A system assigned to maintain a database of NetBIOS names and their services such as domain or workgroup membership.

Meta package | A Linux package that refers to other packages. For example, linux-image is a meta package that refers to the latest version of the generic kernel built for Ubuntu.

Mint | A Linux distribution based on Ubuntu, which is itself based on Debian.

mkfs | A Linux command to format and build a Linux filesystem. It's also a root for filesystem-specific commands such as `mkfs.ext3` and `mkfs.reiserfs`, which set up a filesystem to the given format.

Modular kernel | A kernel with drivers and features in module files that are separate from the kernel proper.

Monolithic kernel | A kernel with all the drivers and features built directly into a single, large kernel file.

Mount point | A directory that effectively becomes a shortcut to an external device. Rather than pointing to a place within the existing volume, the metadata in the filesystem will point to the external volume or device.

Multicast Domain Name Service (mDNS) protocol | Supports automated IP addressing without a DHCP server. Related to Microsoft's automatic private IP addressing and Apple's Bonjour protocols. Communicates using both TCP and UDP over port 5353.

Multi-Processing Modules (MPMs) | Used to add functionality to the Apache Web server.

Multitasking | Allows multiple programs to appear to be running in the operating system simultaneously. In reality, time in the CPU is shared across all programs, but the CPU switches between the programs so quickly that it appears that the programs are all running at the same time.

Multiverse repository | The Ubuntu repositories of unsupported software released under restricted licenses.

MySQL | An open source database program, currently owned by Oracle.

N

nc | The implementation of the Netcat utility, which can test and communicate over TCP and UDP connections.

Nessus | A vulnerability-scanning program with a Web-based interface, based on code that was previously released under open source licenses.

Netcat | A utility that can read TCP and UDP packets. Normally associated with the `nc` command.

netstat | A command used to verify network connections by port, routing tables by IP address, and more.

Network Address Translation (NAT) traversal protocol | A method of enabling IPSec-protected IP datagrams to pass through network address translation.

Network attached storage (NAS) | A storage system connected to a network, normally using file-sharing protocols such as NFS and Samba.

Network Basic Input/Output System (NetBIOS) | A name for a computer system commonly assigned on Microsoft-style networks. Associated with the Session Layer of the OSI model.

Network Information Service (NIS) | A directory service used to store information about systems on the network. This can include usernames and passwords and the service can be used for authentication.

Network Time Protocol (NTP) | A protocol and service for synchronizing clocks across systems on a network.

nfsnobody | The default account for unauthorized users who connect to the NFS file-sharing server.

nmap | A flexible command that can be used to scan TCP/IP network communication by protocol and port.

nobody | The default account for users on certain configured file-sharing servers.

Nonprivileged user | An account with standard end-user operating system permissions. This type of user does not have administrative permissions that would be found with a superuser, root, or administrative account.

Nonce | A random value transmitted by the server to the client when it indicates that authentication is required. The nonce is then combined with the username and password. The result of that is hashed. The resulting hash is what is transmitted to the server.

notice | A log priority that includes messages that might require attention.

O

Obscurity | The hiding of information, sometimes by placing it in nonstandard locations.

Online Certificate Status Protocol (OCSP) | A protocol and server to search for revoked digital certificates.

Open source | Software whose source code is available for anyone to inspect. This doesn't imply there is no cost associated with the software.

Open Systems Interconnection (OSI) Reference Model | A model of networking similar to the TCP/IP protocol suite. While the OSI model has seven layers, the TCP/IP protocol suite has four layers.

optional | A PAM flag that labels a configuration line that is normally ignored unless there are no other PAM flags in the file.

P

Packages | Software bundles that contain all the files required for a piece of software to be installed. Each distribution has software that will unbundle and install the software inside each of these packages.

Paravirtualization | A scenario in which software is installed on the guest operating system such that the guest operating system is aware that it is operating inside a virtual system. If this helper software is not installed, the guest cannot function.

password | A PAM module that can control changes to user passwords and limit the number of login attempts. Also known as password management.

Password Authentication Protocol (PAP) | A protocol for validating users before allowing access.

Password cracker | A specialized piece of software that either guesses the password from a number of known passwords or simply tries every password possible.

Patch | An incremental upgrade to software like the Linux kernel.

Perl | A dynamic scripting language developed by Larry Wall. Frequently used with Apache for Web sites.

PHP: Hypertext Preprocessor (PHP) | A scripting language associated with dynamic Web sites. Frequently used with Apache.

Ping storm | A condition where a system sends a flood of ICMP packets to a server. May be created with the `ping -f` command.

Platform as a Service (PaaS) | A cloud service that may provide a complete Web application solution for easier application development. Depending on the service provider chosen, you may get access to the operating system or it may just give you access to the application development features.

Pluggable authentication modules (PAMs) | A series of configuration files that provide dynamic authentication for administrative and other services.

PolicyKit | Supports fine-grained control administrative tools from regular accounts. The focus of the PolicyKit is on the GNOME desktop environment.

Polkit | Allows for system-wide control of permissions. Formerly called PolicyKit.

Portage system | A set of scripts that build software. The portage system is based on the FreeBSD ports, which is a set of third-party packages that can be installed from source.

Post Office Protocol version 3 (POP3) | An Application Layer e-mail protocol that supports e-mail client downloads of incoming messages.

Postfix | An open source SMTP server originally developed at IBM. It's designed to be simpler than sendmail.

PostgreSQL | An open source database.

Primary Domain Controller (PDC) | A master server on a Microsoft Windows NT domain that controls and can grant access to a number of computer resources based on the usernames and passwords in its database. Linux with Samba can be configured as a PDC.

/proc/kcore | A pseudo file that dynamically represents the contents of RAM on the local system.

Pro File Transfer Protocol daemon (ProFTPD) | A popular FTP server with a basic configuration file similar to the Apache Web server. Supports multiple virtual FTP servers.

pSeries | The IBM Reduced Instruction Set Computing (RISC) server and workstation product line designed for Unix systems. Some Red Hat Enterprise Linux releases are built for the pSeries.

Public switched telephone network (PSTN) | The regular telephone network for voice communications.

Pure File Transfer Protocol daemon (Pure-FTPd) | An actively supported server that can run 100 percent with non-root accounts.

Python | A multi-paradigm programming language frequently used with Apache for Web sites.

Q

Qmail | The self-declared replacement for sendmail, developed by Daniel J. Bernstein, who also developed djbdns.

quotacheck | A command that creates, checks, and repairs quota-management files such as aquota.user and aquota.group.

R

RainbowCrack | A password tool designed to identify passwords that may be easily cracked.

Rainbow tables | Tables containing precomputed password hash values to speed up the cracking of passwords.

Real-time Transport Protocol (RTP) | A standard packet format for VoIP and video communications.

Rebuild | A Linux distribution built from the source code released by another distribution. For example, because CentOS uses Red Hat source code, CentOS Linux is a rebuild of Red Hat Enterprise Linux.

Recursive query | A search of a DNS database that is sent to other DNS servers if the information is not available locally.

Red Flag Linux | A Linux distribution developed in China.

Red Hat | The company that develops the Linux distributions called Red Hat Enterprise Linux and Fedora Core.

Red Hat Bugzilla | A system for bug reports on Red Hat distributions.

Red Hat Network (RHN) | A group of system-management services to manage packages, administer scripts, and more. Such services may be applied to subscribed clients and servers on a network.

Red Hat Package Manager (RPM) | A format used to not only include all the files associated with a program but also to include metadata associated with the contents of the package and any dependencies the package may have.

Red Hat Proxy server | A proxy server dedicated to caching downloaded packages from the Red Hat Network.

Red Hat Satellite server | A version of the Red Hat Network designed for local use on an enterprise network. Includes an embedded Oracle database.

Red Hat security advisories (RHSAs) | Announcements of security issues from the Red Hat Security Team.

reiserfs | The Linux filesystem based on balanced trees, suited for groups of large and small files.

Remote Authentication Dial In User Service (RADIUS) | A system for remote user authentication, frequently used to authenticate connections over telephone modems.

Remote procedure call (RPC) | A way of getting a function or method to run on a separate system. Typically, network file-sharing solutions use RPC to function.

required | A PAM flag that labels a configuration line that must work for the authentication attempt to succeed. If the line fails, PAM continues to check the other lines in the file.

requisite | A PAM flag that labels a configuration line that must work for the authentication attempt to succeed. If the line fails, PAM immediately returns a failure in the authentication attempt.

Restricted repository | The Ubuntu repositories of software released under restricted licenses.

root | By itself, the name of the standard Linux administrative user. The top-level root directory is symbolized by the forward slash (/). In contrast, the home directory of the root user is /root, which is a subdirectory of the top-level root directory (/).

Rootkit | A type of malware that enables a malicious user to take root administrative control of a Linux system. A rootkit will also generally disguise the existence of malware by replacing system utilities.

RSA | A public-key encryption algorithm named for its developers, Rivest, Shamir, and Adleman.

rsync | A command that synchronizes files from one location to another. May be used in conjunction with SSH.

rsyslog | The latest system for system and kernel logs. Also supports secure transmission of log information to a central logging server.

Runlevel | A mode of operation in Linux associated with a group of services and daemons. Specified services and daemons are started or killed when starting a particular runlevel.

S

Salt | A 56-bit key added to a hash to make it more difficult to set up a manageable rainbow table.

secon | An SELinux command that returns the context settings of a specified file or directory.

Secure Hash Algorithm (SHA) | A set of cryptographic hash functions developed by the U.S. National Security Agency.

Secure Sockets Layer (SSL) | Software used to encrypt messages in transit. Commonly, SSL is used with Transport Layer Security (TLS) to encrypt messages between a Web server and a browser, although other applications also use SSL/TLS.

Security Administrator's Integrated Network Tool (SAINT) | A commercial-grade vulnerability scanner.

Security Administrator Tool for Analyzing Networks (SATAN) | An older open source network analyzer. Later versions were released under proprietary licenses as part of SAINT.

Security Enhanced Linux (SELinux) | A mandatory access control system that uses Linux security modules in the kernel. Developed by the U.S. National Security Agency. Not compatible with AppArmor.

sendmail | The open source SMTP server maintained by the Sendmail Consortium. Not to be confused with the commercial SMTP server known as Sendmail.

Sendmail | A commercial SMTP server maintained by Sendmail, Inc. Not to be confused with the open source SMTP server known as sendmail.

Services | Autonomous processes that generally start up at boot time and run in the background while the operating system is running and operational.

session | A PAM module that can control mounting and logging. Also known as session management.

Session Initiation Protocol (SIP) | An IP network protocol frequently used in VoIP communications.

sestatus | An SELinux command that returns the overall status of SELinux on the local system.

setfac1 | A command to create or modify ACL settings for a file.

Set group ID (SGID) bit | A special permission commonly applied to a directory. With the SGID bit, users who are members of the group that owns the directory have permission to read and write to all files in that directory. The SGID bit assigns the group owner of the directory as the group owner of all files copied to that directory.

Set user ID (SUID) bit | A special permission that allows others to execute the given file with the rights of the user owner of the file.

sg | A command that can connect with the privileges of another group. Requires a group password in /etc/gshadow.

Shadow password suite | Files that make up the local Linux password authentication database. The files are /etc/passwd, /etc/shadow, /etc/group, and /etc/gshadow. As originally developed, it also includes the Linux **login**, **su**, and **passwd** commands.

Simple Mail Transfer Protocol (SMTP) | An Application Layer e-mail protocol used to send e-mail messages.

Simplified Mandatory Access Control Kernel (SMACK) | A Linux security module for mandatory access control. Functionally similar to SELinux and AppArmor.

Sleuth Kit | See The Sleuth Kit.

smb.conf | The main configuration file for the Samba/CIFS file server.

Snort | An open source network-based intrusion-detection system.

Socket | A file through which a server communicates with a client.

Software development kit (SDK) | A set of libraries and headers that enable software to be built for a particular device.

Software RAID | A version of redundant array of independent disks (RAID) that uses partitions instead of disks as components of the array.

Solaris | A variant of Unix originally developed by the former Sun Microsystems.

Source code | Human-readable computer language that can be collected and compiled into a computer program, library, or application.

Spacewalk | An open source system-management server based on the source code of the Red Hat Network Satellite server.

SpamAssassin | A program for filtering unwanted e-mail.

Split DNS | A common DNS configuration. With split DNS, you have two DNS servers—one configured to allow public inquiries and the other for internal clients only.

Spoofed | Describes a network transmission with a false source address.

Spyware | A type of malware that collects information about users without their knowledge.

sshd | The daemon for the SSH service.

Sticky bit | A special permission commonly applied to a directory. With the sticky bit and full permissions, all users can write to the associated directory. However, ownership is retained, so users won't be able to overwrite files copied by other users.

Stock kernel | The kernel developed and released by the Linux Kernel Organization without any additional alterations by distribution developers.

Storage area network (SAN) | A storage system that may consist of one or more NAS or DAS systems.

strace | A command that traces the system calls used by another command. Primarily used for troubleshooting.

su | A command that can connect with the privileges of another user. Requires the password of the target user. When no target user is specified, the root administrative user is assumed.

Subversion | A system for version control frequently used on many open source projects, including nmap.

sudo | A command that can connect as the administrative user if authorization is configured in /etc/sudoers.

sufficient | A PAM flag that labels a configuration line. If the line works, PAM immediately returns a success message in the authentication attempt.

SUSE | A Linux distribution originally developed in Germany, now owned by Novell.

syslog | The system log message service, associated with the syslogd daemon. When combined with the kernel log daemon, known as klogd, it is sometimes shown as the sysklogd daemon.

sysstat | A package that tracks the RAM and CPU usage on a system with the help of the cron service.

Systemctl | The primary utility used to control a system that uses systemd. With systemctl, you can send messages to processes or services as well as shut down or reboot the entire system.

Systemd | The master process that controls all the service stops and starts and provides a foundation for management of the entire system.

T

TCP Wrapper | A program that wraps a network service, allowing access control determinations to be made outside of the service program that may not have that functionality built into it.

Telephone modem | A modulator-demodulator for translating data bits into the sine waves associated with the PSTN. Cable modems and DSL modems are not true modems, as they do not modulate or demodulate data.

The Sleuth Kit | A package of tools that can be used to analyze disk images.

Threat vectors | Ways for an attacker or adversary to compromise a target system.

Ticket-granting server (TGS) | The Kerberos server that grants TGTs. The TGS works hand in hand with the KDC.

Ticket-granting ticket (TGT) | A sort of time-limited super-ticket issued by the KDC that supports access to other systems without additional authentication. The TGT is a pass that assures other servers that you have been authenticated.

Tomcat | A Java application server used to handle Web-based requests and then hand them to an application written in Java that the Tomcat server runs.

TOMOYO | A Linux security module for mandatory access control. Functionally similar to SELinux and AppArmor.

Transition SIGnature (TSIG) | A protocol used to authenticate data exchanges between DNS servers.

Translation lookaside buffer (TLB) | A piece of hardware that determines the real, physical memory address based on the virtual memory address the program knows about.

Transport Layer Security (TLS) | Software used to encrypt messages in transit. Commonly, TLS is used with Secure Sockets Layer (SSL) to encrypt messages between a Web server and a browser, although other applications also use SSL/TLS.

Tripwire | An intrusion detection system. Open source and commercial versions are available.

Trivial File Transfer Protocol (TFTP) | A protocol and service that uses a simplified form of FTP.

Trojan horse | A kind of malware that disguises itself as a useful program or command.

Turbolinux | A Linux distribution originally developed in Japan.

U

Ubuntu | A Linux distribution based on Debian. It is managed by Canonical.

Ubuntu security notices (USNs) | Alerts based on security issues that affect different releases of the Ubuntu distribution.

Universe repository | The Ubuntu repositories of unsupported software released under open source licenses.

Unix | An operating system developed in the late 1960s by scientists at AT&T Bell Labs, originally named Unics.

Upstart | A process designed to make system and service initialization more manageable.

useradd | A command that can add a user.

userdel | A command that can delete a user.

User ID (UID) | The number associated with a user name in Linux, as defined in /etc/passwd.

usermod | A command that can modify the settings of a user in the files of the shadow password suite.

User private group scheme | The standard in Linux where a special group is created for every user. By default, the user and group names (along with the UID and GID numbers) are identical. The user is the only standard member of that group.

User-space tools | Tools that do not depend on the Linux kernel.

usrquota | The `mount` command option that sets user quotas. Often found in /etc/fstab.

V

Van Eck phreaking | A method for interpreting the emissions from computer displays to recover the associated image.

Very Secure File Transfer Protocol daemon (vsftpd) | The open source FTP server used by developers of Red Hat, SUSE, and Debian to share their distributions.

Virtual LAN (VLAN) | A LAN created on the same physical network as another LAN. Because both LANs are separate and distinct, they are virtual.

Virtual machines | Logical presentations of physical hardware to an operating system. This allows multiple operating system instances to run on a single set of hardware.

Virtual memory | Memory that appears to exist but is not actually physically available.

Virtual Network Computing (VNC) | A system for sharing views of graphical desktop environments over a network.

Virus | A program that can copy itself and infect a computer with malware.

VMware | A family of virtual machine software that works with everything from software-based virtualization with programs like VMware Player to bare-metal virtualization with programs like vSphere.

vsftpd.conf | The main configuration file for the very secure FTP daemon service.

W

w | A command that lists currently logged in users and the process currently being run by that user.

warn | A log priority that provides warning messages. May also be shown as warning.

Washington University File Transfer Protocol daemon (WU-FTPD) | A popular FTP server that is no longer supported and is reported to have security flaws.

Web application firewall (WAF) | A firewall that protects a Web application.

White-hat hacker | Someone who works only for good using similar techniques and skills as those of a black-hat or gray-hat hacker.

Whitelists | Lists of approved behaviors or systems. For example, a whitelist might provide a list of systems that are allowed to connect.

who | A command that lists currently logged in users.

Wikis | Ways to collaboratively document something. In the case of Linux, wikis are often used to enable the development team or, in some cases, the users to write documentation that is provided to the user community.

Winbind | A component of the Samba file server that supports integration of Linux/Unix and Microsoft authentication information.

Windows Internet Name Service (WINS) servers | Servers that include a database of NetBIOS names and IP addresses.

Wireless intrusion detection system (WIDS) | Software that can help detect unauthorized attacks on a wireless network. One example is available from the aircrack-ng package.

Wireless receiver-transmitter (WRT) software | Customizable firmware for wireless access points. Two Linux-based WRTs are DD-WRT and OpenWrt.

Worm | A self-replicating malware program.

Wrapper | A program that wraps existing service programs, offering them network-level protections.

X

X | A windowing system developed for Unix in the 1980s. X still provides the foundation for graphical user interfaces today.

X11 | A protocol associated with the X Window system.

X Display Manager Control Protocol (XDMCP) | A protocol that enables remote logins to a GUI. It is normally associated with TCP/IP port 177.

X Display Manager (XDM) | A graphical login manager built by the developers of the [X.Org](#) GUI server.

Xfce desktop environment | An alternative desktop environment to GNOME and KDE. The default desktop environment on the Xubuntu variant of Ubuntu Linux.

XFree86 | An implementation of the X graphical interface.

xfs | The Linux filesystem developed by Silicon Graphics suited to larger files.

xinetd | An improved version of inetd.

Xubuntu | A release of the Ubuntu distribution that includes the Xfce desktop environment as the default GUI.

Y

Yellowdog Updater, Modified (yum) | A utility that downloads Red Hat Package Manager (RPM) files from a network site, identifies the packages that must be installed before the package you really want is installed, and then installs everything in the correct order.

Z

ZENworks | A system-management server released by Novell. It can be used to administer patches and more on both SUSE Linux Enterprise Server and Red Hat Enterprise Linux systems.

Zone files | When associated with DNS, a database of hostnames and IP addresses for a specific authoritative domain.

Zone updates | A data exchange between DNS servers with respect to hostnames and IP addresses of a specific domain.

References

- Aircrack-ng. <http://www.aircrack-ng.org/> (accessed March 2010).
- Aircrack-ng. Basic documentation (Aircrack-ng, September 9, 2009). http://www.aircrack-ng.org/doku.php?id=getting_started/ (accessed March 2010).
- AirTight Networks. "WPA/WPA2 TKIP Attack" (AirTight Networks, 2010).
<http://www.airtightnetworks.com/home/resources/knowledge-center/wpa-wpa2-tkip-attack.html> (accessed April 9, 2010).
- Aitchison, Ron. *Pro DNS and BIND*. Berkeley, CA: Apress, 2005. <http://www.zytrax.com/books/dns/> (accessed April 6, 2010).
- Amazon Elastic Compute Cloud (Amazon EC2). Amazon Web Services. <http://aws.amazon.com/ec2/> (accessed April 14, 2010).
- Beale, Jay, and Russ Rogers. *Nessus Network Auditing*, 2nd ed. Burlington, MA: Syngress Publishing, 2008.
- "BIND9 Server Howto" Ubuntu Community Documentation, April 2, 2010.
<https://help.ubuntu.com/community/BIND9ServerHowto/> (accessed April 8, 2010).
- Bowen, Rich and Ken Coar. *Apache Cookbook: Solutions and Examples for Apache Administrators*. Sebastopol, CA: O'Reilly Media, 2008.
- Briglia, Tom. "UNIX and Linux Access Control Standard" (Stanford University, 2007).
https://www.stanford.edu/dept/as/ia/security/policies_standards/AS_standards/libwrap_access_control_standard_1.0.html (accessed March, 2010).
- Burdach, Mariusz. "Forensic Analysis of a Live Linux System, Pt. I." (Symantec, March 22, 2004).
<http://www.symantec.com/connect/articles/forensic-analysis-live-linux-system-pt-1/> (accessed May 2010).
- Burdach, Mariusz. "Forensic Analysis of a Live Linux System, Pt. II." (Symantec, April 11, 2004).
<http://www.symantec.com/connect/articles/forensic-analysis-live-linux-system-pt-2/> (accessed May 2010).
- Carrier, Brian. *The Sleuth Kit Informer*, issue 1, February 15, 2003. <http://www.sleuthkit.org/informer/sleuthkit-informer-1.html> (accessed May 2010).
- Chapman, Ben, and Champ Clark III. *Asterisk Hacking*, Burlington, MA: Syngress Publishing, 2007.
- Chapple, Mike. "Choosing the Right Firewall Topology: Bastion Host, Screened Subnet or Dual Firewalls" ([SearchSecurity.com](#), October 17, 2005). [http://searchsecurity.techtarget.com/tip/1,289483,sid14_gci906407_mem1,00.html/](http://searchsecurity.techtarget.com/tip/1,289483,sid14_gci906407_mem1,00.html) (accessed April 4, 2010).
- . "Nessus 3 Tutorial" ([SearchSecurity.com](#), June 6, 2008).
http://searchsecurity.techtarget.com/generic/0,295582,sid14_gci1159345_mem1,00.html (accessed May 2010).
- Chapple, Michael J., John D'Arcy, and Aaron Striegel. "An Analysis of Firewall Rulebase (Mis) Management Practices." *The ISSA Journal* 7, no. 2, February 2009. <https://dev.issasec.org/Library/Journals/2009/February/ISSA%20Journal%20February%202009.pdf> (accessed April 6, 2010).
- Corbet, Jonathan. "Nftables: A New Packet Filtering Engine" ([LWN.net](#), March 24, 2009). <http://lwn.net/Articles/325196/> (accessed May 2010).
- Curran, Christopher. "Red Hat Enterprise Linux 6 Virtualization Guide" Edition 1 beta (Red Hat, 2010).
http://www.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/6-Beta/html-single/Virtualization/ (accessed May 2010).
- "Department of Defense Trusted Computer System Evaluation Criteria," DoD 5200.28-STD, released December 1985.
<http://csrc.nist.gov/publications/secpubs/rainbow/std001.txt> (accessed February 2010).
- DiMaggio, Len. "Understanding Your (Red Hat Enterprise Linux) Daemons." *Red Hat Magazine*, March 9, 2007.
<http://magazine.redhat.com/2007/03/09/understanding-your-red-hat-enterprise-linux-daemons/> (accessed March 18, 2010).
- DuBois, Paul. *MySQL*, 4th ed. Indianapolis: Addison-Wesley Professional, 2008.
- "eCryptfs—Enterprise Cryptographic Filesystem" (Canonical, Ltd, 2010). <https://launchpad.net/ecryptfs> (accessed March 2010).
- "English Country Names and Code Elements," ISO Standard 3166 (ISO, 2010).
http://www.iso.org/iso/english_country_names_and_code_elements/ (accessed April 6, 2010).

- Evans, Chris. Manpage of vsftpd.conf, the very secure File Transfer Protocol daemon (vsftpd, May 28, 2009).
http://vsftpd.beasts.org/vsftpd_conf.html (accessed April 2010).
- "Federal Desktop Core Configuration" (FDCC, April 16, 2010). <http://nvd.nist.gov/fdcc/> (accessed May 2010).
- Fenzi, Kevin. "Linux Security HOWTO" (Linux Documentation Project, January 2004). <http://tldp.org/HOWTO/Security-HOWTO> (accessed February 2010).
- Fifield, David. "The NMAP Scripting Engine," presentation at the 2010 Free and Open Source Software Developers' European Meeting, February 6, 2010. <http://www.bamsoftware.com/talks/fosdem-2010.pdf> (accessed May 2010).
- Fioretti, Marco. "How to Set Up and Use Tripwire." *Linux Journal*, April 28, 2006. <http://www.linuxjournal.com/article/8758> (accessed April, 2010).
- Free Software Foundation, The. "Gnu Privacy Guard (GnuPG) Mini Howto" (De Winter Information Solutions, August 10, 2004). http://www.dewinter.com/gnupg_howto/english/GPGMiniHowto.html (accessed March 2010).
- [Freestandards.org](#). "Filesystem Hierarchy Standard," developed in 2003; revised January 2004. Information available from <http://www.pathname.com/fhs/> (accessed March 2010). The work of the [Freestandards.org](#) group is now being maintained by The Linux Foundation.
- Fuller, Johnray, John Ha, David O'Brien, Scott Radvan, Eric Christensen, and Adam Ligas. "Fedora 12 Security Guide: A Guide to Securing Fedora Linux, Edition 12.1" (Red Hat, 2009). <http://docs.fedoraproject.org/security-guide/f12/en-US/html/index.html> (accessed February 2010).
- Gerhards, Rainer. "HOWTO install rsyslog" ([Rsyslog.com](#), 2008). <http://www.rsyslog.com/doc-install.html> (accessed April, 2010).
- GNU Project, The. "GNU GRUB" ([GNU.org](#), 2010). <http://www.gnu.org/software/grub/> (accessed February 2010).
- Grubb, Steve. "Hardening Red Hat Enterprise Linux 5." Presented at the Red Hat Summit, 2008.
<http://people.redhat.com/sgrubb/files/hardening-rhel5.pdf> (accessed April 14, 2010).
- Herzog, Peter. *OSSTMM 3.0 Lite: Introduction and Sample to the Open Source Security Testing Methodology Manual* (ISECOM, 2010). <http://www.isecom.org/osstmm/> (accessed February 2010).
- Herzog, Peter, et al. *Hacking Linux Exposed*, 3rd ed. New York, NY: McGraw-Hill, 2008.
- Hoopes, John, ed. *Virtualization for Security: Including Sandboxing, Disaster Recovery, High Availability, Forensic Analysis, and Honeypotting*. Burlington, MA: Syngress Publishing, 2009.
- Hornat, Charles, et al. "Interfacing with Law Enforcement FAQ: For Incident Handlers and Other Information Security Professionals," ver. 1.0 (The SANS Institute, January 15, 2004). [https://www.sans.org\(score/faq/law_enf_faq/](https://www.sans.org(score/faq/law_enf_faq/)) (accessed May 2010).
- Horrigan, John. "Home Broadband Adoption 2009" (Pew Internet and American Life Project, June 2009).
<http://www.pewinternet.org/~media//Files/Reports/2009/Home-Broadband-Adoption-2009.pdf> (accessed April 2010).
- IBM System X Virtualization Strategies. <http://www-01.ibm.com/redbooks/community/pages/viewpage.action?pageld=2788036> (accessed February 2010).
- Jang, Michael. *Linux Annoyances for Geeks*. Sebastopol, CA: O'Reilly Media, 2006.
- . *Linux Patch Management*. Upper Saddle River, NJ: Pearson Education, 2006.
- . *Linux Patch Management: Keeping Linux Systems Up To Date*. New York: McGraw-Hill, 2007.
- . *RHCE Red Hat Certified Engineer Study Guide*. New York: McGraw-Hill, 2007.
- . *Ubuntu Server Administration*. New York: McGraw-Hill, 2008.
- Jones, Pamela. <http://www.groklaw.net/> (accessed February 2010).
- Kingsley-Hughes, Adrian. "Are Ubuntu Users Covered by H.264 License? It Depends" (ZDNet, May 5, 2010).
<http://www.zdnet.com/blog/hardware/are-ubuntu-users-covered-by-h264-license-it-depends/8228> (accessed May 2010).
- Kismet. <http://www.kismetwireless.net/> (accessed March 2010).
- Kotadia, Munir. "New Hacking Tool: Chocolate" (ZDNet, April 20, 2004). <http://www.zdnet.com/news/new-hacking-tool-chocolate/135565> (accessed May 2010).
- Kroah-Hartman, Greg. *Linux Kernel in a Nutshell*. Sebastopol, CA: O'Reilly Media, 2007.
- Kumar, Avinesh, and Sandesh Chopdekar. "Get Started with the Linux Key Retention Service" (IBM, April 11, 2007).
<http://www.ibm.com/developerworks/linux/library/l-key-retention.html/> (accessed April 14, 2010).
- Linux Kernel Organization. "The Linux Kernel Archives" ([Kernel.org](#), nd). <http://kernel.org/> (accessed February 2010).

- Lyon, Gordon "Fyodor." *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning* (Insecure.com, 2009). <http://nmap.org/book/toc.html> (accessed May 2010).
- Matulis, Peter. *Centralised Logging with rsyslog* (Canonical, September 2009).
<http://www.ubuntu.com/system/files/CentralLogging-v4-20090901-03.pdf> (accessed April 2010).
- MIT Kerberos Team. "Kerberos, the Network Authentication Protocol" (MIT.edu, April 8, 2010). <http://web.mit.edu/Kerberos/> (accessed April, 2010).
- National Security Agency, Central Security Service. "Security-Enhanced Linux" (NSA.gov, January 15, 2009).
<http://www.nsa.gov/research/selinux/> (accessed March and April, 2010).
- National Security Agency, Operating Systems Division Unix Team of the Systems and Network Analysis Center. "Guide to the Secure Configuration of Red Hat Enterprise Linux 5, Revision 3" (NSA.gov, October 21, 2009).
http://www.nsa.gov/ia/_files/os/redhat/rhel5-guide-i731.pdf (accessed March 2010).
- Olejniczak, Stephen P., and Brady Kirby. *Asterisk for Dummies*, Hoboken, NJ: Wiley Publishing, 2007.
- Open Source TCG Software Stack, The. <http://trousers.sourceforge.net> (accessed February 2010).
- Operating Systems Division Unix Team of the Systems and Network Analysis Center. "Guide to the Secure Configuration of Red Hat Enterprise Linux 5" revision 3. Released by the National Security Agency, October 21, 2009.
http://www.nsa.gov/ia/guidance/security_configuration_guides/operating_systems.shtml#linux2 (accessed March 18, 2010).
- Orebaugh, Angela, and Becky Pinkard. *Nmap in the Enterprise: Your Guide to Network Scanning*. Burlington, MA: Syngress Publishing, 2008.
- O'Sullivan, Jill, and Gene Ciaola. *Enterprise Resource Planning: A Transitional Approach from the Classroom to the Business World*. Hightstown, NJ: McGraw-Hill Primis Custom Publishing, 2008.
- Pennington, Havoc, Anders Carlsson, and Alexander Larsson. "D-Bus Specification" version 0.12, 2007 (Freedesktop.org, nd).
<http://dbus.freedesktop.org/doc/dbus-specification.html> (accessed February 2010).
- Petullo, Mike. "Implementing Encrypted Home Directories." *Linux Journal*, August 1, 2003.
<http://www.linuxjournal.com/article/6481> (accessed March 2010).
- Pogue, Chris, Cory Altheide, and Todd Haverkos. *UNIX and Linux Forensic Analysis DVD Toolkit*. Burlington, MA: Syngress Publishing. 2008.
- Rash, Michael. *Linux Firewalls: Attack Detection and Response with iptables, psad, and fwsnort*. San Francisco, CA: No Starch Press, 2007.
- Raymond, Eric. *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. Sebastopol, CA: O'Reilly Media, 2001.
- Red Hat. "Red Hat Enterprise Linux 5 Deployment Guide, Edition 4" (Red Hat, October 2008).
https://www.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/5.4/html/Deployment_Guide/ (accessed March and April, 2010).
- . "Red Hat Enterprise Linux 5 Installation Guide, Edition 3" (Red Hat, 2008). http://www.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/5/html/Installation_Guide/index.html (accessed March 2010).
- . "Red Hat Enterprise Linux Documentation" (Red Hat, 2010). <http://www.redhat.com/docs/manuals/enterprise/> (accessed May 2010).
- RFC 4340 Internet Assigned Numbers Authority. <http://www.iana.org/assignments/port-numbers/> (accessed March 11, 2010).
- Russell, Rusty. "Linux Packet Filtering HOWTO" (The netfilter.org Project, January 24, 2002).
<http://www.iptables.org/documentation/HOWTO/packet-filtering-HOWTO.html> (accessed March, 2010).
- SAINT Corporation. "Running a Default Vulnerability Scan: A Step-by-Step Guide" (SAINT Corporation, 2009).
http://www.saintcorporation.com/resources/SAINT_scan.pdf (accessed May 2010).
- Samba Team. The configuration file for the Samba suite. (Samba.org, nd).
<http://samba.org/samba/docs/man/manpages-3/smb.conf.5.html> (accessed April 2010).
- "SELinux Future Work" (National Security Agency, Central Security Service, January 15, 2009).
<http://www.nsa.gov/research/selinux/todo.shtml/> (accessed May 2010).
- "SettingUpNISHowTo." Community Ubuntu Documentation (Ubuntu Documentation, September 10, 2009).
<https://help.ubuntu.com/community/SettingUpNISHowTo> (accessed February 2010).

- Skoric, Miroslav. "LILO Mini-HOWTO" (The Linux Documentation Project, November 8, 2009).
<http://tldp.org/HOWTO/LILO.html> (accessed February 2010).
- Snort Project, The. *SNORT Users Manual 2.8.6* (Sourcefire, Inc., April 26, 2010).
http://www.snort.org/assets/140/snort_manual_2_8_6.pdf (accessed April 16, 2010).
- Stallman, Richard. "Can You Trust Your Computer?" (GNU.org, nd). <http://www.gnu.org/philosophy/can-you-trust.html> (accessed February 2010).
- Suehring, Steve, and Robert Ziegler. *Linux Firewalls*. 3rd ed. Indianapolis, IN: Novell Press, 2006.
- The MITRE Corporation. "Common Vulnerabilities and Exposures" (Mitre.org, March 16, 2009). <http://cve.mitre.org/cve/> (accessed April 2010).
- Verhelst, Wouter. "Securing NFS." *Free Software Magazine*, November 26, 2006.
http://www.freesoftwaremagazine.com/columns/securing_nfs (accessed April 2010).
- Virijevich, Paul. "Intrusion Detection with AIDE" (Linux.com, January 20, 2005).
<http://www.linux.com/archive/feature/113919/> (accessed April 2010).
- Wessels, Duane. *Squid: The Definitive Guide*. Sebastopol, CA: O'Reilly Media, 2004.
- Whitaker, Andrew, and Daniel P. Newman. *Penetration Testing and Network Defense*. Indianapolis: Cisco Press, 2006.
- Zeuthen, David. "Desktop and Hardware Configuration." *Red Hat Magazine*, January 2005.
<http://www.redhat.com/magazine/003jan05/features/hal/> (accessed March 18, 2010).
- . "PolicyKit Library Reference Manual" (Freedesktop.org, 2007). <http://hal.freedesktop.org/docs/PolicyKit/> (accessed February 2010).
- Zhang, Wei. "Build a RADIUS server on Linux" (IBM, May 25, 2005). <http://www.ibm.com/developerworks/library/l-radius/> (accessed April 2010).

Index

The index that appeared in the print version of this title was intentionally removed from the eBook. Please use the search function on your eReading device to search for terms of interest. For your reference, the terms that appear in the print index are listed below.

64-bit kernel

A

a2dismod command
a2enmod command
AAA
acceptable use policy
access control lists (ACLs)
access controls
access modes
access vector cache (AVC)
access vectors
access.log file in CUP
account behavior
account management
account module
acl directive
ACLs. *See* access control lists
ACPI. *See* Advanced Configuration and Power Interface
action mechanism
activate reverse path filtering
Administration tool
administrative access
administrative jobs
administrative login attempts
administrative privileges hierarchy
administrative privileges in services
administrator security
administrators
Adobe applications
Adobe products
ads
ADSL. *See* analog display services interface
Advanced Configuration and Power Interface (ACPI)
Advanced Encryption Standard (AES)
Advanced Intrusion Detection Environment (AIDE)
Advanced Micro Devices (AMD)
Advanced Package Tool (APT)
Advanced RISC Machine (ARM)
AES. *See* Advanced Encryption Standard
AFS tokens. *See* Andrew Filesystem tokens
ah directive

AH protocol. *See* Authentication Header protocol
AIDE. *See* Advanced Intrusion Detection Environment
alert log message priority
alerts
allow-update directive
Amazon Web Services (AWS)
AMD. *See* Advanced Micro Devices
Anaconda installation program
analog display services interface (ADSI)
Andrew Filesystem (AFS) tokens
antivirus systems
Apache
Apache authentication database
Apache configuration file
Apache Foundation
Apache logs
Apache module mod_security
Apache modules
Apache Web server
apache2ctl command
API. *See* application programming interface
APIPA. *See* automatic private IP addressing
App Engine
AppArmor. *See* Application Armor
apparmor_status command
Apple's Bonjour service
Application Armor (AppArmor)
application developers
application programming interface (API)
application security
application server
application-specific bug reports
applications category of virtualization
APT. *See* Advanced Package Tool
apt-* commands
apt-cache search linux-image command
apt-cache show command
apt-get utility
apt-mirror command
aptitude command
aquota.group file
aquota.user file
Arch Linux
architecture of kernels
ARM. *See* Advanced RISC Machine
ARP inspection. *See* dynamic address resolution protocol inspection
Asterisk
at service
attack canvas
attack surface
attack vectors

attacks
auth module
authenticate_* directive
authentication
authentication directives
Authentication Header (AH) protocol
authentication management
authentication scripts
authenticity
authorization log files
authorization log options
AuthorizedKeysFile
authorizing access with polkit
auth_param directive
AuthType Basic directives
automated updates configuration
automatic private IP addressing (APIPA)
automatic Red Hat updates
automatic Ubuntu updates
avahi-daemon service
availability
AVC. *See* access vector cache
AVG anti-malware tools
AVG antivirus
AVG technologies
AWS. *See* Amazon Web Services

B

backport
BackTrack
Backup Domain Controller (BDC)
banner file
bare metal
bare metal virtualization
Base64
baseline backups
baseline Linux system updating
baseline network configuration
baseline system state identification
baseline updates
Bash. *See* Bourne Again Shell
bastion hosts
BDC. *See* Backup Domain Controller
Bell Labs
Berkeley Internet Name Domain (BIND)
Berkeley Standard Distribution (BSD)
Bernstein, Daniel J.
beta tests
/bin/false
binary editor
binary kernel

BIND. *See* Berkeley Internet Name Domain

binding

black-hat hacker

blacklists

blocking IP addresses

block-layer kernel

Bluetooth connections

Bluetooth hardware

/boot/ directory

boot loader

boot process

boot process security

Bootstrap Protocol (BOOTP)

botnet

Bourne Again Shell (Bash)

breach confirmation

broadcasts

browser control options

brute-force attack

BSD. *See* Berkeley Standard Distribution

bug reports

bugs

Bugzilla

bus options

bzcat commands

C

CA. *See* certificate authority

cache poisoning

caching-only DNS server

CAINE. *See* Computer Aided Investigative Environment

Canonical Landscape

Capabilities settings

CapabilityBoundingSet settings

CentOS

central processing unit (CPU)

Cerberus

CERT. *See* Computer Emergency Response Team

certificate authority (CA)

CGI scripts. *See* common gateway interface scripts

chage command

Challenge-Handshake Authentication Protocol (CHAP)

challenge-response authentication mechanism (CRAM)

Chaox

CHAP. *See* Challenge-Handshake Authentication Protocol

chcon commands

chgrouptab command

chkrootkit

chmod command

chown command

Chromium
chroot jail
C-I-A triad
CIDR. *See* Classless InterDomain Routing
CIFS. *See* Common Internet File System
Cinnamon
ck-history command
ck-launch-session command
ck-list-sessions command
Clam AntiVirus (ClamAV) system
Classless InterDomain Routing (CIDR)
cleartext access
cleartext communication
CLI. *See* command line interface
clients
closed source software
cloud computing
cloud management
cloud services
Cobbler project
command injection attack
command line
command line interface (CLI)
commercial security test tools
commercial update managers
common gateway interface (CGI) scripts
Common Internet File System (CIFS)
Common Unix Printing System (CUPS)
Common Vulnerabilities and Exposures (CVE) list
Community Ubuntu Documentation
compilers
complain mode
components testing
compromised data
compromised systems
Computer Aided Investigative Environment (CAINE)
Computer Emergency Response Team (CERT)
confAUTH_MECHANISMS directive
confidentiality
.config file
configuration categories for Linux kernel
configuration menu
console-based Security Level Configuration tool
consolidating remote logs
content addressable memory
Content Scramble System (CSS)
contexts
Core Rules Set (CRS)
corporate support
CPU. *See* central processing unit
crackers

CRAM. *See* challenge-response authentication mechanism
cron script
 cron service
 CRS. *See* Core Rules Set
 Cryptcat
 cryptographic application programming interface (API)
 cryptographic hash
 cryptographic keys
 cryptography
cryptsetup command
 CSS. *See* Content Scramble System
 CUPS. *See* Common Unix Printing System
 custom configuration
 custom kernels
 custom Linux kernels
 CVE list. *See* Common Vulnerabilities and Exposures list
 Cyrus

D

DAEMON_OPTIONS directive
 daemons program
 DAHDI. *See* Digium Asterisk Hardware Device Interface
 Daniel J. Bernstein's DNS (djbdns)
 DAS. *See* direct attached storage
 data center
 Data Security Standard (DSS)
dbus system
dd command
 DDoS. *See* distributed denial of service
dd_rescue command
 Debian Package Manager
debsums command
 debug-level messages
 debug log message priority
 default **nmap** scripts
 default permissions
 default rsyslog configuration
 default rules, change
 default runlevel
 default wireless hardware
 defense in depth approach
 DEFT. *See* Digital Evidence and Forensic Toolkit
 DEFT Linux
 delivery platform
 demilitarized zone (DMZ) firewall
 denial of service (DoS) attacks
 dependencies
 desktop applications
 desktop distribution
 desktop environments

desktop virtualization
desktops vs. servers
DetectionOnly directive
developers
development tools
device drivers in Linux kernel
DHCP. *See* Dynamic Host Configuration Protocol
dictionary attack
digest authentication
digital certificates
Digital Evidence and Forensic Toolkit (DEFT)
Digital Millennium Copyright Act (DMCA)
digital rights management (DRM)
Digital Signature Algorithm (DSA)
Digium
Digium Asterisk Hardware Device Interface (DAHDI)
DIR directive
direct attached storage (DAS)
directory
directory services
disaster recovery
discovery scripts for **nmap**
discretionary access controls
disk encryption subsystem (dm_crypt)
disk images
distributed denial of service (DDoS)
distribution-related applications
distribution release installation
distribution security
distribution-specific functional kernels
distribution-specific kernels security issues
distribution-specific Linux kernel
distribution upgrades
distributions
DistroWatch
djbdns. *See* Daniel J. Bernstein's DNS
DMCA. *See* Digital Millennium Copyright Act
dm_crypt. *See* disk encryption subsystem
DMZ firewall. *See* demilitarized zone firewall
dnl directive
DNS. *See* Domain Name System
dns proxy
DNSSEC. *See* Domain Name System Security Extensions
documentation
DocumentRoot directive
domain
domain controller options
domain master
domain members options
Domain Name System (DNS)
Domain Name System Security Extensions (DNSSEC)

DoS attacks. *See* denial of service attacks

Dovecot

dpkg commands

DRM. *See* digital rights management

DSA. *See* Digital Signature Algorithm

DSS. *See* Data Security Standard

dynamic address resolution protocol (ARP) inspection

dynamic data

Dynamic Host Configuration Protocol (DHCP)

dynamic list of processes

E

-e rc4-hmac

EAP. *See* Extensible Authentication Protocol

ecrypt-setup-private command

eCryptfs. *See* enterprise cryptographic filesystems

ecryptfs option

edquota command

education of users

electromagnetic output

electromagnetic radiation

ELF. *See* executable linkable format

Elgamal encryption scheme

Elgamal, Taher

e-mail access

e-mail clients

e-mail protocols

e-mail services

emerg log message priority

enable loadable module support

Encapsulating Security Payload (ESP) protocol

encrypted connection protocols

encrypted directories

encrypted files system

encrypted partitions and volumes

encrypting wireless networks

encryption

enterprise cryptographic filesystems (eCryptfs)

environment variable

EPEL. *See* Extra Packages for Enterprise Linux

err log message priority

error messages

ErrorDocument directive

error.log file in CUP

esp directive

ESP protocol. *See* Encapsulating Security Payload protocol

/etc/apt/sources.list configuration file

/etc/apt/sources.list file

/etc/group file

/etc/gshadow file

/etc/hosts.allow files
 /etc/hosts.deny files
/etc/login/defs file
/etc/passwd file
 /etc/racoon/racoon.conf file
/etc/rsyslog.conf file
 /etc/services file
/etc/shadow file
/etc/sudoers configuration file
 /etc/yum.conf configuration file
 /etc/yum.repos.d/directory
 /etc/yum/yum-cron.conf file
 ethernet
 evidence principles
 evince
 executable file formats/emulations
 executable files
 executable linkable format (ELF)
 Exim
 exploitation tools, Kali Linux
ext2 filesystem format
ext3 filesystem format
ext4 filesystem format
 extended internet super server
 extended service set identifier (ESSID)
 Extensible Authentication Protocol (EAP)
 Extra Packages for Enterprise Linux (EPEL)
 EXTRAVERSION directive

F

facilities
failovermethod
 FCCU Linux
 FDCC. *See* Federal Desktop Core Configuration
fdisk drive management tool
FEATURE directive
 Federal Desktop Core Configuration (FDCC)
 Fedora
 Fedora Core. *See* Fedora
 Fedora Core Linux
 Fedora distribution updates
 FHS. *See* filesystem hierarchy standard
 file and folder permissions
 file ownership
 file permissions
 file sharing
 File Transfer Protocol (FTP) servers
 files
 filesystem hierarchy standard (FHS)
 filesystems

firewall commands
 Firewall Configuration dialog box
 firewall-management tools
firewalld
firewalld command
 firewalls
 firmware drivers
fmem
 forensic live media
 forensics kit
 formats, filesystem
 forwarding-only DNS server
 fourth extended filesystem (**ext4**)
 Free Software Foundation (FSF)
 FreeBSD ports
freedesktop.org group
 FreeRADIUS configuration file
 freeware
 --frequent option
 front-end accelerator
 FSF. *See* Free Software Foundation
 FTP. *See* File Transfer Protocol
 functional bugs
 functional updates
 Fuzz attacks

G

GDM. *See* GNOME Display Manager
 General Public License (GPL)
 generic security services application program interface (GSSAPI)
 Gentoo Linux
getfacl command
 GID number. *See* group ID number
 GIMP. *See* GNU Image Manipulation Program
 GNOME. *See* GNU Network Object Model Environment
 GNOME Display Manager (GDM)
 GNU. *See* GNU's Not Unix
 GNU Image Manipulation Program (GIMP)
 GNU Network Object Model Environment (GNOME)
 GNU Privacy Guard (GPG) keys
 GNU's Not Unix (GNU)
 gold baseline
 Google Compute Engine
 Google compute instance
 Google's Chrome browser
gpasswd command
 GPG. *See* GNU Privacy Guard keys
gpg command
 GPG keys. *See* GNU Privacy Guard keys
 GPL. *See* General Public License
 GPTs. *See* GUID Partition Tables

Grand Unified Bootloader (GRUB)
 graphical desktop environment
 graphical login manager
 graphical user interface (GUI)
 gratis concept vs. libre concept
 gray-hat hacker
 Greenbone Security Desktop
 group administrators
Group directives
 group ID (GID) number
 group-management commands
groupadd
groupdel
groupmod
 groups
groups
grpquota option
 GRUB. *See* Grand Unified Bootloader
 gshadow file
 GSSAPI. *See* generic security services application program interface
gufw command
 GUI. *See* graphical user interface
 GUI Security Level Configuration tool
 GUID Partition Tables (GPTs)

H

hackers
 hard disk preservation
 hardening system
 hardware-assisted virtualization
 hardware certification
 hardware hacking tool
 hash algorithm
 hash function
 Helix Knoppix
 Helix live response
 hierarchy of administrative privileges
/home / directory
 home hobbyists updates
 honeypot
 host discovery
 host-to-host IPsec on Red Hat
 host-to-host IPsec on Ubuntu
HostKey directive
hosts allow directives
 Hping3 command-line packet analyzer
 .htaccess file
 HTCP. *See* Hypertext Caching Protocol
http_access directive
htdigest command

html directory
htpasswd command
HTTP. *See* Hypertext Transfer Protocol
http_access directive
httpd command
http_reply_access directive
HTTPS. *See* Hypertext Transfer Protocol, secure
HTTPPTunnel
Hydra tool
Hypertext Caching Protocol (HTCP)
Hypertext Transfer Protocol (HTTP)
Hypertext Transfer Protocol, secure (HTTPS)
hypervisor

|

IaaS. *See* Infrastructure as a Service
IANA. *See* Internet Assigned Numbers Authority
ICMP. *See* Internet Control Message Protocol
ICP. *See* Inter-Cache Protocol
icp_access directive
IDS. *See* Snort intrusion detection system
IIS. *See* Internet Information Server
IKE protocol. *See* Internet Key Exchange protocol
IMAP. *See* Internet Mail Access Protocol
IMAP. *See* Internet Message Access Protocol
IMAP4. *See* Internet Message Access Protocol version 4
incident response plan
include directive
independent software vendor (ISV) support
inetd
info log message priority
info priority
information gathering tool
Infrastructure as a Service (IaaS)
init program
initial installation screen
inline mode of Snort
inodes
insmod command
Institute for Security and Open Methodologies (ISECOM)
Integrated Services Digital Network (ISDN)
integrity checks
integrity scanners
Inter-Cache Protocol (ICP)
interface
interfaces directives
internal network firewall
International Telecommunication Union (ITU)
Internet access
Internet access of users

Internet access policies
 Internet Assigned Numbers Authority (IANA)
 Internet Control Message Protocol (ICMP)
 Internet DNS
 Internet-facing services
 Internet Information Server (IIS)
 Internet Key Exchange (IKE) protocol
 Internet Mail Access Protocol (IMAP)
 Internet Message Access Protocol (IMAP)
 Internet Message Access Protocol version 4 (IMAP4)
 Internet Protocol (IP) addresses
 Internet Protocol Security (IPSec)
 Internet Protocol version 4 (IPv4)
 Internet Protocol version 6 (IPv6)
 Internet Relay Chat (IRC) wars
 Internet service providers (ISPs)
 intrusive **nmap** command scripts
 IP addresses. *See* Internet Protocol addresses
 IP Source Guard
 ipchains
 IPSec. *See* Internet Protocol Security
 iptables
 iptables-based firewall
iptables command
 IPv4. *See* Internet Protocol version 4
 IPv6. *See* Internet Protocol version 6
 IRC. *See* Internet Relay Chat wars
 ISDN. *See* Integrated Services Digital Network
 ISECOM. *See* Institute for Security and Open Methodologies
 iSeries
 ISPs. *See* Internet service providers
 ISV. *See* independent software vendor support
 itanium
 ITU. *See* International Telecommunication Union
iwconfig command
iwlist command
iwspy command

J

Java Cryptography Extension (JCE)
 JBoss
 JCE. *See* Java Cryptography Extension
 John the Ripper tool
 journaled filesystem

K

Kali Linux
 Kaspersky Lab
 Kaspersky suite of anti-malware products
 K Desktop Environment (KDE)

KDC. *See* key distribution center
kdc_timesync
KDE. *See* K Desktop Environment
KDE Display Manager (KDM)
KeepAlive directives
KeepAliveTimeout directive
Kerberos configuration
Kerberos principal
Kerberos realm
Kerberos server
Kerberos Telnet
Kerberos ticket
kernel
kernel-based firewall
Kernel-based Virtual Machine (KVM) system
kernel command
kernel configuration
kernel customization process
kernel development
kernel-hacking options
kernel log services
kernel module
kernel numbering systems
kernel security
kernel source code
kernel-space tools
kernel updates
key distribution center (KDC)
key pairs
Kickstart-based installations
Knoppix
Knoppix live CD
Knoppix Security Tools Distribution (STD)
Konqueror Web browser
KPDF
krb524init command
Kroah-Hartman, Greg
Kubuntu distribution
KVM system. *See* Kernel-based Virtual Machine system

L

LAMP stack. *See* Linux/Apache/MySQL/P stack
LAN. *See* local area network
landscape
last command
--last option
Launchpad
laws
layered defense
LDAP-related Samba directives
LDAP server. *See* Lightweight Directory Access Protocol server

ldd command
legitimate access
libre concept vs. gratis concept
LibreOffice
licensing issues
Lightweight Directory Access Protocol (LDAP) server
LILO. *See* Linux Loader
Line Printer next generation (LPRng)
Linux
Linux/Apache/MySQL/P (LAMP) stack
Linux distributions
Linux extended partition
Linux file-encryption commands
Linux kernel
Linux Kernel Organization
Linux live CD/DVDs
Linux Loader (LILO)
Linux logs
Linux LVM partition
Linux Mint
Linux operating system updates
Linux partition
Linux patch management
Linux quotas
Linux raid auto partition
Linux security
Linux Security HOWTO
Linux security modules (LSMs)
Linux Standard Base (LSB)
Linux swap partition
Linux Terminal Server Project (LTSP)
Linux unified key setup (LUKS)
Linux user authentication database
Linux users
Listen directive
Listen**A**ddress directive
listening ports
live boot media
live bootable operating systems
live CD
local area network (LAN)
local authority features of polkit
local file and folder permissions
local logs
local **m**aster
local repositories
Local Samba servers
local services obscurity
Location directives
log access into the network
log file excerpt

log files
 log spoofed packets
LogFormat directive for Apache
 loggable authentication events
 logged-in users
 logging options
 login banners
login command
 login messages
 login scripts
 logs
 Logwatch
 long-term support (LTS)
 loopback interface
 LPRng. *See* Line Printer next generation
 LSB. *See* Linux Standard Base
lsmod command
 LSMs. *See* Linux security modules
lsof command
 LTS. *See* long-term support
 LTSP. *See* Linux Terminal Server Project
 LUKS. *See* Linux unified key setup

M

Mac OS X
 mail delivery agents (MDAs)
 mail retrieval agents (MRAs)
 mail servers
 mail submission agents (MSAs)
 mail transfer agents (MTAs)
 mail user agents (MUAs)
 mailing lists
 main repository
 malicious user
 malware detection
 malware scripts
 management commands
 managing security
 mandatory access controls
 Mandriva
 mangled ICMP messages
 Martian packets
 masquerading
 master browser
 master DNS server
 Mate
max_log_size directives
MaxAuthTries
MaxKeepAliveRequests directive
 MD5. *See* Message Digest 5
 MDAs. *See* mail delivery agents

mDNS protocol. *See* multicast Domain Name Service protocol
menuconfig command
 Message Digest 5 (MD5)
 meta packages
 Metasploit Framework
 Microsoft Active Directory network
 Microsoft networks
 Microsoft virtual server
 Minimal Install selection
 MINIX
 Mint
mkfs command
mkswap command
 mod_security module
 modular kernel
 monitoring local logs
 monolithic kernel
 Morris, Robert T.
 mount point
 Mozilla Firefox browser
 Mozilla Foundation
 Mozilla project applications
 MPMs. *See* Multi-Processing Modules
 MRAs. *See* mail retrieval agents
 MSAs. *See* mail submission agents
 MTAs. *See* mail transfer agents
 MUAs. *See* mail user agents
 multicast Domain Name Service (mDNS) protocol
 multi-core CPU processors
 Multiplexed Information and Computing Service (Multics)
 Multi-Processing Modules (MPMs)
 multitasking
 multiverse repository
 MySQL
mysqladmin command

N

name resolution
 name service switch file
 named services testing
 NAS. *See* network attached storage
 NAT. *See* network address translation
 National Security Agency (NSA)
 NATO. *See* North Atlantic Treaty Organization
nc command
 Nessus
 NetBIOS. *See* Network Basic Input/Output System
netbios name directive
netcat service
 netcat utility
 Netcraft

[netlogon] stanza
netstat -atun command
netstat command
network address translation (NAT)
network address translation (NAT) traversal protocol
network attached storage (NAS)
Network Basic Input/Output System (NetBIOS)
Network File System (NFS)
network IDS mode of Snort
Network Information Service (NIS)
network protection
network-related options
Network Security Toolkit (NST)
network services
network services obscurity
network switch
Network Time Protocol (NTP)
network-to-network IPSec on Red Hat
network-to-network IPSec on Ubuntu
network user verification tool
Networked Application Security best practices
networked file and folder permission
networked filesystems best practice
networking options submenu
networking-support menu
networks best practices
networks on encryption principles
new releases
newgrp commands
Nexpose
NFS. *See* Network File System
NFS Kerberos tickets
NFS version 4 (NFSv4) systems
nfsnobody account
NFSv4. *See* NFS version 4 systems
nftables
Nginx
NIS. *See* Network Information Service
nmap commands
nobody account
nonce
non–open source licenses
nonprivileged user
nonrepudiation
nonstandard connections
nonstandard port numbers
NORMAL directive
North Atlantic Treaty Organization (NATO)
notice priority
Novell’s ZENworks
NSA. *See* U.S. National Security Agency

NSEC3 key
 NSFNet
 NST. *See* Network Security Toolkit
 NTP. *See* Network Time Protocol
 null modem
nullok option

O

object mechanism
 obscure open ports
 obscure ports
 obscurity
 OCSP server. *See* Online Certificate Status Protocol server
oldconfig command
 Online Certificate Status Protocol (OCSP) server
 open network ports
 open port problem
 Open Source Assurance Program
 open source community
 open source principles
 open source proxy server
 open source security
 Open Source Security Testing Methodology Manual (OSSTMM)
 open source security-testing tools
 open source sendmail MTA
 open source server applications
 Open Source Telephony Project
 open source update managers
 Open Systems Interconnection (OSI) Reference Model
 Open Vulnerability Assessment System (OpenVAS)
 Open Web Application Security Project (OWASP)
 OpenLDAP
OpenOffice.org suite
openssl command
 OpenVAS. *See* Open Vulnerability Assessment System
 operating system
 operating system detection
/opt/ directory
optional control flag
Options directive
os level
 OSI Reference Model. *See* Open Systems Interconnection Reference Model
 OSSTMM. *See* Open Source Security Testing Methodology Manual
 overbooking
 OWASP. *See* Open Web Application Security Project
OwnerMode directive
 ownership

P

PaaS. *See* Platform as a Service

package-management systems
package manager
package verification
packages
packet filtering firewalls
packet-logger mode of Snort
pacman
page . log file in CUP
PAM configuration file
PAM modules
PAMs. *See* pluggable authentication modules
PAP. *See* Password Authentication Protocol
paravirtualization
paravirtualized shortcuts
Parker, Donn
Parkerian hexad
partition
partition types
passphrases
passwd command
password attacks tool
Password Authentication Protocol (PAP)
password complexity options
password cracker
password management
password module
password policies
passwords
patches
path directive
Payment Card Industry (PCI)
PBX. *See* private branch exchange
PCI. *See* Payment Card Industry
PDC. *See* Primary Domain Controller
PDF. *See* Portable Document Format
PDF readers
penetration-testing tools
performance audits
Perl
permissions
permissive mode
PERMS directive
personally identifiable information (PII)
PHP: Hypertext Preprocessor (PHP)
physical access
physical security
physical system
PII. *See* personally identifiable information
ping command
ping -f command
ping flood

ping storms
Platform as a Service (PaaS)
platform-level virtual machines
plug-ins sampling
pluggable authentication modules (PAMs)
Point-to-Point Tunneling Protocol (PPTP) services
policies
PolicyKit
polkit
POP. *See* Post Office Protocol
POP3. *See* Post Office Protocol version 3 server
Port 443
port directive
port numbers for services
port-scanning software
port specification and scan order
Portable Document Format (PDF)
portage system
ports
possession/control
Post Office Protocol (POP)
Post Office Protocol version 3 (POP3) server
Postfix
Postfix mail server
PostgreSQL
power management
power users updates
PPTP. *See* Point-to-Point Tunneling Protocol services
preferred master
pre-shared key (PSK)
prevention
Primary Domain Controller (PDC)
printer administrators
printing options
private branch exchange (PBX)
private key cryptography
private/public key
PrivateTmp settings
privileged user
privileges
Pro File Transfer Protocol daemon (ProFTPD)
problem sharing
/proc directory filesystem
/proc directory for forensic analysis
process tree
/proc/kcore file
Prod directive
production releases of kernel
[profiles] stanz
ProFTP server
ProFTPD. *See* Pro File Transfer Protocol daemon

Project Athena
ProtectSystem
 protocols
ps command
pSeries
 PSK. *See* pre-shared key
 PSTN. *See* public switched telephone network
 public key cryptography
 public/private key pair encryption
 public switched telephone network (PSTN)
 pulling updates
 Pure File Transfer Protocol daemon (Pure-FTPd)
 pushing updates
 Python

Q

Qmail
[qualifiers] stanza
 quota configuration process
 quota grace periods
 quota management
 quota reports
quotacheck command
 quotas

R

r-services
 RADIUS. *See* Remote Authentication Dial In User Service
 RADIUS configuration files
 RAID. *See* redundant array of independent disks
 rainbow tables
 RainbowCrack tool
 RAM. *See* random access memory
 RAM disk
 random access memory (RAM)
 Raymond, Eric
rc4
 RDP. *See* Remote Desktop Protocol
 README file
 read-only filesystems
 read-only mount points
 read-only operating systems
 Real-time Transport Protocol (RTP)
 Really Simple Syndication (RSS)
 rebuild of RHEL
 recovery tools
 recursive query
 Red Flag Linux
 Red Hat
 Red Hat AIDE configuration

Red Hat-based firewall
Red Hat baseline
Red Hat Bugzilla database
Red Hat configuration
Red Hat distributions
Red Hat Enterprise Linux (RHEL)
Red Hat kernel
Red Hat log configuration
Red Hat Network (RHN)
Red Hat Package Management system
Red Hat Package Manager (RPM)
Red Hat Proxy server
Red Hat repositories
Red Hat Satellite server
Red Hat security advisories (RHSAs)
Red Hat Security Level Configuration tool
Red Hat Spacewalk
Red Hat support options
Red Hat's authentication configuration tool
redundancy
redundant array of independent disks (RAID)
registered ports
regular internet super server
regular permissions
regular user updates
reiserfs filesystem
remote access
remote administration of CUPS servers
Remote Authentication Dial In User Service (RADIUS)
remote desktop configuration
Remote Desktop Protocol (RDP)
remote logs
remote procedure call (RPC)
remote repositories
replacement of compromised systems
replacement systems
reporting tools, Kali Linux
repositories
required control flag
requisite control flag
response times
restart command
restricted repository
reverse engineering tools, Kali Linux
reverse path filtering
RHEL. *See* Red Hat Enterprise Linux
RHN. *See* Red Hat Network
rhn_register command
RHSAs. *See* Red Hat security advisories
Ritchie, Dennis
Rivest, Shamir, and Adleman (RSA)

rmmod command
 Robbins, Daniel
root
 root administrative password
 root nameserver
 root user
 rootkit
 Rootkit Hunter
 routing tables
 RPC. *See* remote procedure call
 RPM. *See* Red Hat Package Manager
rpm commands
 RSA. *See* Rivest, Shamir, and Adleman
 RSS. *See* Really Simple Syndication
rsync command
 rsyslog configuration
 rsyslog global directives
 rsyslog modules
 rsyslog package
 rsyslog service
 RTP. *See* Real-time Transport Protocol
 runlevels
 runtime information collection
rw

S

SAINT. *See* System Administrator's Integrated Network Tool
 salt
 Samba
 Samba logs
 SAN. *See* storage area network
 SANS. *See* SysAdmin Audit Network Security
 SANS Investigative Forensics Toolkit (SIFT)
 SATAN. *See* Security Administrator Tool for Analyzing Networks
 /sbin/nologin
 scp. *See* secure copy client
scp command
 scripting languages
 scripts associated with **nmap**
 SDK. *See* software development kit
secon
 second extended filesystem (**ext2**)
 secure alternatives
 secure backup
 secure baseline best practices
 secure copy client (scp)
 Secure Hash Algorithm (SHA)
 secure kernel
 secure limits
 secure portmapper

Secure Shell (SSH)
secure shell passphrase
Secure Sockets Layer (SSL)
secure SSH copying
secure Telnet servers
secure Web sites
secure Windows server
securing groups of users
securing modem connections
securing NFS
securing remote logs
security
security administrator
Security Administrator Tool for Analyzing Networks (SATAN)
security baseline configuration
security breaches
security device
Security Enhanced Linux (SELinux)
security issues
security operations management
security options
security policies
security-related Apache directives
security-related MySQL directives
security-related Squid directives
security reports
security risks
security testing
security-testing tools
security threats
security updates
self-signed certificate
SELinux. *See* Security Enhanced Linux
SELinux Administration tool
SELinux boolean settings
SELINUX directives
SELinux troubleshooter
sendmail (commercial)
sendmail e-mail service
serial ports
server
server applications
server distribution
Server Message Block (SMB)
server room
server.csr file
server.key files
ServerTokens directive
servers vs. desktops
service applications
service command

service developers
service management
service process security
service-specific bug reports
service-specific gold baseline systems
service version detection
services
services administrative privileges
Session Initiation Protocol (SIP)
session management
session module
sesstatus command
set group ID (SGID) bit
set user ID (SUID) bit
setfacl command
[**settings**] stanza
sg command
SGID bit. *See* set group ID bit
SHA. *See* Secure Hash Algorithm
shadow password suite
shared networking services
shared NFS directories
shared printers
Shares Definitions
simple baseline configuration
Simple Mail Transfer Protocol (SMTP)
simple services
simplified mandatory access control kernel support (SMACK)
simulated hardware
single sign-on
SIP. *See* Session Initiation Protocol
Size directive
Slackware
slave DNS server
SMACK. *See* simplified mandatory access control kernel support
SMB. *See* Server Message Block
smb.conf configuration file
smbpasswd
SMP. *See* symmetrical multiprocessing
SMTP. *See* Simple Mail Transfer Protocol
smurf attack
snapshots
sniffer mode
sniffing/spoofing tools
Snort
Snort intrusion detection system (IDS)
socket
soft link
software
software bundles
software development

software development kit (SDK)
software RAID array
Solaris
source-based distributions
source code
source code installation
source-routed packets
source routing
Spacewalk
Spam Assassin
special groups
special permissions
split DNS
spoofed broadcast
spoofed ICMP messages
spoofing
spyware
Squid
Squid configuration
/srv/ directory
SSH. *See* Secure Shell
SSH clients and servers
SSH daemon (sshd)
SSH logins
SSH remote access methods
SSH server
SSH service
SSH services attacks
sshd. *See* SSH daemon
SSL. *See* Secure Sockets Layer
stable distribution
Stallman, Richard
standalone server options
standard rsyslog configuration file
Stanford University
star command
status command
STD. *See* Knoppix Security Tools Distribution
sticky bit
sticky-bit permissions
stock kernel
stock Linux kernel
storage area network (SAN)
strace command
stress testing tool
StrictModes
su command
subject mechanism
subversion repository
sudo command
sufficient control flag

SUID bit. *See* set user ID bit
 Sun Microsystems
 SunOS
 support issues
 SUSE
 SUSE Linux
 symlink
 symmetrical multiprocessing (SMP)
 SYN flood
sync
 syntax checks
 SysAdmin Audit Network Security (SANS)
 syslog configuration rule
SyslogFacility
sysstat package
 System Administrator's Integrated Network Tool (SAINT)
 system and kernel log services
 system authentication logs
 system integrity recalibration
 system logs
 system monitoring
 system security
 systemctl
 systemd
SystemGroup directive
 SysV Init
 SysVR4

T

table switch
 Tanenbaum, Andrew
tar command
 tarballs
 tasksel browser
tasksel command
 TCP/IP. *See* Transmission Control Protocol/ Internet Protocol
 TCP Wrappers
tdbsam
 telephone modems
 Telnet
telnet command
 TEMPEST study
testparm command
 text-configuration file
 TFTP. *See* Trivial File Transfer Protocol
 /tftpboot/ directory
 TGS. *See* ticket-granting server
 TGT. *See* ticket-granting ticket
 The Sleuth Kit (TSK)
 third extended filesystem (**ext3**)

third-party applications management
 Thompson, Ken
 threat vectors
 ticket-granting server (TGS)
 ticket-granting ticket (TGT)
 time service protection
 TLB. *See* translation lookaside buffer
 TLS. *See* Transport Layer Security
 /tmp/ directory
 Tomcat
 TOMOYO
top command
 top-level installation menu
 Torvalds, Linus
 tracking access
 traditional proxy server
 traffic between network
 Transition SIGnature (TSIG) key
 translation lookaside buffer (TLB)
 Transmission Control Protocol/Internet Protocol (TCP/IP)
 Transport Layer Security (TLS)
 Tripwire
 Tripwire intrusion detection system
 trivial bit
 Trivial File Transfer Protocol (TFTP)
 Trojan horse
 trojan_horse
 TRUST_AUTH_MECH directive
 Trusted Services window
 TSIG. *See* Transition SIGnature key
 TSK. *See* The Sleuth Kit
 Turbolinux
 type 1 hypervisors
 type 2 hypervisors

U

Ubuntu
 Ubuntu AIDE
 Ubuntu-based network-to-network IPsec connections
 Ubuntu baseline
 Ubuntu/Debian packages verification
 Ubuntu distributions
 Ubuntu kernel
 Ubuntu kernel source code
 Ubuntu Launchpad
 Ubuntu Linux kernel packages
 Ubuntu log configuration
 Ubuntu security notices (USNs)
 Ubuntu Server Edition
 Ubuntu software options
 Ubuntu systems

UCE. *See* unsolicited commercial e-mail

UDP. *See* User Datagram Protocol

UID number. *See* user ID number

umask command

uname -m command

uname -r command

unauthorized user

unencrypted protocols

Unics

Unity desktop environment

universally unique identifier (UUID)

universe repository

Unix systems

unprivileged user

unsolicited commercial e-mail (UCE)

unstable distributions

update administration

update-rc.d command

updated kernel installation

updates

upgrades

upstart system

U.S. National Security Agency (NSA)

USE variable

user applications

user-authentication databases

User Datagram Protocol (UDP)

User directives

user home directories

user ID (UID) number

user identification

user installation

user-management commands

user private group scheme

user privileges

user security

user-space tools

useradd

userdel

user_list

usermod command

users-admin command

USNs. *See* Ubuntu security notices

usrquota option

utility

UUID. *See* universally unique identifier

V

Van Eck phreaking

/var / directory

version category of scripts
 version numbers of services
VERSIONID directive
 very secure File Transfer Protocol daemon (vsftpd)
 very secure File Transfer Protocol (vsftp) service
 Vi iMproved (Vim) editor
 vigilance
 Vim. *See* Vi iMproved editor
virsh command
 virtual guests
 virtual hosts
 Virtual LANs (VLANs)
 virtual machine monitor
 virtual machines
 virtual memory
 Virtual Network Computing (VNC) server
 virtual physical security
 virtual private network (VPN)
 VirtualBox open source edition
 virtualization
 virtualized hardware
 virus
 VLAN ACLs
 VLANs. *See* Virtual LANs
VMware systems
 VNC server. *See* Virtual Network Computing server
 Voice over IP (VoIP)
 volumes
 VPN. *See* virtual private network
 vsftp configuration files
 vsftp configuration option
 vsftp service. *See* very secure File Transfer Protocol service
 vsftpd. *See* very secure File Transfer Protocol daemon
vsftpd.conf configuration file
 vulnerability analysis tool
 vulnerability category of scripts
 vulnerability scanners for Linux

W

w command
 W3 Techs Web technology surveys
 WAF. *See* Web application firewall
 war dialers
warn priority
 Washington University File Transfer Protocol daemon (WU-FTPD)
 Web application firewall (WAF)
 Web applications tool
 Web-based administrative tool
 Web browsers
 Web proxy server
 Web site accessibility

Web site configuration
 Web sites
 well-known ports
WEP. See wired equivalent privacy
 white-hat hacker
 whitelists
who command
 Wi-Fi Protected Access (WPA)
 Wi-Fi Protected Access, version 2 (WPA2)
 wide area network (WAN)
WIDS. See wireless intrusion detection systems
 wikis
 Winbind service
 Windows
 Windows Internet Name Service (WINS) servers
 Windows network access
Wine Is Not an Emulator (WINE)
wins proxy
wins server
 WINS servers. *See Windows Internet Name Service servers*
wins support
 wired equivalent privacy (WEP)
 wireless attacks tool
 wireless hardware
 wireless intrusion detection systems (WIDS)
 wireless networking
 wireless receiver-transmitter (WRT) software
 Wireshark
 worm
WPA. See Wi-Fi Protected Access
 wrappers
WRT software. See wireless receiver-transmitter software
WU-FTPD. See Washington University File Transfer Protocol daemon

X

X Display Manager (XDM)
 X11 protocol
XDM. See X Display Manager
 Xen
 Xen kernel
 Xen virtual machine monitor
 Xfce desktop environment
 XFree86
xfs filesystem
 xhost
 xinetd
 Xubuntu distribution

Y

Yellowdog Linux

Yellowdog Updater (yup)
Yellowdog Updater, Modified (yum)
yum command

Z

zcat command

ZENworks

zone files

zone updates