PEARSON IT CERTIFICATION

Save 10% on Exam Voucher
See Inside

Practice Tests

Flash Cards

Review Exercises

Study Planner

Cert Guide

Advance your IT career with hands-on learning

CompTIA®

Linux+

XK0-005

ROSS BRUNSON

FREE SAMPLE CHAPTER

CompTIA® Linux+ XK0-005 Cert Guide

Companion Website and Pearson Test Prep Access Code

Access interactive study tools on this book's companion website, including practice test software, review exercises, Key Term flash card application, a study planner, and more!

To access the companion website, simply follow these steps:

1. Go to **www.pearsonitcertification.com/register**.
2. Enter the **print book ISBN: 9780137866885**.
3. Answer the security question to validate your purchase.
4. Go to your account page.
5. Click on the **Registered Products** tab.
6. Under the book listing, click on the **Access Bonus Content** link.

When you register your book, your Pearson Test Prep practice test access code will automatically be populated with the book listing under the Registered Products tab. You will need this code to access the practice test that comes with this book. You can redeem the code at **PearsonTestPrep.com**. Simply choose Pearson IT Certification as your product group and log in to the site with the same credentials you used to register your book. Click the **Activate New Product** button and enter the access code. More detailed instructions on how to redeem your access code for both the online and desktop versions can be found on the companion website.

If you have any issues accessing the companion website or obtaining your Pearson Test Prep practice test access code, you can contact our support team by going to **pearsonitp.echelp.org**.

This page intentionally left blank

CompTIA® Linux+ XK0-005 Cert Guide

Ross Brunson

Pearson
Hoboken, New Jersey

CompTIA® Linux+ XK0-005 Cert Guide

ISBN-13: 978-0-13-786688-5

ISBN-10: 0-13-786688-7

Library of Congress Control Number: 2023913510

$PrintCode

Vice President, IT Professional
Mark Taub

Director, ITP Product Line Management
Brett Bartow

Executive Editor
Nancy Davis

Development Editor
Ellie Bru

Managing Editor
Sandra Schroeder

Senior Project Editor
Mandie Frank

Copy Editor
Bill McManus

Indexer
Ken Johnson

Proofreader
Jennifer Hinchliffe

Technical Editor
Casey Boyles

Publishing Coordinator
Cindy Teeters

Designer
Chuti Prasertsith

Compositor
codeMantra

Pearson's Commitment to Diversity, Equity, and Inclusion

Pearson is dedicated to creating bias-free content that reflects the diversity of all learners. We embrace the many dimensions of diversity, including but not limited to race, ethnicity, gender, socioeconomic status, ability, age, sexual orientation, and religious or political beliefs.

Education is a powerful force for equity and change in our world. It has the potential to deliver opportunities that improve lives and enable economic mobility. As we work with authors to create content for every product and service, we acknowledge our responsibility to demonstrate inclusivity and incorporate diverse scholarship so that everyone can achieve their potential through learning. As the world's leading learning company, we have a duty to help drive change and live up to our purpose to help more people create a better life for themselves and to create a better world.

Our ambition is to purposefully contribute to a world where

- Everyone has an equitable and lifelong opportunity to succeed through learning
- Our educational products and services are inclusive and represent the rich diversity of learners
- Our educational content accurately reflects the histories and experiences of the learners we serve
- Our educational content prompts deeper discussions with learners and motivates them to expand their own learning (and worldview)

While we work hard to present unbiased content, we want to hear from you about any concerns or needs with this Pearson product so that we can investigate and address them.

Please contact us with concerns about any potential bias at https://www.pearson.com/report-bias.html.

This page intentionally left blank

Contents at a Glance

Table of Contents

About the Author

Ross Brunson has more than 30 years of experience as a Linux and open-source trainer, training manager, and certification architect, and is the author of the now-classic *LPIC-1 Exam Cram 2*, several iterations of the *CompTIA Linux+ Cert Guide*, and dozens of technical courses for major organizations.

Ross is currently the Education Architect at Grafana Labs (www.grafana.com), where he focuses on building a learning framework and training offerings that help employees and customers make the best use of Grafana to observe, troubleshoot, and maintain their environments.

Previously, Ross was a Senior Technical Training Engineer for NGINX, where he completely redid the Fundamentals learning track, authored a number of Getting Started guides, and taught a number of customer engagements to help new NGINX customers take full advantage of the platform.

Before NGINX, Ross enjoyed a few years at Linux Academy/A Cloud Guru where as a Senior Training Architect, he authored the SUSE Certified Administrator and Engineer courses, did the Red Hat Certified System Administrator Labs, created many additional courses on systemd, VIM and the screen command, and wrote and reviewed way too many exam questions.

Ross has also put in a tour of duty as the Certification Architect at SUSE, where he helped redesign and modernize the entire certification program. He has also spent five years as the Director of Member Services for the Linux Professional Institute, where he contributed to placing several LPI courses into the Cisco Networking Academy, conducted dozens of train-the-trainer sessions, and provided sales enablement support for the worldwide Master Affiliate network, spanning more than 100 countries and nearly a million certified professionals.

Ross holds a number of key IT certifications and is author of several successful technical books and dozens of technical courses for major organizations (including the first U.S. LPI Certification Bootcamps). He is skilled at both contributing to and building community around IT products.

Ross lives in Paradise Valley, Montana, with his family and enjoys traveling far and wide, participating in hiking, winter sports, photography, and playing the drums with great vigor (although not everyone around him appreciates it).

Dedication

My heartfelt thanks to all of my mentors and friends who have helped me get where I am: Andres Fortino, Arnold Villeneuve, Ken Haug, Ted Jordan, Edward Denzler, and many more. I am eternally grateful for the love and support of my wife and daughter, who understand what it means when "daddy is writing" and still love me anyway.

I also want to shout out to all our previous edition readers who made this book likely and possible. I love the emails and photos of you all and your certifications; it really makes a difference when we get something that lets us know we are somehow making even a tiny difference in someone's career and life.

I want to hear from YOU. Let me know what you liked, what I can improve, and how you're doing. Please send pics of you and your study tools, you and your certification, and so on...

—Ross E. Brunson, July 2023

Acknowledgments

This book is a result of the concerted efforts of many dedicated people, without whom this book would not be a reality. I would like to thank the technical reviewer, Casey Boyles, whose efforts and patience made this a better book for all to use, and to Chris Cleveland, who helped me navigate the adjustments to new CompTIA Linux+ versions over the years.

Much thanks to William (Bo) Rothwell for the courseware and writing MACHINE that he is, and for being a great author and technical editor over these many years—I couldn't have done it without you, buddy!

Thanks also to Nancy Davis, Executive Editor, for her help and continuous support during the development of this book. I wish to also express my appreciation to Mary Beth Ray, executive editor at Pearson/Cisco Press (retired), for her confidence in me throughout years of working on book projects.

Much thanks to Ellie Bru for both her superb editorial skills and acumen, but especially her good humor and geek-wrangling skills; it is a pleasure to work with her on every book!

In addition, many thanks to Dr. James Stanger for being such a great supporter of the world of Linux and open source. He's a good friend and a hugely relevant person in the world of getting our customers and attendees the skills they need!

It has been a huge undertaking to pull together all the pieces of this project. It is due to the dedication of those mentioned above that this book is not only large in scope but high in quality. It is my sincerest hope that our combined efforts will help you, the readers and users of this book, achieve your goals in an IT career.

About the Technical Reviewer

Casey Boyles started working in the IT field more than 28 years ago and quickly began to work with distributed application and database development. Casey later moved on to technical training and development; he specializes in full stack Internet application development, database architecture, and systems security. Casey typically spends his time smoking cigars while "reading stuff and writing stuff."

We Want to Hear from You!

As the reader of this book, *you* are our most important critic and commentator. We value your opinion and want to know what we're doing right, what we could do better, what areas you'd like to see us publish in, and any other words of wisdom you're willing to pass our way.

We welcome your comments. You can email or write to let us know what you did or didn't like about this book—as well as what we can do to make our books better.

Please note that we cannot help you with technical problems related to the topic of this book.

When you write, please be sure to include this book's title and author as well as your name and email address. We will carefully review your comments and share them with the author and editors who worked on the book.

Email: community@informit.com

Reader Services

Register your copy of *CompTIA Linux+ XK0-005 Cert Guide* at www.pearsonitcertification.com for convenient access to downloads, updates, and corrections as they become available. To start the registration process, go to www.pearsonitcertification.com/register and log in or create an account*. Enter the product ISBN 9780137866885 and click Submit. When the process is complete, you will find any available bonus content under Registered Products.

*Be sure to check the box that you would like to hear from us to receive exclusive discounts on future editions of this product.

Introduction

In mid-2022, CompTIA released a new version of the Linux+ certification exam, labeled XK0-005. To throw a monkey wrench is to goof up or confuse or sabotage. The gears of other content authors may be messed up by this change, but I remain unaffected.

In particular, the new exam version has more DevOps-related and cloud-specific topic areas, allocates more space to Git and containers, and adds the revised Troubleshooting domain, which covers significant space on the exam and comprises approximately 28% of the scoring.

Most of the Linux+ exam will be multiple choice, much like the previous exams. However, you should also be prepared for a handful of scenario questions in which you will be asked to answer some questions based on a particular situation. In addition, you'll encounter some simulation questions, where you're running what appears to be a command-line terminal and you have to answer the question by actually typing the right commands and so forth. (Please note, you can use the **commandname help** option for *all* of these simulation questions, which will really help you puzzle out what the questions require!)

Use this book as a reference to all of the key exam-testable topics. This book provides an excellent roadmap on your journey to learning Linux and passing the Linux+ certification exam.

Study hard and study well. Pore over the exam objectives, and if you don't know something, I guarantee that you will see it on the exam, so make sure to locate the topic or term in this book's TOC or index and read the relevant material.

Good luck!

—Ross E. Brunson, July 5, 2023

Goals and Methods

The number-one goal of this book is a simple one: to help you pass the CompTIA Linux+ XK0-005 certification exam.

Because the CompTIA Linux+ certification exam now stresses problem-solving abilities and reasoning more than memorization of terms and facts, my goal is to help you master and understand the required objectives for the exam.

To aid you in mastering and understanding the Linux+ certification exam objectives, this book uses the following methods:

- **Opening topics list:** The list at the beginning of each chapter defines the topics to be covered in the chapter, followed by identification of the corresponding CompTIA Linux+ objective.
- **Foundation Topics:** The body of the chapter explains the topics from both hands-on and theory-based standpoints, including in-depth descriptions, tables, and figures that help you build your knowledge so that you can pass the Linux+ exam. The chapters are broken down into several topics each.
- **Key Topics:** Key Topics icons indicate important figures, tables, and lists of information that you should know for the exam. They are interspersed throughout the chapter and are listed in table format at the end of the chapter.
- **Key Terms:** Key terms without definitions are listed at the end of each chapter. Write down the definition of each term and check your work against the key terms in the glossary.
- **Review Questions:** These quizzes and answers with explanations are meant to gauge your knowledge of the subjects covered in the chapter. If an answer to a question doesn't come readily to you, be sure to review that portion of the chapter.
- **Practice Exams:** The practice exams are included in the Pearson Test Prep practice test software. These exams test your knowledge and skills in a realistic testing environment. Take them after you have read through the entire book. Master one, then move on to the next.

The Linux+ Domains and Objectives

The Linux+ XK0-005 exam consists of the following domains and objectives:

1.0 System Management (32% of the exam)

1.1 Summarize Linux fundamentals

1.2 Given a scenario, manage files and directories

1.3 Given a scenario, configure and manage storage using the appropriate tools

1.4 Given a scenario, configure and use the appropriate processes and services

1.5 Given a scenario, use the appropriate networking tools or configuration files

1.6 Given a scenario, build and install software

1.7 Given a scenario, manage software configurations

2.0 Security (21% of the exam)

2.1 Summarize the purpose and use of security best practices in a Linux environment

2.2 Given a scenario, implement identity management

2.3 Given a scenario, implement and configure firewalls

2.4 Given a scenario, configure and execute remote connectivity for system management

2.5 Given a scenario, apply the appropriate access controls

3.0 Scripting, Containers, and Automation (19% of the exam)

3.1 Given a scenario, create simple shell scripts to automate common tasks

3.2 Given a scenario, perform basic container operations

3.3 Given a scenario, perform basic version control using Git

3.4 Summarize common infrastructure as code technologies

3.5 Summarize container, cloud, and orchestration concepts

4.0 Troubleshooting (28% of the exam)

4.1 Given a scenario, analyze and troubleshoot storage issues

4.2 Given a scenario, analyze and troubleshoot network resource issues

4.3 Given a scenario, analyze and troubleshoot central processing unit (CPU) and memory issues

4.4 Given a scenario, analyze and troubleshoot user access and file permissions

4.5 Given a scenario, use systemd to diagnose and resolve common problems with a Linux system.

Be sure to visit CompTIA's web page at https://certification.comptia.org/certifications/linux to ensure that you have the latest information for the CompTIA Linux+ exam.

How This Book Maps to the Exam Objectives

All exam objectives are covered in this book and each chapter is devoted to a specific exam objective. But, in the interest of presenting a logical learning path, the order of the content in each chapter does not exactly match the order of the topics listed within the corresponding objective. To help you focus on the exam objectives for

which you might need some additional learning and preparation, this table shows you which chapters cover the various exam objectives:

Chapter	Exam Objective(s) Covered
Chapter 1, "Understanding Linux Fundamentals"	1.1
Chapter 2, "Managing Files and Directories"	1.2
Chapter 3, "Configuring and Managing Storage"	1.3
Chapter 4, "Managing Processes and Services"	1.4
Chapter 5, "Using Network Tools and Configuration Files"	1.5
Chapter 6, "Building and Installing Software"	1.6
Chapter 7, "Managing Software Configurations"	1.7
Chapter 8, "Understanding Linux Security Best Practices"	2.1
Chapter 9, "Implementing Identity Management"	2.2
Chapter 10, "Implementing and Configuring Firewalls"	2.3
Chapter 11, "Using Remote Connectivity for System Management"	2.4
Chapter 12, "Understanding and Applying Access Controls"	2.5
Chapter 13, "Automating Tasks via Shell Scripting"	3.1
Chapter 14, "Performing Basic Container Operations"	3.2
Chapter 15, "Performing Basic Version Control Using Git"	3.3
Chapter 16, "Understanding Infrastructure as Code"	3.4
Chapter 17, "Understanding Containers, Cloud, and Orchestration"	3.5
Chapter 18, "Analyzing and Troubleshooting Storage Issues"	4.1
Chapter 19, "Analyzing and Troubleshooting Network Resource Issues"	4.2
Chapter 20, "Analyzing and Troubleshooting CPU and Memory Issues"	4.3
Chapter 21, "Analyzing and Troubleshooting User and File Permissions"	4.4
Chapter 22, "Analyzing and Troubleshooting Common Problems Using Systemd"	4.5

Book Features

To help you customize your study time using this book, the core chapters have several features that help you make the best use of your time:

- **Foundation Topics:** These core sections of each chapter explain the concepts that are important to the chapter.
- **Exam Preparation Tasks:** This section lists a series of study activities that you should do at the end of the chapter:
  - **Review All Key Topics:** The Key Topic icon appears next to the most important items in the "Foundation Topics" section of the chapter. The "Review All Key Topics" activity lists the key topics from the chapter, along with their page numbers. Although the contents of the entire chapter could be on the exam, you should definitely know the information listed in each key topic, so be sure to review them.
  - **Define Key Terms:** Although the Linux+ exam is unlikely to ask an open-ended question such as "Define this term," the exam does require that you learn and know a lot of industry-related terminology. This section lists the most important terms from the chapter, asking you to write a short definition and compare your definition to the glossary definition at the end of the book.
  - **Review Questions:** Confirm that you understand the content that is covered in the chapter by answering these questions and reading the answer explanations.
- **Web-Based Practice Exams:** The companion website includes the Pearson Cert IT certification test engine, which allows you to take practice exams. Use it to prepare with a sample exam and to pinpoint topics where you need more study.

What's New?

If you are used to the objectives of the older Linux+ exam and the content of the previous version of this book, you should read the following which describes how both the exam objectives and the layout of this book have changed.

For more information about how the CompTIA Linux+ certification can help your career or to download the latest official objectives, access CompTIA's Linux+ web page at https://www.comptia.org/certifications/linux.

As the Linux+ objectives are now presented in an order that makes sense from a learning perspective, this book is patterned with each chapter taking on an objective topic in its entirety. (Thanks, CompTIA!)

However, as a long-time technical instructor who likes things to make sense even within a chapter, I have taken some liberties with the in-chapter order of each objective's contents, to ensure that everything flows nicely as you read and study so that you truly understand the subtopics.

You might be wondering how different the current Linux+ exam compares to the previous version.

As mentioned earlier in this Introduction, the main changes are

- An expanded focus on DevOps and cloud topics
- The new Troubleshooting section

Finally, as with most other CompTIA exams, you can expect a handful of scenario questions. In many cases, you will be asked to configure or manage a system using several steps or to describe a collection of Linux features and how they relate to each other. Note this that is not new, but worth mentioning.

In addition, look for the newer simulation questions, which are just like a Linux terminal session, and if you don't already know how to do these, you will by the end of this book!

The rest of the questions will be multiple-choice questions that require you to choose one, choose two, choose three, or choose all that apply.

Who Should Read This Book?

The CompTIA Linux+ certification exam will verify the successful candidate has the knowledge and skills required to configure, manage, operate, and troubleshoot Linux on-premises and cloud-based server environments, while using security best practices, scripting, containerization, and automation.

The level of knowledge and skills expected of the examinee is equivalent to at least 12 months of hands-on experience working with Linux servers in a junior Linux support engineer or junior cloud/DevOps support engineer job role. Additionally, CompTIA does specifically mention that having the experience of passing the A+, Network+, and Server+ exams is considered a recommended prerequisite for taking the Linux+ exam.

This book is for you if you are attempting to attain a position in the IT field or if you want to keep your skills sharp or perhaps retain your job if your company mandates that you take the latest Linux+ exam.

Strategies for Exam Preparation

Strategies for exam preparation will vary depending on your existing skills, knowledge, and equipment available. The ideal exam preparation would consist of building a few virtual machines from scratch and installing and configuring the operating systems covered.

The next best step you can take is to read through the chapters in this book, jotting notes down with key concepts or configurations on a notepad. Each chapter contains a quiz near the end of the chapter that you can use to test your knowledge of the chapter's topics.

Try *all* of the commands you see, look through the configuration files, experiment on your virtual machines, and use the snapshot and rollback feature that is on every virtualization software's toolbar these days—it'll make for a much more pleasant experience when you can try out commands and then revert to a previous snapshot.

After you have read through the book, take a look at the current exam objectives for the CompTIA Linux+ certification exam, listed at https://www.comptia.org/certifications/linux. If there are any areas shown in the certification exam outline that you would still like to study, find the appropriate sections in this book and review them.

When you feel confident in your skills, attempt the practice exams included on this book's companion website. As you work through a practice exam, note the areas where you lack confidence and review those concepts or configurations in the book. After you have reviewed the areas, work through the practice exam a second time and rate your skills. Keep in mind that the more you work through the practice exams, the more familiar the questions will become.

After you have worked through each practice exam a second time and feel confident with your skills, schedule the real CompTIA Linux+ (XK0-005) exam through Pearson VUE (https://home.pearsonvue.com/). To prevent the information from evaporating out of your mind, you should typically take the exam within a week of when you consider yourself ready to take the exam.

My usual advice for all my certification classes and courses stands: Drink a liter of water and have a nice ripe banana before you go take the exam. The exam is a long one, and you need your brain to function well; the water will help keep that computer in between your ears humming along, and the nutrients (particularly the niacin) in the banana will help you concentrate.

I can't tell you how many pictures of readers I have been sent with their liter of water, a banana, a newly achieved certification, and a BIG SMILE!

In fact, look me up on LinkedIn, at https://www.linkedin.com/in/rossbrunson/, and message me with your picture of your water, banana, and certification! I'll be sure to include you in any giveaways of book copies and so forth.

Companion Website

Register this book to get access to the Pearson IT certification test engine and other study materials, as well as additional bonus content. Check this site regularly for new and updated postings written by the author that provide further insight into the more troublesome topics on the exam. Be sure to check the box indicating that you would like to hear from us to receive updates and exclusive discounts on future editions of this product or related products.

To access this companion website, follow these steps:

Step 1. Go to www.pearsonitcertification.com/register and log in or create a new account.

Step 2. Enter the ISBN: **9780137866885**.

Step 3. Answer the challenge question as proof of purchase.

Step 4. Click the Access Bonus Content link in the Registered Products section of your account page to be taken to the page where your downloadable content is available.

Please note that many of the companion content files—especially image and video files—are very large.

If you are unable to locate the files for this title by following these steps, please visit www.pearsonITcertification.com/contact and select the Site Problems/Comments option. Our customer service representatives will assist you.

Pearson Test Prep Practice Test Software

As noted previously, this book comes complete with the Pearson Test Prep practice test software, including two full exams. These practice tests are available to you either online or as an offline Windows application. To access the practice exams that were developed with this book, please see the instructions below.

How to Access the Pearson Test Prep (PTP) App

You have two options for installing and using the Pearson Test Prep application: a web app and a desktop app. To use the Pearson Test Prep application, start by finding the registration code that comes with the book. You can find the code in these ways:

- You can get your access code by registering the print ISBN (9780137866885) on pearsonitcertification.com/register. Make sure to use the print book ISBN, regardless of whether you purchased an eBook or the print book. After you register the book, your access code will be populated on your account page

under the Registered Products tab. Instructions for how to redeem the code are available on the book's companion website by clicking the Access Bonus Content link.

- Premium Edition: If you purchase the Premium Edition eBook and Practice Test directly from the Pearson IT Certification website, the code will be populated on your account page after purchase. Just log in at pearsonitcertification.com, click Account to see details of your account, and click the digital purchases tab.

NOTE After you register your book, your code can always be found in your account under the Registered Products tab.

Once you have the access code, to find instructions about both the PTP web app and the desktop app, follow these steps:

Step 1. Open this book's companion website as shown earlier in this Introduction under the heading, "Companion Website."

Step 2. Click the **Practice Exams** button.

Step 3. Follow the instructions listed there for both installing the desktop app and using the web app.

Note that if you want to use the web app only at this point, just navigate to pearsontestprep.com, log in using the same credentials used to register your book or purchase the Premium Edition, and register this book's practice tests using the registration code you just found. The process should take only a couple of minutes.

Customizing Your Exams

When you are in the exam settings screen, you can choose to take exams in one of three modes:

- **Study Mode:** This mode allows you to fully customize your exams and review answers as you are taking the exam. This is typically the mode you use first to assess your knowledge and identify information gaps.
- **Practice Exam Mode:** This mode locks certain customization options in order to present a realistic exam experience. Use this mode when you are preparing to test your exam readiness.
- **Flash Card Mode:** This mode strips out the answers and presents you with only the question stem. This mode is great for late-stage preparation, when you really want to challenge yourself to provide answers without the benefit

of seeing multiple-choice options. This mode does not provide the detailed score reports that the other two modes provide, so it is not the best mode for helping you identify knowledge gaps.

In addition to these three modes, you can select the source of your questions. You can choose to take exams that cover all of the chapters, or you can narrow your selection to just a single chapter or the chapters that make up specific parts in the book. All chapters are selected by default. If you want to narrow your focus to individual chapters, simply deselect all the chapters then select only those on which you wish to focus in the Objectives area.

You can also select the exam banks on which to focus. Each exam bank comes complete with a full exam of questions that cover topics in every chapter. You can have the test engine serve up exams from all four banks or just from one individual bank by selecting the desired banks in the exam bank area.

There are several other customizations you can make to your exam from the exam settings screen, such as the time allowed to take the exam, the number of questions served up, whether to randomize questions and answers, whether to show the number of correct answers for multiple-answer questions, and whether to serve up only specific types of questions. You can also create custom test banks by selecting only questions that you have marked or questions on which you have added notes.

Updating Your Exams

If you are using the online version of the Pearson Test Prep software, you should always have access to the latest version of the software as well as the exam data. If you are using the Windows desktop version, every time you launch the software while connected to the Internet, it checks if there are any updates to your exam data and automatically downloads any changes that were made since the last time you used the software.

Sometimes, due to a number of factors, the exam data might not fully download when you activate your exam. If you find that figures or exhibits are missing, you might need to manually update your exams. To update a particular exam you have already activated and downloaded, simply select the Tools tab, and click the Update Products button. Again, this is only an issue with the desktop Windows application.

If you wish to check for updates to the Windows desktop version of the Pearson Test Prep exam engine software, simply select the Tools tab and click the Update Application button. Doing so enables you to ensure that you are running the latest version of the software engine.

Credits

Figure 4-1-Figure 4-3, Figure 4-5, Figure 5-5: Linus Torvalds

Figure 5-6, Figure 5-7: Wireshark Foundation

Figure 6-1, Figure 7-3, Figure 15-9, Figure 15-10, Figure 18-1-Figure 18-17, Figure 19-2-Figure 19-6, Figure 20-1, Figure 22-1: Canonical Ltd

Figure 6-2: Debian

Figure 6-3-Figure 6-10, Figure 7-1: SUSE

Figure 7-2: Microsoft

Figure 13-2: The GNOME Project

Figure 13-3-Figure 13-6: KDE

Figure 14-1-Figure 14-4: Red Hat, Inc

Figure 15-7-Figure 15-8: Atlassian

Figure 18-18: Gparted

Cover credit: Quardia/Shutterstock

Using Remote Connectivity for System Management

Used to be that you could just move your chair and work on a server system, way back in the day, but so very quickly that changed to having all the servers in the server room somewhere else, and not wanting to walk there, or even being in the location or even city or country as the servers!

Having grown up in the era of computing when using Telnet (an unsecure remote connectivity option that preceded SSH) and clear-text FTP was commonplace, I've witnessed the evolution of the world of remote networking from a much kinder and gentler place in which hacking was not very common to the current situation in which hacking is a persistent threat and secure terminal access to connect to remote systems safely and reliably to do work, mainly administering them, is an absolute requirement.

Today, you must have SSH installed and configured to be the most secure you can make it, as described in this chapter. You also need to know the importance of using passphrase authentication instead of password authentication to connect to a remote server or group of servers.

Another topic of great importance discussed in this chapter is the concept of privilege elevation or, as the Linux+ exam objectives state, "executing commands as another user." This requires you to have an understanding of several different tools, both for the exam and as a responsible systems administrator.

The elevation of privilege is even more appropriate in conjunction with SSH because typically you should never allow the root user to sign in over SSH. Gaining access to root-restricted resources means you need to elevate yourself to having root access or equivalent when you get *to* the other system.

"Do I Know This Already?" Quiz

The "Do I Know This Already?" quiz enables you to assess whether you should read this entire chapter or simply jump to the "Exam Preparation Tasks" section for review. If you are in doubt, read the entire chapter. Table 11-1 outlines the major headings in this chapter and the corresponding "Do I Know This Already?" quiz questions. You can find the answers in Appendix A, "Answers to the 'Do I Know This Already?' Quizzes and Review Questions."

Table 11-1 "Do I Know This Already?" Foundation Topics Section-to-Question Mapping

Foundation Topics Section	Questions Covered in This Section
SSH (Secure Shell)	1–3
Executing Commands as Another User	4–6

CAUTION The goal of self-assessment is to gauge your mastery of the topics in this chapter. If you do not know the answer to a question or are only partially sure of the answer, you should mark that question as wrong for purposes of the self-assessment. Giving yourself credit for an answer you correctly guess skews your self-assessment results and might provide you with a false sense of security.

1. You want to use a more secure tool than rpc to remotely copy data across the network. Which of the following tools would you use?
   a. ssh-add
   b. sftp
   c. ssh-agent
   d. scp

2. You want to disable Secure Shell logins for all users except the root user. Which of the following files would you create to make this happen?
   a. **/etc/nossh**
   b. **/etc/nologin**
   c. **/etc/disablessh**
   d. **/etc/sshrootonly**

3. The process of allowing remote-running GUI-based applications to display locally when connected to the remote system via SSH is called _____.
   a. Remote Display
   b. SSH GUI Mode
   c. X11 Forwarding
   d. Tunnel Mode

4. Which command allows you to execute commands as another user, but only if you know the other user's password?
   a. **runas**
   b. **pkexec**
   c. **sudo**
   d. **su**

5. Which option to the su command allows you to fully take on the user's account settings, including settings that are applied during the login process?
   a. **-a**
   b. **-u**
   c. **-l**
   d. **-r**

6. Which file is used to configure sudo access?
   a. **/etc/config/sudo.config**
   b. **/etc/default/sudoers**
   c. **/etc/sudo**
   d. **/etc/sudoers**

Foundation Topics

SSH (Secure Shell)

As mentioned at the beginning of the chapter, the Telnet protocol sends passwords and data in clear text and shouldn't be trusted for important sessions and tasks. The *Secure Shell (SSH)* suite includes a protocol, a daemon, and client utilities that make your host-to-host shell sessions much more secure—about as secure as being at the physical console.

One of the features that makes SSH desirable as a remote protocol is its end-to-end encryption, which encrypts not only the username and password but also all communications.

The SSH suite replaces **telnet**, as well as **rsh**, **rexec**, **rcp**, and other unsecure utilities. You can use SSH to connect for a shell session, or you can use the **scp** command to remotely transfer files through the secure pipe that SSH builds between the hosts.

SSH Components

Key Topic

SSH includes a number of programs and files:

- ***ssh*:** Used for remote shell sessions on another host, it replaces the **telnet**, **rsh**, and **rexec** commands.
- **scp:** Used for remote copying operations, it replaces the **rcp** command.
- **sshd:** The SSH daemon.
- **ssh-agent:** Runs as a wrapper to the user's session and provides authentication when requested.
- ***ssh-add*:** Loads the user's key(s) into the agent.

The SSH package configuration files are somewhat scattered. SSH daemon and global configuration files are kept in the **/etc/ssh** directory, and local or user-specific configuration files are kept in the **~/.ssh** directory for each user.

Key Topic

The global configuration files include

- ***/etc/ssh/sshd_config*:** This is the main configuration for the **sshd** daemon.
- **/etc/ssh/ssh_host_[dr]sa_key:** These files, the **ssh_host_dsa_key** file and the **ssh_host_rsa_key** file, are in the same directory and are the private parts of the host's key structure and should be protected from public view. The

permissions for these files are 600 or rw for the root user and no permissions for anyone else.

- **/etc/ssh/ssh_host_[dr]sa_key.pub:** These files, the **ssh_host_dsa_key.pub** file and the **ssh_host_rsa_key.pub** file, are in the same directory and are the public parts of the host's key structure. These must be world-readable and write-only by the root user or set to 644.
- **/etc/nologin:** This isn't a part of SSH. However, if it's present, no one can log in via SSH except the root user. Non-root users see the contents of the **/etc/nologin** file and then are denied access to the system.

A couple of special file pairs affect how SSH works, particularly the **/etc/ssh/ssh_known_hosts** and **~/.ssh/*known_hosts*** files. The global file (**/etc/ssh/ssh_known_hosts**) is used to check the public key of a host attempting to attach via SSH. The local file (**~/.ssh/known_hosts**) is the file from which the client gets the public key of the remote server. If a new connection is begun to a previously unknown host, the user sees a message saying that the host is unknown and asking whether the user wants to store the host's key in his known hosts file. If the user answers in the affirmative, the host's public key is added to the **~/.ssh/known_hosts** file.

The **/etc/ssh/ssh_known_hosts** file should be world-readable and root-writable. The **~/.ssh/known_hosts** file must be owned by and writable for the user.

Key Topic

A file of interest, the **~/.ssh/authorized_keys** file, affects only a particular user's environment. This file is used to store the public keys that can be used for logging in as this user. These keys are matched with the keys presented by an **ssh** or **scp** client upon login request.

The SSH client utilities are versatile, with a number of options available to customize the experience. You just need to know the basics for the Linux+ exam, but this section includes a few fun options.

The SSH client command is used to replace the RSH and Telnet programs specifically. Its syntax is as follows:

Key Topic

```
# ssh -l username remotehost
```

If you don't specify a username with the **-l** option, the **ssh** command assumes that you want to use the name of the account with which you are locally logged in. For example, if you are logged in as the user ross and you execute the **ssh** command without the **-l** option, the command attempts to log you in as ross on the remote system.

For example, I could attach to the host mp3server as the user snuffy with this command:

```
# ssh -l snuffy mp3server
```

If I have not connected to this server before, I get a message similar to what's shown here:

```
The authenticity of host 'mp3server (192.168.33.44)' can't be
established.
RSA key fingerprint is 73:4f:fa:b0:42:a4:3a:a8:64:2c:ad:26:1
d:b1: 21:e0.
Are you sure you want to continue connecting (yes/no)?
```

If I answer **yes**, the host's public key is added to my **~/.ssh/known_hosts** file and looks something like this:

```
192.168.3.44 ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAIEA1gFIB9VQpFKWAZUzNM+ac/U81Tk9R8OCFfUkegVJXw
j6nqCISPyV2iJwaukcVVaVAQ+JR3EhvOvh4PhoSg4yzBSUkJ8aUBYoRSGj7PCD+vyWyi19
22HGxWbWooMBAO/Was8I7N0zQ6jxDO9qNOHcrIFeU7qbOCrKjQDM08HQjk0=
```

Rather than work with RCP or FTP for file transfer work, I tend to use SCP. The **scp** command uses the SSH protocol and encrypts the files sent from one host to another host. For example, if I wanted to transfer **file1** from my root user's home directory on my machine to the same location on a host named remotehost, I could use the following command:

```
# scp /root/file1 root@remotehost:/root/file1
```

The system would prompt me with the RSA key question (as shown in the previous **ssh** example) if I have not connected to this system previously from this account. I would be prompted for the password, and then the system would transfer the files. The output from a file transfer looks like this:

```
# root@192.168.1.73's password:
mypubkey.txt 100% |**********************| 1379 00:00
```

You can copy files from your host to another host, as shown previously, or copy files from a remote host to your system by reversing the source and target specifications.

You can even copy files from one remote system to another remote system. For example, the following command recursively copies the **/data** directory and all its contents from the remote1 host to the remote2 host after prompting you for the password for both hosts:

```
# scp -r root@remote1:/data root@remote2:/data
```

Another use of the SSH protocol is to log in to a host and use SSH to forward the output from an X client back to your display. This feature, which can be specifically invoked with the **-x** option, is referred to as an *X11 tunnel*.

SSH allows for skipping the password prompt when signing on between computers, which can be convenient if you use the **ssh** or **scp** command frequently and don't mind the possibility that someone could sit down at your accidentally unlocked station and have her way with your network!

NOTE There has been a lot of talk about why it's important to delete **.rhosts** files from user directories. Basically, if you have a user who has a hostname in her **.rhosts** file and that host also has the user's hostname in its **/etc/hosts_equiv** file, that user can log in without a password by using the **rlogin** command. This would be a security risk, so my advice is to delete these files with the following command:


```
# find /home -iname .rhosts -exec rm -f {} \;
```

This deletes all **.rhosts** files it finds in users' home directories and does not prompt you for each deletion.

NOTE The system-wide configuration for the SSH client is kept in the **/etc/ssh/*ssh_config*** file, while each user's individual configuration file for the SSH client is kept in the user's **~/.ssh/config** file.

The following example shows the steps required to enable SSH use without a password. In this example I have two machines, fattyre and murphy, both of which are Linux workstations with the necessary SSH software loaded, as per the defaults. This demonstration assumes that fattyre and murphy are both in each other's **/etc/hosts** files or resolvable via DNS.

Here's how you can enable SSH use without passwords:

Step 1. Log in to fattyre as the root user.

Step 2. For this example, create a new user named user1:

```
useradd -m user1
```

Step 3. Set user1's password with the **passwd** command to whatever you want:

```
passwd user1
```

Step 4. Switch to the user1 user account:

```
su - user1
```

Step 5. Create and set the permissions for the **.ssh** directory:

```
mkdir .ssh ; chmod 700 .ssh
```

Step 6. Generate an RSA key by using the ***ssh-keygen*** command:

```
ssh-keygen -b 1024 -t rsa
```

Step 7. When prompted for the location for the file, press **Enter** to accept the default.

Step 8. When prompted for a passphrase, enter

```
seatec astronomy
```

Step 9. Reenter the passphrase when prompted.

Step 10. Change to the **.ssh** directory and set the permissions on the **id_rsa.pub** file:

```
cd .ssh ; chmod 644 id_rsa.pub
```

Step 11. Copy the **id_rsa.pub** file to a new file called **authorized_keys**:

```
cp id_rsa.pub authorized_keys
```

NOTE The next steps take place on the host murphy.

Step 12. From the host murphy, ensure that you can contact the host fattyre with a ping:

```
ping fattyre
```

Step 13. Sign on to the host murphy as the root user.

Step 14. Add a user named user2:

```
useradd -m user2
```

Step 15. Set the password for user2:

```
passwd user2
```

Step 16. Enter the password twice to confirm it.

Step 17. Switch to the user2 account:

```
su - user2
```

Step 18. Make a directory and set its permissions with the following command:

```
mkdir .ssh ; chmod 700 .ssh
```

NOTE The next steps take place on the host fattyre.

Step 19. From the host **fattyre**, connect to the murphy host as user2:

```
ssh -l user2 murphy
```

Step 20. When prompted about the RSA key, answer **yes** and then enter user2's password.

Step 21. While logged in as user2 on the host murphy via SSH, copy the **authorized_keys** file from the fattyre host with the following command:

```
scp user1@fattyre:~/.ssh/authorized_keys ~/.ssh
```

The output of the **scp** program should look similar to this:

```
authorized_keys 100% |**********************| 236 00:00
```

Step 22. Exit user2 on the host murphy and return to being user1 on fattyre.

Step 23. On fattyre as user1, invoke the **ssh-agent** as a wrapper to your shell:

```
ssh-agent $SHELL
```

Step 24. Add your key to the agent:

```
ssh-add
```

Step 25. When prompted for the passphrase, enter the following:

```
no more tears
```

You then see output similar to this:

```
Identity added: /home/ssha/.ssh/id_rsa (/home/ssha/.ssh/
id_rsa)
```

Step 26. Try to log in as user2 on murphy and watch what happens:

```
ssh -l user Murphy
```

You shouldn't see any password prompt; you should see only the confirmation of where you last logged in from:

```
Last login: Wed May 26 13:46:55 from fattyre
```

Step 27. If you do see a prompt for the passphrase, enter **no more tears** as you did before.

This is all it takes to get two accounts and machines set up to use SSH utilities without having to enter anything but the **ssh-agent** command along with the passphrase. Remember that **ssh-agent** resides in memory and wraps a security blanket around your shell session, answering any SSH-related security requests for you. The **ssh-add** utility is for adding key information into the agent and doesn't have to be run again as long as your key information remains the same.

NOTE The ***ssh-copy-id*** command is also a possible choice for sending a user's authorized key to another server. For example, to have the user zakkw's authorized key exist on the Eternal server, you could use the command

```
# ssh-copy -id -i ~/zakkw/.ssh/keyfile zakkw@eternal
```

This will copy the user zakkw's file to the remote server and install it into the **authorized_keys** file, prompting for a password to authenticate the process.

Ideally, this would be performed by the root user for both systems and assumes that you are on a remote system from the Eternal server.

Tunneling

Key Topic

One of the greatest features of SSH is that it can *tunnel*—provide a conduit from inside one network, and even behind a firewall, through to another network. In many cases, using ***tunneling*** can enable you to do things that either your network administrator doesn't want you to do or you need to do because of an overly restrictive policy, such as at a coffee shop or Internet cafe.

Let's talk about some of the various scenarios where tunneling can come in handy.

X11 Forwarding

X is complex and hard to set up sometimes, and it might seem that tunneling X from another machine to show on yours would be hard too, but ***X11 forwarding*** is fairly straightforward due to the magical properties of ssh tunneling.

Let's say you have a Linux system named cygnusx1 on which you want to run an application in the GUI environment, but you want that application that runs on the remote host to display on your local system.

Here's a possible set of steps you might take:

Step 1. On a Mac, download and install XQuartz (https:/www.xquartz.org).

Step 2. Run the command **ssh -X ursulak@cygnusx1**.

Step 3. After a shell opens in the terminal on the remote host (cygnusx1), run the app.

Step 4. In a second or two, the remote application, running as a process on the remote host, will display on your system as if it were being run locally.

This is just an example of how forwarding X11 applications from the host they are running on to your local system would work. All sorts of things could go wrong, but

that topic is beyond the scope of the Linux+ exam. The main thing is to understand the concept of X11 forwarding, which we have more than adequately covered.

Port Forwarding

Port forwarding is typically used in scenarios in which there is a need to get around some overly strict or controlling network or firewall. Keep in mind, though, that often there are *very* good reasons for those restrictions and rules being in place, so be responsible and don't willingly cause issues using port forwarding.

Using SSH to forward ports takes several paths, but the main concept is the same: you are using the ssh client on one system to tunnel out to the ssh server on another system and cause services that are represented by a port on the latter system to be mapped, or to appear to be connected, to the other system.

In general, port forwarding occurs in three main ways:

- **Local port forwarding:** This enables you to cause a remote port to be mapped to, and to appear as if it were on, your local system. Kind of like mounting an NFS share locally, mapping a port from a remote system to yours locally effectively makes your local system appear as if it is providing that service.
- **Remote port forwarding:** Flip the scenario around and allow your local system resources to be used by a remote machine. For example, I might map a remote system's port 8080 to my local 5500 port, and anyone connecting to that remote server on the 8080 port will get transported to my local port and service.
- **Dynamic port forwarding:** The term ***dynamic port forwarding***, also known as dynamic SOCKS proxying, is a method used to securely tunnel network traffic through a remote server or proxy. Sometimes you don't want to explicitly assign ports and just want the SOCKS proxy on your system to use dynamically assigned local ports and handle all the details. Think of a situation where you need to access ports and protocols that are not allowed through a convention center's network setup. You can use what is effectively a VPN/proxy to drill through the local restrictive network stack and connect to and communicate freely with your desired target hosts, services, and ports.

NOTE The beauty of using SSH tunneling for these purposes is that you don't have to worry that by doing so you are exposing the local network or system unnecessarily; you're using the very secure SSH protocols and stack to do all of this!

Executing Commands as Another User

There are times when you need to execute a command as a different user account. For example, if you log in to the system as a non-root user, but need to execute a command with root privileges.

This section describes methods of running commands as different user accounts, including the **sudo** command, the **su** command and the **pkexec** command.

The sudo Command

The problem with the ***su*** command is that to provide a user with elevated privileges, you need to provide the user with the root password, which would give that user full administrative access to the system.

Often you want to allow a regular user to execute some commands, but not all commands, as the root user. For example, if a network error occurs on a user's workstation, you might want that user to be allowed to restart the networking service. On some systems, this can be accomplished by executing the following command:

```
# /etc/rc.d/init.d/network restart
```

To execute this command successfully, the user needs to have root privileges. This is where you either give the user the root password (which is not recommended) or you give limited root access the correct and reasonable way, via the ***sudo*** command and its partner tools.

Instead of providing the user with the root password, you can set up the **sudo** command to allow the user to run just the necessary command. To do this, you need to log in as the root user and then execute the ***visudo*** command.

```
# visudo
```

This command allows you to edit the ***/etc/sudoers*** file, the file that allows you to provide root access for specific commands to specific users. The **visudo** command automatically assumes that you want to use the **vi** editor to edit this file. To use a different editor, such as the **nano** editor, execute a command like the following:

```
# export EDITOR=nano
```

Key Topic

NOTE Why use the **visudo** command instead of editing the **/etc/sudoers** file directly? The **visudo** command performs some error checking when you exit the editor to make sure you didn't make formatting mistakes.

The sudoedit Command

One of the conundrums of granting a user access to edit a configuration file is that if you are using **vi/vim**, you are essentially giving the user the ability to run *any* command as root.

To prevent a user from gaining shell access with a simple set of keystrokes from **vi/vim** while running it as root, there exists the ***sudoedit*** command, which is really just a symbolic link to a function contained in the **sudo** command.

Key Topic

The **sudoedit** command lets you allow a user to use any editor, not just **vi/vim**. It also enables the user to edit the using sudo access, rather than having to log in to the root user account.

When a user edits a file by using the **sudoedit** functionality, a temporary copy of the file(s) is made, and it is owned by the user in question. Since the user is now the owner of the temporary file(s), he can successfully edit the file(s) without having root access. Upon saving the file(s), the temporary copy owned by the user is copied back to the original file location, and the original ownership is restored; the now unnecessary temporary copy is discarded.

To configure **sudoedit**, add the following line to the **/etc/sudoers** file:

```
%newsudo ALL = sudoedit /some/path/to/a/file
```

Configure the newsudo group in **/etc/sudoers** to have the users you want to use **sudoedit**, and then all they need to do is run the command:

```
sudoedit /path/to/that/file
```

The **/etc/sudoers** file has many options. For the Linux+ certification exam, you just need to know how to provide a user with the ability to execute commands as the root user. For example, if you want a user account with the name ross to be able to run all commands as the root user, add the following line:

```
ross ALL=(ALL) ALL
```

To limit a user to a specific command, such as the **/etc/rc.d/init.d/network** command, add the following line:

```
ross ALL=(ALL) /etc/rc.d/init.d/network
```

For a user to execute a command as root, she needs to use the **sudo** command. The syntax is as follows:

```
# sudo /etc/rc.d/init.d/network restart
```

The user is then prompted for her own password (not the root password). If the correct password is given and the access is permitted based on an entry in the **/etc/sudoers** file, the command is executed as the root user. If the user attempts to execute a command that she is not authorized to execute, an error message appears on the screen, and the attempt is logged.

User Privilege Escalation

Users on a Linux system come in the following types, and it is important to know all three types, which type you are logged in, and how to escalate or deescalate your privileges at will by switching from one type to another:

- **Root:** This is the root user, who is the super user and the most privileged user on the system. There should be only one of them, characterized by the name root and the UID (user ID) 0.
- **Standard:** Otherwise known as "regular" or "normal" users, these are the rank-and-file users on the system; they have no special privileges and typically have UIDs that range from 1000 and higher.
- **Service:** These are the accounts that have to exist to ensure that every service or daemon on the system is running as a user, since every process must have a user attached. These accounts are never signed into; they exist in the **/etc/passwd** file and may even have **/bin/nologin** as the specified shell.

The best security practice is to avoid logging in as the root user unless you need to perform specific administration commands. In most cases, you should not log in as the root user directly but rather should gain root access by executing either the **su** command or the **sudo** command.

NOTE The wheel group is an odd thing on the Linux system these days. Traditionally used on Unix systems to allow users to gain root access, the wheel group is often now tied directly to having **sudo** access.

If the wheel group is configured to have privileged access via **sudo** and the **/etc/sudoers** file, then adding a user to the wheel group gives the user those privileges. For example, in our openSUSE system, the wheel group is set up to be able to allow members of that group to execute any command, just as the root would be able to:

```
%wheel  ALL=(ALL) ALL
```

This entry is normally commented out, but it would be very easy to remove the single **#** comment in front of it in the default file to enable the wheel group (and its members) to have full administrative access to the system.

The su Command

To gain access to another user account with the **su** command, use the following syntax:

```
su account_name
```

Key Topic

For example, to switch to the root account, execute the following command:

```
# su root
```

This provides you with a non-login shell for the root user. Typically you want a login shell because it provides you with the full user environment (environment variables, shell customizations, and so on). This can be accomplished by using the **-l** option or just a **-** option:

```
# su - root
# su -l root
```

To gain access to a regular user account, you must provide the account name. However, if you don't provide an account name, the **su** command assumes that you want to switch to the root account. As a result, the following commands are all the same:

- **su - root**
- **su -l root**
- **su -**
- **su -l**

When switching from the root account to a regular user account, no password is required. This means the root user can switch to a regular user account to test that account's features (or troubleshoot problems for the user) without having to change that user's password.

To switch from a regular user account to any other account, you must know the password of the account you are switching to.

NOTE Some distributions' versions of the **su** command allow for the use of X and remote X; simply use the **sux** command instead of the **su** command. This is most notably present on the openSUSE and SUSE Linux Enterprise distributions.

PolicyKit

PolicyKit, also known as polkit, is a system service in Linux that provides a framework for controlling system-wide privileges and permissions.

PolicyKit exists to provide application-level definition and handling of unprivileged access to privileged processes. For example, you might use PolicyKit to provide a user the ability (and the rights) to perform a task by executing a command with elevated privileges. If you think that sounds like the **sudo** command, it's understandable, because they both have fairly similar goals.

One difference is that PolicyKit is a little easier to use, and certainly less tedious, because you don't have to preface almost everything you do with the **sudo** command.

NOTE The name PolicyKit is used in this book to match the Linux+ exam objectives, but the current package has been renamed Polkit. One of the main positives of PolicyKit is that it's a central place for defining and accessing policies that allow unprivileged users to perform what would normally be privileged actions.

The PolicyKit local configuration is kept in **/etc/polkit-1/localauthority** and uses the common method of include files that contain PolicyKit configuration and end either in **.conf** or, for the more specialized files, **.pkla**.

The following are examples of the types of actions PolicyKit can be configured for:

- Allow the user to configure wireless connections
- Make it possible to mount and unmount USB and other detached media devices
- Let the user manage shutdown, reboot, and hibernate events
- Make devices accessible that are traditionally difficult to access, such as system audio

The pkexec Command

With the previous discussion of the PolicyKit package, **pkexec** makes a lot more sense, as it's the most common way to utilize the PolicyKit rules.

The ***pkexec*** command, when used to run another command, will execute that command as the targeted user. The user can be specified, but if it is not, **pkexec** attempts to execute the target command as the root user.

For example, to execute the **lemmy.sh** script as the root user, you would type

```
# pkexec lemmy.sh
```

Since a user is not specified, the default for **pkexec** is to attempt to run the subsequent command, script, or executable as the root user.

Summary

This chapter focused on how to remotely and securely connect with systems for the purposes of administering them, using the SSH suite of technologies and the various **ssh**-prefaced commands you learned about in this chapter.

You also learned about the methods for privilege elevation, or running commands or acting as another user, such as **su**, **sudo**, and **pkexec**.

Exam Preparation Tasks

As mentioned in the section "Goals and Methods" in the Introduction, you have a couple of choices for exam preparation: the exercises here, Chapter 23, "Final Preparation," and the exam simulation questions in the Pearson Test Prep Software Online.

Review All Key Topics

Review the most important topics in this chapter, noted with the Key Topic icon in the left margin of the page. Table 11-2 lists these key topics and the page number on which each is found.

Key Topic

Table 11-2 Key Topics for Chapter 11

Key Topic Element	Description	Page Number
List	Programs and files that SSH includes	408
List	SSH global configuration files	408
Paragraph	Description of the **~/.ssh/authorized_keys** file	409
Paragraph	Example syntax for connecting to a remote system via the **ssh** command	409
Note	Example of deleting all the **.rhosts** files on a given system	411
Step list	Enabling SSH use without passwords	411
Section	Tunneling	414
Note	Reason to use the **visudo** command instead of editing **/etc/sudoers** directly	416
Paragraph	Using the **sudoedit** command to allow a user to use any editor	417
Paragraph	Switching to the root account using the **su** command	419
Paragraph	Example of using the **pkexec** command to run a script as an different user	420

Define Key Terms

Define the following key terms from this chapter and check your answers in the glossary:

ssh, **ssh-add**, **/etc/ssh/sshd_config**, **known_hosts**, **ssh_config**, **ssh-keygen**, **ssh-copy-id**, tunneling, X11 forwarding, port forwarding, dynamic forwarding, privilege escalation, **su**, **sudo**, **visudo**, **/etc/sudoers**, **sudoedit**, PolicyKit, **pkexec**

Review Questions

The answers to these review questions are in Appendix A.

1. After configuring the PolicyKit rules for your system, what command would you use to use those rules when executing a target command that your current user doesn't have rights to execute alone?
   a. **sudo**
   b. **pkexec**
   c. **suexec**
   d. **execit**

2. When configuring the **sudo** command, what is the full path and filename of its primary configuration file?

   _____

3. You are able to access a remote system using just a passphrase for authentication. What must you have copied from your system to the remote system in order for this to happen?
   a. Your personal public key
   b. The system's public key
   c. The wheel group's public key
   d. The remote user's private key

4. When configuring your system to allow or deny certain groups or users to sign in via SSH, what is the full path and filename of the configuration file where these settings are kept?

   _____

5. If you invoke the **ssh** command with the **-X** option, what are you likely to be doing after you sign on to the remote system?
   a. Just standard commands
   b. Running xeyes locally and displaying remotely
   c. Running X11 on the remote system and displaying locally
   d. Running remote X apps that display locally

6. Which of the following commands is specifically designed to make it more secure to edit files when using **sudo** to elevate your privileges?
   a. **sudovim**
   b. **visudo**
   c. **sudoedit**
   d. **nanobot**

This page intentionally left blank

Index

Symbols

Numbers

A

B

C

D

E

F

G

H

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z