Module 1

Coursera : Tools of the trade : Linux & SQL

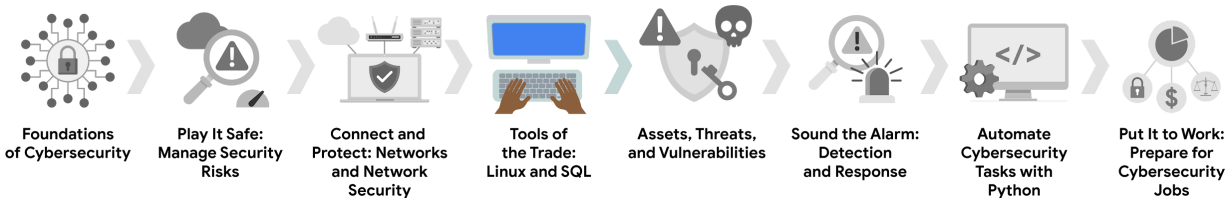
Course 4 overview

Hello, and welcome to Tools of the Trade: Linux and SQL, the fourth course in the Google Cybersecurity Certificate. You're on an exciting journey!

By the end of this course, you will develop a greater understanding of the basics of computing that will support your work as a security analyst. You will learn foundational concepts related to understanding operating systems, communicating with the Linux operating system through commands, and querying databases with Structured Query Language (SQL). These are key concepts in the cybersecurity field and understanding them will help you keep organizations secure.

Certificate program progress

The Google Cybersecurity Certificate program has eight courses. Tools of the Trade: Linux and SQL is the fourth course.



1. Foundations of Cybersecurity — Explore the cybersecurity profession, including significant events that led to the development of the cybersecurity field and its continued importance to organizational operations. Learn about entry-level cybersecurity roles and responsibilities.
2. Play It Safe: Manage Security Risks — Identify how cybersecurity professionals use frameworks and controls to protect business operations, and explore common cybersecurity tools.
3. Connect and Protect: Networks and Network Security — Gain an understanding of network-level vulnerabilities and how to secure networks.
4. Tools of the Trade: Linux and SQL — *(current course)* Explore foundational computing skills, including communicating with the Linux operating system through the command line and querying databases with SQL.
5. Assets, Threats, and Vulnerabilities — Learn about the importance of security controls and developing a threat actor mindset to protect and defend an organization's assets from various threats, risks, and vulnerabilities.
6. Sound the Alarm: Detection and Response — Understand the incident response lifecycle and practice using tools to detect and respond to cybersecurity incidents.
7. Automate Cybersecurity Tasks with Python — Explore the Python programming language and write code to automate cybersecurity tasks.

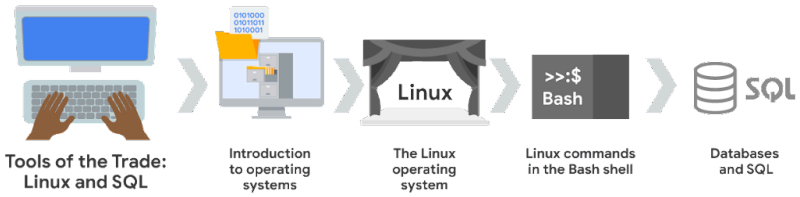
8. Put It to Work: Prepare for Cybersecurity Jobs — Learn about incident classification, escalation, and ways to communicate with stakeholders. This course closes out the program with tips on how to engage with the cybersecurity community and an introduction to AI in cybersecurity.
9. Accelerate Your Job Search with AI — Gain practical job search strategies and learn how to leverage AI tools (like Gemini and NotebookLM) to uncover your most valuable skills, create a job search plan, manage your applications, and practice for interviews as you navigate your path to your next role.

Course 4 content

Each course of this certificate program is broken into modules. You can complete courses at your own pace, but the module breakdowns are designed to help you finish the entire Google Cybersecurity Certificate in about six months.

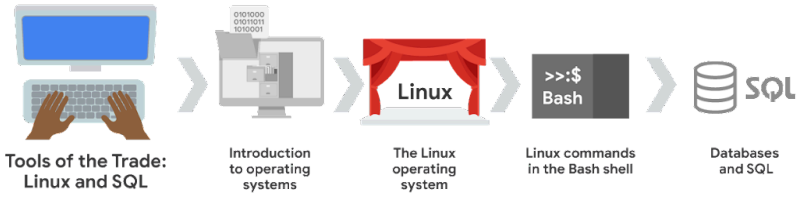
What's to come? Here's a quick overview of the skills you'll learn in each module of this course.

Module 1: Introduction to operating systems



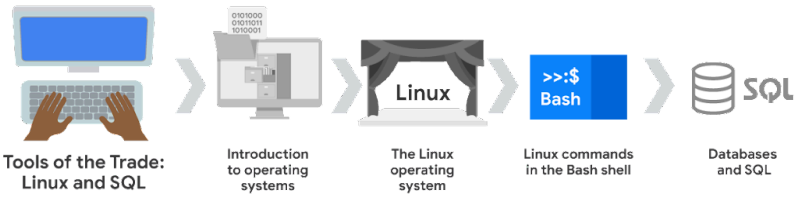
You will learn about the relationship between operating systems, hardware, and software, and become familiar with the primary functions of an operating system. You'll recognize common operating systems in use today and understand how the graphical user interface (GUI) and command-line interface (CLI) both allow users to interact with the operating system.

Module 2: The Linux operating system



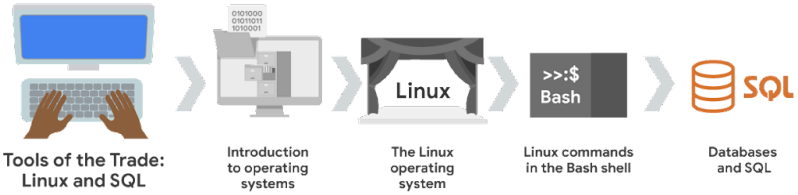
You will be introduced to the Linux operating system and learn how it is commonly used in cybersecurity. You'll also learn about Linux architecture and common Linux distributions. In addition, you'll be introduced to the Linux shell and learn how it allows you to communicate with the operating system.

Module 3: Linux commands in the Bash shell



You will be introduced to Linux commands as entered through the Bash shell. You'll use the Bash shell to navigate and manage the file system and to authorize and authenticate users. You'll also learn where to go for help when working with new Linux commands.

Module 4: Databases and SQL



You will practice using SQL to communicate with databases. You'll learn how to query a database and filter the results. You'll also learn how SQL can join multiple tables together in a query.

What to expect

Each course offers many types of learning opportunities:

- Videos led by Google instructors teach new concepts, introduce the use of relevant tools, offer career support, and provide inspirational personal stories.
- Readings build on the topics discussed in the videos, introduce related concepts, share useful resources, and describe case studies.
- Discussion prompts explore course topics for better understanding and allow you to chat and exchange ideas with other learners in the discussion forums.
- Self-review activities and labs give you hands-on practice in applying the skills you are learning and allow you to assess your own work by comparing it to a completed example.
- Interactive plug-ins encourage you to practice specific tasks and help you integrate knowledge you have gained in the course.
- In-video quizzes help you check your comprehension as you progress through each video.
- Practice quizzes allow you to check your understanding of key concepts and provide valuable feedback.
- Graded quizzes demonstrate your understanding of the main concepts of a course. You must score 80% or higher on each graded quiz to obtain a certificate, and you can take a graded quiz multiple times to achieve a passing score.

Tips for success

- It is strongly recommended that you go through the items in each lesson in the order they appear because new information and concepts build on previous knowledge.
- Participate in all learning opportunities to gain as much knowledge and experience as possible.
- If something is confusing, don't hesitate to replay a video, review a reading, or repeat a self-review activity.
- Use the additional resources that are referenced in this course. They are designed to support your learning. You can find all of these resources in the Resources tab.
- When you encounter useful links in this course, bookmark them so you can refer to the information later for study or review.
- Understand and follow the Coursera Code of Conduct to ensure that the learning community remains a welcoming, friendly, and supportive place for all members.

Compare operating systems

You previously explored why operating systems are an important part of how a computer works. In this reading, you'll compare some popular operating systems used today. You'll also focus on the risks of using legacy operating systems.

Common operating systems

The following operating systems are useful to know in the security industry: Windows, macOS®, Linux, ChromeOS, Android, and iOS.

Windows and macOS

Windows and macOS are both common operating systems. The Windows operating system was introduced in 1985, and macOS was introduced in 1984. Both operating systems are used in personal and enterprise computers.

Windows is a closed-source operating system, which means the source code is not shared freely with the public. macOS is partially open source. It has some open-source components, such as macOS's kernel. macOS also has some closed-source components.

Linux

The first version of Linux was released in 1991, and other major releases followed in the early 1990s. Linux is a completely open-source operating system, which means that anyone can access Linux and its source code. The open-source nature of Linux allows developers in the Linux community to collaborate.

Linux is particularly important to the security industry. There are some distributions that are specifically designed for security. Later in this course, you'll learn about Linux and its importance to the security industry.

ChromeOS

ChromeOS launched in 2011. It's partially open source and is derived from Chromium OS, which is completely open source. ChromeOS is frequently used in the education field.

Android and iOS

Android and iOS are both mobile operating systems. Unlike the other operating systems mentioned, mobile operating systems are typically used in mobile devices, such as phones, tablets, and watches. Android was introduced for public use in 2008, and iOS was introduced in 2007. Android is open source, and iOS is partially open source.

Operating systems and vulnerabilities

Security issues are inevitable with all operating systems. An important part of protecting an operating system is keeping the system and all of its components up to date.

Legacy operating systems

A legacy operating system is an operating system that is outdated but still being used. Some organizations continue to use legacy operating systems because software they rely on is not compatible with newer operating systems. This can be more common in industries that use a lot of equipment that requires embedded software—software that's placed inside components of the equipment.

Legacy operating systems can be vulnerable to security issues because they're no longer supported or updated. This means that legacy operating systems might be vulnerable to new threats.

Other vulnerabilities

Even when operating systems are kept up to date, they can still become vulnerable to attack. Below are several resources that include information on operating systems and their vulnerabilities.

- Microsoft Security Response Center (MSRC): A list of known vulnerabilities affecting Microsoft products and services
- Apple Security Updates: A list of security updates and information for Apple® operating systems, including macOS and iOS, and other products
- Common Vulnerabilities and Exposures (CVE) Report for Ubuntu: A list of known vulnerabilities affecting Ubuntu, which is a specific distribution of Linux
- Google Cloud Security Bulletin: A list of known vulnerabilities affecting Google Cloud products and services

Keeping an operating system up to date is one key way to help the system stay secure. Because it can be difficult to keep all systems updated at all times, it's important for security analysts to be knowledgeable about legacy operating systems and the risks they can create.

Key takeaways

Windows, macOS, Linux, ChromeOS, Android, and iOS are all commonly used operating systems. Security analysts should be aware of vulnerabilities that affect operating systems. It's especially important for security analysts to be familiar with legacy operating systems, which are systems that are outdated but still being used.

Requests to the operating system

Operating systems are a critical component of a computer. They make connections between applications and hardware to allow users to perform tasks. In this reading, you'll explore this complex process further and consider it using a new analogy and a new example.

Booting the computer

When you boot, or turn on, your computer, either a BIOS or UEFI microchip is activated. The Basic Input/Output System (BIOS) is a microchip that contains loading instructions for the computer and is prevalent in older systems. The Unified Extensible Firmware Interface (UEFI) is a microchip that contains loading instructions for the computer and replaces BIOS on more modern systems.

The BIOS and UEFI chips both perform the same function for booting the computer. BIOS was the standard chip until 2007, when UEFI chips increased in use. Now, most new computers include a UEFI chip. UEFI provides enhanced security features.

The BIOS or UEFI microchips contain a variety of loading instructions for the computer to follow. For example, one of the loading instructions is to verify the health of the computer's hardware.

The last instruction from the BIOS or UEFI activates the bootloader. The bootloader is a software program that boots the operating system. Once the operating system has finished booting, your computer is ready for use.

Completing a task

As previously discussed, operating systems help us use computers more efficiently. Once a computer has gone through the booting process, completing a task on a computer is a four-part process.

User

The first part of the process is the user. The user initiates the process by having something they want to accomplish on the computer. Right now, you're a user! You've initiated the process of accessing this reading.

Application

The application is the software program that users interact with to complete a task. For example, if you want to calculate something, you would use the calculator application. If you want to write a report, you would use a word processing application. This is the second part of the process.

Operating system

The operating system receives the user's request from the application. It's the operating system's job to interpret the request and direct its flow. In order to complete the task, the operating system sends it on to applicable components of the hardware.

Hardware

The hardware is where all the processing is done to complete the tasks initiated by the user. For example, when a user wants to calculate a number, the CPU figures out the answer. As another example, when a user wants to save a file, another component of the hardware, the hard drive, handles this task.

After the work is done by the hardware, it sends the output back through the operating system to the application so that it can display the results to the user.

The OS at work behind the scenes

Consider once again how a computer is similar to a car. There are processes that someone won't directly observe when operating a car, but they do feel it move forward when they press the gas pedal. It's the same with a computer. Important work happens inside a computer that you don't experience directly. This work involves the operating system.

You can explore this through another analogy. The process of using an operating system is also similar to ordering at a restaurant. At a restaurant you place an order and get your food, but you don't see what's happening in the kitchen when the cooks prepare the food.

Ordering food is similar to using an application on a computer. When you order your food, you make a specific request like “a small soup, very hot.” When you use an application, you also make specific requests like “print three double-sided copies of this document.”

You can compare the food you receive to what happens when the hardware sends output. You receive the food that you ordered. You receive the document that you wanted to print.

Finally, the kitchen is like the OS. You don’t know what happens in the kitchen, but it’s critical in interpreting the request and ensuring you receive what you ordered. Similarly, though the work of the OS is not directly transparent to you, it’s critical in completing your tasks.

An example: Downloading a file from an internet browser

Previously, you explored how operating systems, applications, and hardware work together by examining a task involving a calculation. You can expand this understanding by exploring how the OS completes another task, downloading a file from an internet browser:

- First, the user decides they want to download a file that they found online, so they click on a download button near the file in the internet browser application.
- Then, the internet browser communicates this action to the OS.

- The OS sends the request to download the file to the appropriate hardware for processing.
- The hardware begins downloading the file, and the OS sends this information to the internet browser application. The internet browser then informs the user when the file has been downloaded.

Key takeaways

Although it operates in the background, the operating system is an essential part of the process of using a computer. The operating system connects applications and hardware to allow users to complete a task.

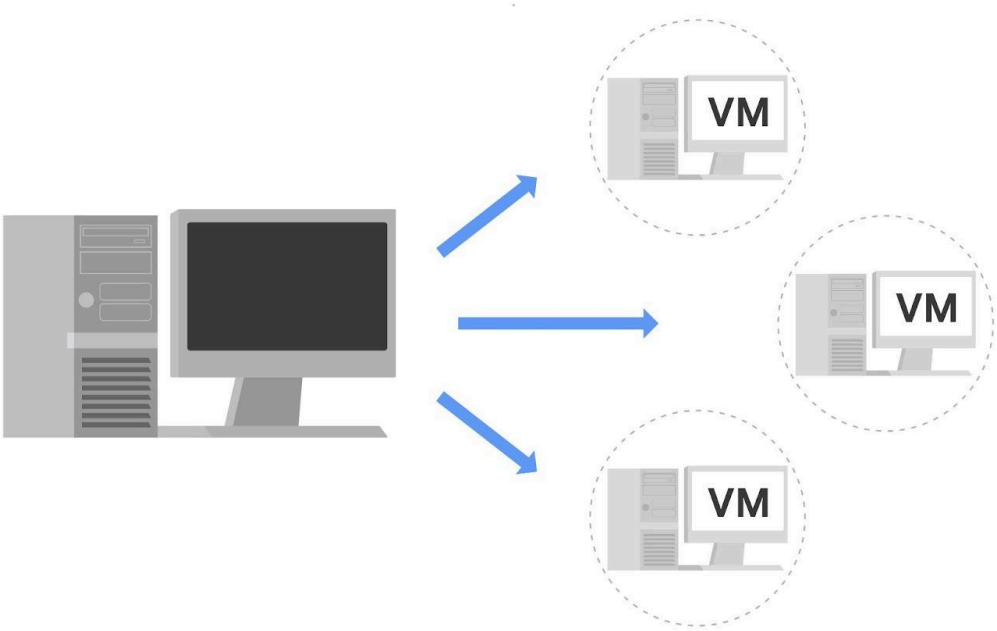
Virtualization technology

You've explored a lot about operating systems. One more aspect to consider is that operating systems can run on virtual machines. In this reading, you'll learn about virtual machines and the general concept of virtualization. You'll explore how virtual machines work and the benefits of using them.

What is a virtual machine?

A virtual machine (VM) is a virtual version of a physical computer. Virtual machines are one example of virtualization. Virtualization is the process of using software to create virtual representations of various physical machines. The term "virtual" refers to machines that don't exist physically, but operate like they do because their software

simulates physical hardware. Virtual systems don’t use dedicated physical hardware. Instead, they use software-defined versions of the physical hardware. This means that a single virtual machine has a virtual CPU, virtual storage, and other virtual hardware. Virtual systems are just code.



You can run multiple virtual machines using the physical hardware of a single computer. This involves dividing the resources of the host computer to be shared across all physical and virtual components. For example, Random Access Memory (RAM) is a hardware component used for short-term memory. If a computer has 16GB of RAM, it can host three virtual machines so that the physical computer and virtual machines each have 4GB of RAM. Also, each of these virtual machines would have their own operating system and function similarly to a typical computer.

Benefits of virtual machines

Security professionals commonly use virtualization and virtual machines. Virtualization can increase security for many tasks and can also increase efficiency.

Security

One benefit is that virtualization can provide an isolated environment, or a sandbox, on the physical host machine. When a computer has multiple virtual machines, these virtual machines are “guests” of the computer. Specifically, they are isolated from the host computer and other guest virtual machines. This provides a layer of security, because virtual machines can be kept separate from the other systems. For example, if an individual virtual machine becomes infected with malware, it can be dealt with more securely because it’s isolated from the other machines. A security professional could also intentionally place malware on a virtual machine to examine it in a more secure environment.

Note: Although using virtual machines is useful when investigating potentially infected machines or running malware in a constrained environment, there are still some risks. For example, a malicious program can escape virtualization and access the host machine. This is why you should never completely trust virtualized systems.

Efficiency

Using virtual machines can also be an efficient and convenient way to perform security tasks. You can open multiple virtual machines at once and switch easily between them. This allows you to streamline security tasks, such as testing and exploring various applications.

You can compare the efficiency of a virtual machine to a city bus. A single city bus has a lot of room and is an efficient way to transport many people simultaneously. If city buses didn’t exist, then everyone on the bus would have to drive their own cars. This uses more gas, cars, and other resources than riding the city bus.

Similar to how many people can ride one bus, many virtual machines can be hosted on the same physical machine. That way, separate physical machines aren't needed to perform certain tasks.

Managing virtual machines

Virtual machines can be managed with a software called a hypervisor. Hypervisors help users manage multiple virtual machines and connect the virtual and physical hardware. Hypervisors also help with allocating the shared resources of the physical host machine to one or more virtual machines.

One hypervisor that is useful for you to be familiar with is the Kernel-based Virtual Machine (KVM). KVM is an open-source hypervisor that is supported by most major Linux distributions. It is built into the Linux kernel, which means it can be used to create virtual machines on any machine running a Linux operating system without the need for additional software.

Other forms of virtualization

In addition to virtual machines, there are other forms of virtualization. Some of these virtualization technologies do not use operating systems. For example, multiple virtual servers can be created from a single physical server. Virtual networks can also be created to more efficiently use the hardware of a physical network.

Key takeaways

Virtual machines are virtual versions of physical computers and are one example of virtualization. Virtualization is a key technology in the security industry, and it’s important for security analysts to understand the basics. There are many benefits to using virtual machines, such as isolation of malware and other security risks. However, it’s important to remember there’s still a risk of malicious software escaping their virtualized environments.

The command line in use

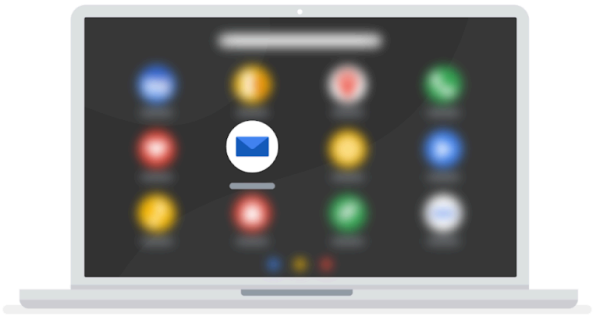
Previously, you explored graphical user interfaces (GUI) and command-line interfaces (CLI). In this reading, you'll compare these two interfaces and learn more about how they're used in cybersecurity.

CLI vs. GUI

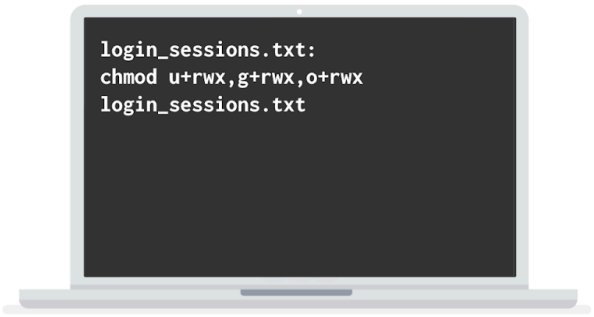
A graphical user interface (GUI) is a user interface that uses icons on the screen to manage different tasks on the computer. A command-line interface (CLI) is a text-based user interface that uses commands to interact with the computer.

Display

One notable difference between these two interfaces is how they appear on the screen. A GUI has graphics and icons, such as the icons on your desktop or taskbar for launching programs. In contrast, a CLI only has text. It looks similar to lines of code.



Graphical user interface
(GUI)



Command-line interface
(CLI)

Function

These two interfaces also differ in how they function. A GUI is an interface that only allows you to make one request at a time. However, a CLI allows you to make multiple requests at a time.

Advantages of a CLI in cybersecurity

The choice between using a GUI or CLI is partly based on personal preference, but security analysts should be able to use both interfaces. Using a CLI can provide certain advantages.

Efficiency

Some prefer the CLI because it can be used more quickly when you know how to manage this interface. For a new user, a GUI might be more efficient because they're easier for beginners to navigate.

Because a CLI can accept multiple requests at one time, it's more powerful when you need to perform multiple tasks efficiently. For example, if you had to create multiple new files in your system, you could quickly perform this task in a CLI. If you were using a GUI, this could take much longer, because you have to repeat the same steps for each new file.

History file

For security analysts, using the Linux CLI is helpful because it records a history file of all the commands and actions in the CLI. If you were using a GUI, your actions are not necessarily saved in a history file.

For example, you might be in a situation where you're responding to an incident using a playbook. The playbook's instructions require you to run a series of different commands. If you used a CLI, you'd be able to go back to the history and ensure all of the commands were correctly used. This could be helpful if there were issues using the playbook and you had to review the steps you performed in the command line.

Additionally, if you suspect an attacker has compromised your system, you might be able to trace their actions using the history file.

Key takeaways

GUIs and CLIs are two types of user interfaces that security analysts should be familiar with. There are multiple differences between a GUI and a CLI, including their displays and how they function. When working in cybersecurity, a CLI is often preferred over a GUI because it can handle multiple tasks simultaneously and it includes a history file.

Glossary terms from module 1

Terms and definitions from Course 4, Module 1

Application: A program that performs a specific task

Basic Input/Output System (BIOS): A microchip that contains loading instructions for the computer and is prevalent in older systems

Bootloader: A software program that boots the operating system

Command-line interface (CLI): A text-based user interface that uses commands to interact with the computer

Graphical user interface (GUI): A user interface that uses icons on the screen to manage different tasks on the computer

Hardware: The physical components of a computer

Legacy operating system: An operating system that is outdated but still being used

Operating system (OS): The interface between computer hardware and the user

Random Access Memory (RAM): A hardware component used for short-term memory

Unified Extensible Firmware Interface (UEFI): A microchip that contains loading instructions for the computer and replaces BIOS on more modern systems

User interface: A program that allows the user to control the functions of the operating system

Virtual machine (VM): A virtual version of a physical computer.

Module 2

Linux architecture explained

Understanding the Linux architecture is important for a security analyst. When you understand how a system is organized, it makes it easier to understand how it functions. In this reading, you'll learn more about the individual components in the Linux architecture. A request to complete a task starts with the user and then flows through applications, the shell, the Filesystem Hierarchy Standard, the kernel, and the hardware.

User

The user is the person interacting with a computer. They initiate and manage computer tasks. Linux is a multi-user system, which means that multiple users can use the same resources at the same time.

Applications

An application is a program that performs a specific task. There are many different applications on your computer. Some applications typically come pre-installed on your computer, such as calculators or calendars. Other applications might have to be installed, such as some web browsers or email clients. In Linux, you'll often use a package manager to install applications. A package manager is a tool that helps users install, manage, and remove packages or applications. A package is a piece of software that can be combined with other packages to form an application.

Shell

The shell is the command-line interpreter. Everything entered into the shell is text based. The shell allows users to give commands to the kernel and receive responses from it. You can think of the shell as a translator between you and your computer. The

shell translates the commands you enter so that the computer can perform the tasks you want.

Filesystem Hierarchy Standard (FHS)

The Filesystem Hierarchy Standard (FHS) is the component of the Linux OS that organizes data. It specifies the location where data is stored in the operating system.

A directory is a file that organizes where other files are stored. Directories are sometimes called “folders,” and they can contain files or other directories. The FHS defines how directories, directory contents, and other storage is organized so the operating system knows where to find specific data.

Kernel

The kernel is the component of the Linux OS that manages processes and memory. It communicates with the applications to route commands. The Linux kernel is unique to the Linux OS and is critical for allocating resources in the system. The kernel controls all major functions of the hardware, which can help get tasks expedited more efficiently.

Hardware

The hardware is the physical components of a computer. You might be familiar with some hardware components, such as hard drives or CPUs. Hardware is categorized as either peripheral or internal.

Peripheral devices

Peripheral devices are hardware components that are attached and controlled by the computer system. They are not core components needed to run the computer system. Peripheral devices can be added or removed freely. Examples of peripheral devices include monitors, printers, the keyboard, and the mouse.

Internal hardware

Internal hardware are the components required to run the computer. Internal hardware includes a main circuit board and all components attached to it. This main circuit board is also called the motherboard. Internal hardware includes the following:

- The Central Processing Unit (CPU) is a computer's main processor, which is used to perform general computing tasks on a computer. The CPU executes the instructions provided by programs, which enables these programs to run.
- Random Access Memory (RAM) is a hardware component used for short-term memory. It's where data is stored temporarily as you perform tasks on your computer. For example, if you're writing a report on your computer, the data needed for this is stored in RAM. After you've finished writing the report and closed down that program, this data is deleted from RAM. Information in RAM cannot be accessed once the computer has been turned off. The CPU takes the data from RAM to run programs.
- The hard drive is a hardware component used for long-term memory. It's where programs and files are stored for the computer to access later. Information on the hard drive can be accessed even after a computer has been turned off and on again. A computer can have multiple hard drives.

Key takeaways

It's important for security analysts to understand the Linux architecture and how these components are organized. The components of the Linux architecture are the user, applications, shell, Filesystem Hierarchy Standard, kernel, and hardware. Each of these components is important in how Linux functions.

More Linux distributions

Previously, you were introduced to the different distributions of Linux. This included KALI LINUX ™. (KALI LINUX ™ is a trademark of OffSec.) In addition to KALI LINUX ™, there are multiple other Linux distributions that security analysts should be familiar with. In this reading, you'll learn about additional Linux distributions.

KALI LINUX ™

KALI LINUX ™ is an open-source distribution of Linux that is widely used in the security industry. This is because KALI LINUX ™, which is Debian-based, is pre-installed with many useful tools for penetration testing and digital forensics. A penetration test is a simulated attack that helps identify vulnerabilities in systems, networks, websites, applications, and processes. Digital forensics is the practice of collecting and analyzing data to determine what has happened after an attack. These are key activities in the security industry.

However, KALI LINUX ™ is not the only Linux distribution that is used in cybersecurity.

Ubuntu

Ubuntu is an open-source, user-friendly distribution that is widely used in security and other industries. It has both a command-line interface (CLI) and a graphical user interface (GUI). Ubuntu is also Debian-derived and includes common applications by default. Users can also download many more applications from a package manager, including security-focused tools. Because of its wide use, Ubuntu has an especially large number of community resources to support users.

Ubuntu is also widely used for cloud computing. As organizations migrate to cloud servers, cybersecurity work may more regularly involve Ubuntu derivatives.

Parrot

Parrot is an open-source distribution that is commonly used for security. Similar to KALI LINUX ™, Parrot comes with pre-installed tools related to penetration testing and digital forensics. Like both KALI LINUX ™ and Ubuntu, it is based on Debian.

Parrot is also considered to be a user-friendly Linux distribution. This is because it has a GUI that many find easy to navigate. This is in addition to Parrot's CLI.

Red Hat® Enterprise Linux®

Red Hat Enterprise Linux is a subscription-based distribution of Linux built for enterprise use. Red Hat is not free, which is a major difference from the previously mentioned distributions. Because it's built and supported for enterprise use, Red Hat also offers a dedicated support team for customers to call about issues.

AlmaLinux

AlmaLinux is a community-driven Linux distribution that was created as a stable replacement for CentOS. CentOS was an open-source distribution that is closely related to Red Hat, and its final stable release, CentOS 8, was in December 2021. CentOS used source code published by Red Hat to provide a similar platform. AlmaLinux is designed to be a drop-in replacement for CentOS 8. This ensures that applications and configurations that worked on CentOS will continue to function on AlmaLinux.

Key takeaways

KALI LINUX ™, Ubuntu, Parrot, Red Hat, and CentOS are all widely used Linux distributions. It's important for security analysts to be aware of these distributions that they might encounter in their career.

Package managers for installing applications

Previously, you learned about Linux distributions and that different distributions derive from different sources, such as Debian or Red Hat Enterprise Linux distribution. You were also introduced to package managers, and learned that Linux applications are commonly distributed through package managers. In this reading, you'll apply this knowledge to learn more about package managers.

Introduction to package managers

A package is a piece of software that can be combined with other packages to form an application. Some packages may be large enough to form applications on their own.

Packages contain the files necessary for an application to be installed. These files include dependencies, which are supplemental files used to run an application.

Package managers can help resolve any issues with dependencies and perform other management tasks. A package manager is a tool that helps users install, manage, and remove packages or applications. Linux uses multiple package managers.

Note: It’s important to use the most recent version of a package when possible. The most recent version has the most up-to-date bug fixes and security patches. These help keep your system more secure.

Types of package managers

Many commonly used Linux distributions are derived from the same parent distribution. For example, KALI LINUX ™, Ubuntu, and Parrot all come from Debian. CentOS comes from Red Hat.

This knowledge is useful when installing applications because certain package managers work with certain distributions. For example, the Red Hat Package Manager (RPM) can be used for Linux distributions derived from Red Hat, and package managers such as dpkg can be used for Linux distributions derived from Debian.

Different package managers typically use different file extensions. For example, Red Hat Package Manager (RPM) has files which use the `.rpm` file extension, such as `Package-Version-Release_Architecture.rpm`. Package managers for Debian-derived Linux distributions, such as dpkg, have files which use the `.deb` file extension, such as `Package_Version-Release_Architecture.deb`.

Package management tools

In addition to package managers like RPM and dpkg, there are also package management tools that allow you to easily work with packages through the shell. Package management tools are sometimes utilized instead of package managers because they allow users to more easily perform basic tasks, such as installing a new package. Two notable tools are the Advanced Package Tool (APT) and Yellowdog Updater Modified (YUM).

Advanced Package Tool (APT)

APT is a tool used with Debian-derived distributions. It is run from the command-line interface to manage, search, and install packages.

Yellowdog Updater Modified (YUM)

YUM is a tool used with Red Hat-derived distributions. It is run from the command-line interface to manage, search, and install packages. YUM works with `.rpm` files.

Key takeaways

A package is a piece of software that can be combined with other packages to form an application. Packages can be managed using a package manager. There are multiple package managers and package management tools for different Linux distributions. Package management tools allow users to easily work with packages through the shell.

Debian-derived Linux distributions use package managers like dpkg as well as package management tools like Advanced Package Tool (APT). Red Hat-derived distributions use the Red Hat Package Manager (RPM) or tools like Yellowdog Updater Modified (YUM).

Resources for completing Linux labs

Qwiklabs has updated their terms of services to include an age requirement of 18+ to use the platform, in order to comply with regulations in the US and EU. Learners without access to Qwiklabs are still able to complete the certification and gain the badge by reviewing the Qwiklab instructions, exemplars, and participating in other hands-on activities throughout the certificate. This participation is essential to understanding the certificate's concepts and preparing learners for graded assessments.

This course features hands-on lab activities where you'll have the opportunity to practice Linux commands in the terminal. You'll use a platform called Qwiklabs to complete these labs. In this reading, you'll learn how to use Qwiklabs.

This reading first provides a section on how to use Qwiklabs, which includes details on how to launch a lab, how to interact within the Qwiklabs environment, and how to end a lab. This is followed by another section on helpful navigation tips and keyboard shortcuts; these may be useful when working in the terminal.

Note: You will not launch Qwiklabs directly from this reading and instead will do this through lab activities and exemplars that you encounter throughout the course.

How to use Qwiklabs

Launching Qwiklabs

When you select a lab, you start from a Coursera page. You will need to click Launch App on that page. After you click Launch App, a new tab will open with a Qwiklabs page that contains instructions for that particular lab.

Start Lab button

On the Qwiklabs page, you must click Start Lab to open a temporary terminal. The instructions for the lab will move to the right side of the screen.

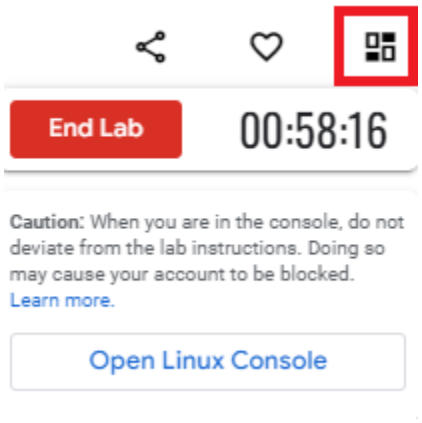
Read the instructions and complete all the tasks in the lab by entering commands in the terminal.

Note: It may take a moment for the terminal to start.

Lab control dialog box

After you click Start Lab, the lab control dialog box opens. It contains the End Lab button, the timer, and the Open Linux Console button.

You can hide or unhide the dialog box by clicking the following icon in the red box:



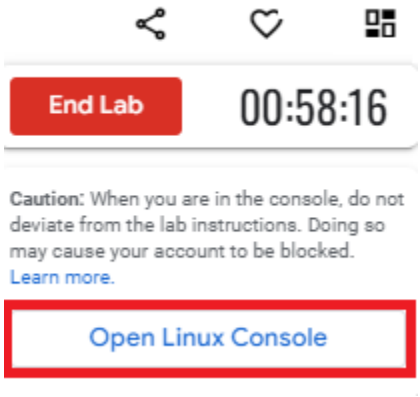
The timer

The timer starts when the terminal has loaded. The timer keeps track of the amount of time you have left to complete a lab. The timer counts down until it reaches 00:00:00. When it does, your temporary terminal and resources are deleted.

You will have ample time to complete the labs. But, stay focused on completing the tasks to ensure you use your time well.

Open Linux Console button

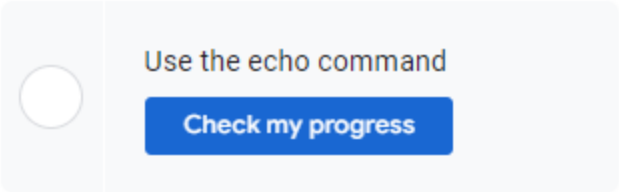
When you click the button to Open Linux Console, the terminal opens in a new browser window:



Use this feature if you want a full-screen view of the terminal. You can close this window at any time. Closing the window does not end your lab, and you can continue working in the terminal in the original tab.

Check progress

You can check your progress by clicking Check my progress at the end of each task.



If you haven't yet completed a task, you'll receive hints on what you must do to complete it.

You can click Check my progress whenever you want to check the completion status of a task or receive a hint.

Using copy/paste commands

The first time you try to use copy or paste keyboard shortcuts (such as CTRL + C), you'll receive a pop-up requesting permission to use your device's clipboard: "googlecoursera.qwiklabs.com wants to see text and images copied to the clipboard." Please click Allow if you would like to be able to use these shortcuts in the Qwiklabs platform. If you choose not to allow Qwiklabs access to your clipboard, you cannot use keyboard shortcuts but you can still complete the lab.

Code block

Certain steps may include a code block. Click the copy button to copy the code provided and then paste it into the terminal.

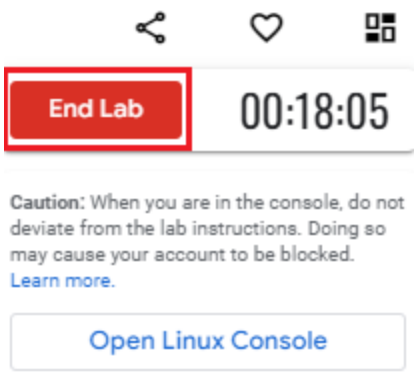
To paste code or other text content that you have copied from the instructions into the terminal, activate the terminal by clicking anywhere inside it. The terminal is active when the cursor in the terminal changes from a static empty outline to a flashing solid block.

Once the terminal is active, use the keyboard shortcut CTRL + V (hold down the CTRL key and press the V key) to insert the copied text into the terminal at the location of the flashing cursor.

Scrolling

In certain situations, you may want to scroll within the terminal window. To do so, use the scroll wheel on your mouse or the touchpad of your computer.

End Lab button



Finally, click End Lab when you've completed the tasks in the lab.

Note: Don't click End Lab until you're finished; you'll lose access to the work you've done throughout the lab.

Tracking progress on Coursera

If you complete a lab but your progress hasn't been tracked on Coursera, you may need to refresh the page for your progress to be registered. Once you complete the lab and refresh the page, the green check mark should appear.

Helpful navigation tips and keyboard shortcuts

The following contains a list of navigation tips and keyboard shortcuts you may find useful when completing your Linux labs. Your cursor must be in the terminal window to use these navigation tips and keyboard shortcuts.

- `CTRL + C`: Terminates a command that is currently running; from the instructions portion of Qwiklabs, you can use `CTRL + C` to copy, but within the terminal, it will only terminate a command and if one isn't running, it will display `^C` at the prompt
- `CTRL + V`: Pastes text
- `clear`: Clears the terminal screen; this can also be done by entering `CTRL + L`
- `CTRL + A`: Sets your cursor at the beginning of a command
- `CTRL + E`: Sets your cursor at the end of a command
- `Left arrow` key: Moves left within a command
- `Right arrow` key: Moves right within a command
- `Up arrow` key: Provides the last command you entered into the command line; can be entered multiple times to go through multiple commands from the command history
- `Down arrow` key: Provides the next command in the command history; must be after using the `up arrow` key
- `Tab` key: Provides available suggestions for completing your text

Key takeaways

Knowing how to navigate Qwiklabs will be useful as you complete the labs throughout this course. These labs can help you practice what you've learned in an interactive environment.

Lab tips and troubleshooting steps

Qwiklabs has updated their terms of services to include an age requirement of 18+ to use the platform, in order to comply with regulations in the US and EU. Learners without access to Qwiklabs are still able to complete the certification and gain the badge by reviewing the Qwiklab instructions, exemplars, and participating in other hands-on activities throughout the certificate. This participation is essential to understanding the certificate's concepts and preparing learners for graded assessments.

Throughout this certificate you will use Qwiklabs and Jupyter Notebooks to complete hands-on activities that include Linux command line, packet capture, and Python programming tasks. In this reading, we will cover some tips and troubleshooting steps for using Qwiklabs and Jupyter Notebooks on your computer.

Browser compatibility

Make sure your internet browser is updated regularly. Qwiklabs and Jupyter Notebooks require the latest version of Google Chrome, Firefox, or Microsoft Edge. If your browser is outdated or you are using a browser that is not supported by Qwiklabs or Jupyter Notebooks, you may encounter a problem. If your browser is up to date and you are using one of the browsers listed above and still encountering problems try restarting

your browser or clearing your browser's cache and cookies. You can also use incognito mode which prevents your browser from storing cookies and other temporary data.

Note: The Qwiklabs user interface works best with Google Chrome.

Internet connection

Qwiklabs and Jupyter Notebooks require a stable internet connection. If you are experiencing problems starting or completing Qwiklabs or Jupyter Notebooks, your internet connection may be slow or unreliable. Some signs of an unstable internet connection may be freezing labs, difficulty connecting to virtual machines, or the inability to type or enter commands within the lab environment.

Pro Tip: If you are unable to complete a Qwiklab or Jupyter Notebooks lab on one device, try using another device.

Troubleshooting steps

To summarize, here are the troubleshooting steps to try if you encounter a problem with Qwiklabs or Jupyter Notebooks.

1. Make sure you are using the latest version of a supported browser: Google Chrome, Firefox, or Microsoft Edge.
2. Restart your browser and clear your browser's cache and cookies. You can also use incognito mode.
3. Check your internet connection and make sure it is stable. You can try restarting your router and modem to regain a stable connection.
4. Try restarting Qwiklabs or Jupyter Notebooks again.
5. For Qwiklabs only: If problems persist or you receive a message stating that you have exceeded the quota for a Qwiklab, submit this form to Qwiklabs support for assistance.

Activity: Install software in a Linux distribution

Introduction

In this lab, you'll learn how to install and uninstall applications in Linux. You'll use Linux commands in the Bash shell to complete this lab. You'll also use the Advanced Package Tool (APT) package manager to install and uninstall the Suricata and tcpdump applications.

Disclaimer: For optimal performance and compatibility, it is recommended to use either Google Chrome or Mozilla Firefox browsers while accessing the labs.

What you'll do

You have multiple tasks in this lab:

- Confirm APT is installed in Bash
- Install Suricata with APT
- Uninstall Suricata with APT
- Install tcpdump with APT
- Reinstall Suricata with APT

Lab instructions

Start the lab

Before you start, you can review the Resources for completing Linux labs. Then from this page, click Launch App. A Qwiklabs page will open and from that page, click Start Lab to begin the activity!

You may attempt this lab a maximum of 5 times, and you will have 60 minutes to complete this lab during each attempt.

End the lab

From within the lab, click End Lab to end your lab.

Additionally, sometimes you need to refresh your Coursera page in order for your progress to be registered. If you refresh this page after you complete your lab, the green check mark should appear.

Best practices for completing labs:

- Make sure your browser is up to date with the latest version.
- Make sure your internet connection is stable.
- After you complete the lab, leave the lab window open for at least 10 minutes in order to allow the system to record your progress.
- If you run into issues connecting to the lab, try logging into Coursera in an Incognito mode and completing the lab there.
- Review Lab tips and troubleshooting steps for more information.

This course uses a third-party app, Activity: Install software in a Linux distribution, to enhance your learning experience. The app will reference basic information like your name, email, and Coursera ID.

Exemplar: Install software in a Linux distribution

Activity overview

In this lab activity, you used the Advanced Package Tool (APT) and sudo to install and uninstall applications in a Linux Bash shell.

While installing Linux applications can be a complex task, the APT package manager manages most of this complexity for you and allows you to quickly and reliably manage the applications in a Linux environment.

You used Suricata and tcpdump as an example. These are network security applications that can be used to capture and analyze network traffic.

The virtual machine you accessed in this lab has a Debian-based distribution of Linux running, and that works with the APT package manager. Using a virtual machine prevents damage to a system in the event its tools are used improperly. It also gives you the ability to revert to a previous state.

As a security analyst, it's likely you'll need to know how to install and manage applications on a Linux operating system. In this lab activity, you'll learn how to do exactly that!

This exemplar is a walkthrough of the previous Qwiklab activity, including detailed instructions and solutions. You may use this exemplar if you were unable to complete the lab and/or you need extra guidance in competing lab tasks. You may also refer to this exemplar to prepare for the graded quiz in this module.

Scenario

Your role as a security analyst requires that you have the Suricata and tcpdump network security applications installed on your system.

In this scenario, you have to install, uninstall, and reinstall these applications on your Linux Bash shell. You also need to confirm that you've installed them correctly.

Here's how you'll do this: First, you'll confirm that APT is installed on your Linux Bash shell. Next, you'll use APT to install the Suricata application and confirm that it is

installed. Then, you'll uninstall the Suricata application and confirm this as well. Next, you'll install the tcpdump application and list the applications currently installed. Finally, you'll reinstall the Suricata application and confirm that both applications are installed.

OK, it's time to learn how to install some applications!

Note: The lab starts with your user account, called analyst, already logged in to the Bash shell. This means you can start with the tasks as soon as you click the Start Lab button.

Task 1. Ensure that APT is installed

First, you'll check that the APT application is installed so that you can use it to manage applications. The simplest way to do this is to run the apt command in the Bash shell and check the response.

The Bash shell is the command-line interpreter currently open on the left side of the screen. You'll use the Bash shell by typing commands after the prompt. The prompt is represented by a dollar sign ($) followed by the input cursor.

- Confirm that the APT package manager is installed in your Linux environment. To do this, type `apt` after the command-line prompt and press ENTER.

The command to complete this step:

```
1
```

When installed, `apt` displays basic usage information when you run it. This includes the version information and a description of the tool:

```
1
2
3
```

```
4
5
6
7
8
9
```

APT is already installed by default in the Linux Bash shell in this lab because this is a Debian-based system. APT is also the recommended package manager for Debian. If you're using another distribution, a different package manager, such as YUM, may be available instead.

Task 2. Install and uninstall the Suricata application

In this task, you must install Suricata, a network analysis tool used for intrusion detection, and verify that it installed correctly. Then, you'll uninstall the application.

1. Use the APT package manager to install the Suricata application.

Type `sudo apt install suricata` after the command-line prompt and press ENTER.

The command to complete this step:

```
1
```

Note: The `apt install` *and* `apt remove` *commands must be prefixed with the* `sudo` *command as elevated privileges are required to install and uninstall software in Linux.*

The Suricata application can take a few minutes to install.

When you install an application with APT, the output displays details of all the software to be installed. This may include additional applications that depend on the new software. These additional applications are called the dependencies of the software to be installed.

When prompted to continue, press the ENTER key to respond with the default response. (In this case, the default response is Yes.)

1. Verify that Suricata is installed by running the newly installed application.

Type `suricata` after the command-line prompt and press ENTER.

The command to complete this step:

```
1
```

When Suricata is installed, version and usage information is listed:

```
1
2
3
4
5
6
```

3. Use the APT package manager to uninstall Suricata.

Type `sudo apt remove suricata` after the command-line prompt and press ENTER. Press ENTER (Yes) when prompted to continue.

The command to complete this step:

```
1
```

When prompted to continue, press the ENTER key to respond with the default response. (In this case, the default response is Yes.)

4. Verify that Suricata has been uninstalled by running the application command again.

Type `suricata` after the command-line prompt and press ENTER.

The command to complete this step:

```
1
```

If you have uninstalled Suricata, the output is an error message:

```
1
```

This message indicates that Suricata can't be found anymore.

Task 3. Install the tcpdump application

In this task, you must install the tcpdump application. This is a command-line tool that can be used to capture network traffic in a Linux Bash shell.

- Use the APT package manager to install tcpdump.

Type `sudo apt install tcpdump` after the command-line prompt and press ENTER.

The command to complete this step:

```
1
```

Task 4. List the installed applications

Next, you need to confirm that you've installed the required applications. It's important to be able to validate that the correct applications are installed. Often you may want to check that the correct versions are installed as well.

1. Use the APT package manager to list all installed applications.

Type `apt list --installed` after the command-line prompt and press ENTER.

The command to complete this step:

```
1
```

This produces a long list of applications because Linux has a lot of software installed by default.

2. Search through the list to find the tcpdump application you installed.

The Suricata application is not listed because you installed and then uninstalled that application:

```
1
```

```
2
```

```
3
```

Note: The specific version of `tcpdump` *that you see displayed may be different from what is shown above.*

Task 5. Reinstall the Suricata application

In this task, you must reinstall the Suricata application and verify that it has installed correctly.

1. Run the command to install the Suricata application.

Type `sudo apt install suricata` after the command-line prompt and press ENTER.

The command to complete this step:

```
1
```

When prompted to continue, press the ENTER key to respond with the default response. (In this case, the default response is Yes.)

2. Use the APT package manager to list the installed applications.

Type `apt list --installed` after the command-line prompt and press ENTER.

The command to complete this step:

```
1
```

3. Search through the list to confirm that the Suricata application has been installed.

The output should include the following lines:

```
1
2
3
4
5
```

Conclusion

Great work!

You now have practical experience with the APT package manager. You learned to

- install applications,
- uninstall applications, and
- list installed applications.

Being able to manage installed applications in Linux is a key skill for any security analyst.

Different types of shells

Knowing how to work with Linux shells is an important skill for cybersecurity professionals. Shells can be used for many common tasks. Previously, you were introduced to shells and their functions. This reading will review shells and introduce you to different types, including the one that you'll use in this course.

Communicate through a shell

As you explored previously, the shell is the command-line interpreter. You can think of a shell as a translator between you and the computer system. Shells allow you to give commands to the computer and receive responses from it. When you enter a command into a shell, the shell executes many internal processes to interpret your command, send it to the kernel, and return your results.

Types of shells

The many different types of Linux shells include the following:

- Bourne-Again Shell (bash)
- C Shell (csh)
- Korn Shell (ksh)
- Enhanced C shell (tcsh)
- Z Shell (zsh)

All Linux shells use common Linux commands, but they can differ in other features. For example, ksh and bash use the dollar sign (`$`) to indicate where users type in their commands. Other shells, such as zsh, use the percent sign (`%`) for this purpose.

Bash

Bash is the default shell in most Linux distributions. It's considered a user-friendly shell. You can use bash for basic Linux commands as well as larger projects.

Bash is also the most popular shell in the cybersecurity profession. You'll use bash throughout this course as you learn and practice Linux commands.

Key takeaways

Shells are a fundamental part of the Linux operating system. Shells allow you to give commands to the computer and receive responses from it. They can be thought of as a translator between you and your computer system. There are many different types of shells, but the bash shell is the most commonly used shell in the cybersecurity profession. You'll learn how to enter Linux commands through the bash shell later in this course.

Exemplar: Examine input and output in the shell

Activity overview

Previously, you discussed how the Bash shell helps you communicate with a computer’s operating system.

When you communicate with the shell, the commands in the shell can take input and return output or error messages.

In this lab activity, you’ll use the echo command to examine how input is received and how output is returned in the shell. Next, you’ll use the expr command to further explore input and output while performing some basic calculations in the shell.

This activity will build foundations in understanding how you communicate with the Linux operating system through the shell. As a security analyst, you'll need to input commands into the shell and recognize when the shell returns either output or an error message.

Next, you'll explore the scenario!

This exemplar is a walkthrough of the previous Qwiklab activity, including detailed instructions and solutions. You may use this exemplar if you were unable to complete the lab and/or you need extra guidance in competing lab tasks. You may also refer to this exemplar to prepare for the graded quiz in this module.

Scenario

As a security professional, it’s important to understand the concept of communicating with your computer via the shell.

In this scenario, you have to input a specified string of text that you want the shell to return as output. You'll also need to input a few mathematical calculations so the OS (operating system) can return the result.

Here’s how you’ll do this: First, you’ll use the `echo` command to generate some output in the shell. Second, you’ll use the `expr` command to perform basic mathematical calculations. Next, you’ll use the `clear` command to clear the Bash shell window. Finally, you’ll have an opportunity to explore the `echo` and `expr` commands further.

Get ready to examine input and output in the Bash shell!

Task 1. Generate output with the echo command

The `echo` command in the Bash shell outputs a specified string of text. In this task, you’ll use the `echo` command to generate output in the Bash shell.

1. Type `echo hello` into the shell and press ENTER.

The command to complete this step:

```
1
```

The `hello` string should be returned:

```
1
```

The command `echo hello` is the input to the shell, and `hello` is the output from the shell.

1. Rerun the command, but include quotation marks around the string data. Type `echo "hello"` into the shell and press ENTER.

The command to complete this step:

```
1
```

The `hello` string should be returned again:

```
1
```

Note: The output is the same as before. The quotation marks are optional in this case, but they tell the shell to group a series of characters together. This can be useful if you need to pass a string that contains certain characters that might be otherwise misinterpreted by the command.

3. Use the `echo` command to output your name to the shell.

Type `echo "name"` into the shell, replacing `"name"` with your own name, and press ENTER.

The command to complete this step:

```
1
```

The name you've entered as the string should return as the output.

Task 2. Generate output with the expr command

In this task, you'll use the expr command to generate some additional output in the Bash shell. The expr command performs basic mathematical calculations and can be useful when you need to quickly perform a calculation.

Imagine that the system has shown you that you have 32 alerts, but only 8 required action. You want to calculate how many alerts are false positives so that you can provide feedback to the team that configures the alerts.

To do this, you need to subtract the number of alerts that required action from the total number of alerts.

1. Calculate the number of false positives using the `expr` command.

Type `expr 32 - 8` into the shell and press ENTER.

The command to complete this step:

The following result should be returned:

1

Note: The `expr` *command requires that all terms and operators in an expression are separated by spaces. For example:* `expr 32 - 8`*, and not* `expr 32-8`*.*

Now, you need to calculate the average number of login attempts that are expected over the course of a year. From the information you have, you know that an average of 3500 login attempts have been made each month so far this year.

So, you should be able to calculate the total number of logins expected in a year by multiplying 3500 by 12.

2. Type `expr 3500 * 12` into the shell and press ENTER.

The command to complete this step:

1

The correct result should now be returned:

1

Task 3. Clear the Bash shell

In this task, you'll use the `clear` command to clear the Bash shell of all existing output. This allows you to start with the cursor at the top of the Bash shell window.

When you work in a shell environment, the screen can fill with previous input and output data. This can make it difficult to process what you're working on. Clearing the screen allows you to create a clutter-free text environment to allow you to focus on what is important at that point in time.

- Type `clear` into the shell and press ENTER.

The command to complete this step:

```
1
```

Note: All previous commands and output will be cleared, and the user prompt and cursor will return to the upper left of the shell window.

Optional task: Perform more calculations with the expr command

You have the opportunity to explore input and output further using the echo and expr commands.

1. Generate at least one new output using the `echo` command.

(Remember the `echo "hello"` output you generated).

The command to complete this step:

```
1
```

2. Perform at least one new calculation using the `expr` command.

The mathematical operators you can use with the `expr` command for adding, subtracting, dividing, and multiplying are +, -, / and *.

Note: The `expr` *command performs integer mathematical calculations only, so you cannot use the decimal point or expect a fractional result. All results are rounded down to the nearest integer. Also, all terms and operators in an expression need to be separated by spaces. For example:* `expr 25 + 15`*, and not* `expr 25+15`*.*

Conclusion

Great work!

You now have practical experience in using basic Linux Bash shell commands to

- generate output with the `echo` command,
- generate output with the `expr` command, and
- clear the Bash shell with the `clear` command.

Understanding input and output is essential when communicating through the shell. It's important that you're comfortable with these basic concepts before you go on to work with additional commands.

Glossary terms from module 2

Terms and definitions from Course 4, Module 2

Application: A program that performs a specific task

Bash: The default shell in most Linux distributions

CentOS: An open-source distribution that is closely related to Red Hat

Central Processing Unit (CPU): A computer’s main processor, which is used to perform general computing tasks on a computer

Command: An instruction telling the computer to do something

Digital forensics: The practice of collecting and analyzing data to determine what has happened after an attack

Directory: A file that organizes where other files are stored

Distributions: The different versions of Linux

File path: The location of a file or directory

Filesystem Hierarchy Standard (FHS): The component of the Linux OS that organizes data

Graphical user interface (GUI): A user interface that uses icons on the screen to manage different tasks on the computer

Hard drive: A hardware component used for long-term memory

Hardware: The physical components of a computer

Internal hardware: The components required to run the computer

Kali Linux ™: An open-source distribution of Linux that is widely used in the security industry

Kernel: The component of the Linux OS that manages processes and memory

Linux: An open source operating system

Package: A piece of software that can be combined with other packages to form an application

Package manager: A tool that helps users install, manage, and remove packages or applications

Parrot: An open-source distribution that is commonly used for security

Penetration test (pen test): A simulated attack that helps identify vulnerabilities in systems, networks, websites, applications, and processes

Peripheral devices: Hardware components that are attached and controlled by the computer system

Random Access Memory (RAM): A hardware component used for short-term memory

Red Hat® Enterprise Linux® (also referred to simply as Red Hat in this course): A subscription-based distribution of Linux built for enterprise use

Shell: The command-line interpreter

Standard error: An error message returned by the OS through the shell

Standard input: Information received by the OS via the command line

Standard output: Information returned by the OS through the shell

String data: Data consisting of an ordered sequence of characters

Ubuntu: An open-source, user-friendly distribution that is widely used in security and other industries

User: The person interacting with a computer

Module 3

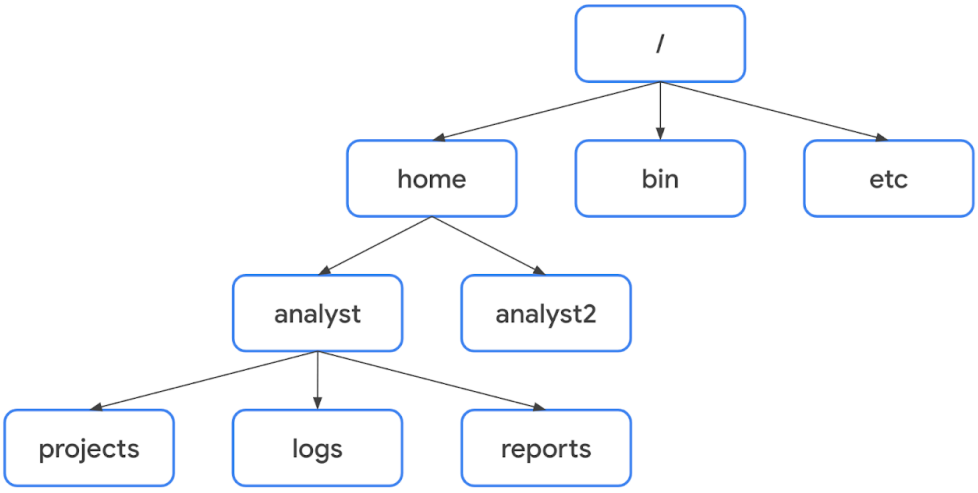
Navigate Linux and read file content

In this reading, you'll review how to navigate the file system using Linux commands in Bash. You'll further explore the organization of the Linux Filesystem Hierarchy Standard, review several common Linux commands for navigation and reading file content, and learn a couple of new commands.

Filesystem Hierarchy Standard (FHS)

Previously, you learned that the Filesystem Hierarchy Standard (FHS) is the component of Linux that organizes data. The FHS is important because it defines how directories, directory contents, and other storage is organized in the operating system.

This diagram illustrates the hierarchy of relationships under the FHS:



Under the FHS, a file's location can be described by a file path. A file path is the location of a file or directory. In the file path, the different levels of the hierarchy are separated by a forward slash (`/`).

Root directory

The root directory is the highest-level directory in Linux, and it's always represented with a forward slash (`/`). All subdirectories branch off the root directory. Subdirectories can continue branching out to as many levels as necessary.

Standard FHS directories

Directly below the root directory, you'll find standard FHS directories. In the diagram, **`home`**, **`bin`**, and **`etc`** are standard FHS directories. Here are a few examples of what standard directories contain:

- `/home`: Each user in the system gets their own home directory.
- `/bin`: This directory stands for “binary” and contains binary files and other executables. Executables are files that contain a series of commands a computer needs to follow to run programs and perform other functions.
- `/etc`: This directory stores the system’s configuration files.
- `/tmp`: This directory stores many temporary files. The `/tmp` directory is commonly used by attackers because anyone in the system can modify data in these files.
- `/mnt`: This directory stands for “mount” and stores media, such as USB drives and hard drives.

Pro Tip: You can use the `man hier` command to learn more about the FHS and its standard directories.

User-specific subdirectories

Under `home` are subdirectories for specific users. In the diagram, these users are `analyst` and `analyst2`. Each user has their own personal subdirectories, such as `projects`, `logs`, or `reports`.

Note: When the path leads to a subdirectory below the user’s home directory, the user’s home directory can be represented as the tilde (`~`). For example, `/home/analyst/logs` can also be represented as `~/logs`.

You can navigate to specific subdirectories using their absolute or relative file paths. The absolute file path is the full file path, which starts from the root. For example, `/home/analyst/projects` is an absolute file path. The relative file path is the file path that starts from a user's current directory.

Note: Relative file paths can use a dot (`.`) to represent the current directory, or two dots (`..`) to represent the parent of the current directory. An example of a relative file path could be `../projects`.

Key commands for navigating the file system

The following Linux commands can be used to navigate the file system: `pwd`, `ls`, and `cd`.

pwd

The `pwd` command prints the working directory to the screen. Or in other words, it returns the directory that you're currently in.

The output gives you the absolute path to this directory. For example, if you're in your `home` directory and your username is `analyst`, entering `pwd` returns `/home/analyst`.

Pro Tip: To learn what your username is, use the `whoami` command. The `whoami` command returns the username of the current user. For example, if your username is `analyst`, entering `whoami` returns `analyst`.

ls

The `ls` command displays the names of the files and directories in the current working directory. For example, in the video, `ls` returned directories such as `logs`, and a file called `updates.txt`.

Note: If you want to return the contents of a directory that's not your current working directory, you can add an argument after `ls` with the absolute or relative file path to the desired directory. For example, if you're in the `/home/analyst` directory but want to list the contents of its `projects` subdirectory, you can enter `ls /home/analyst/projects` or just `ls projects`.

cd

The `cd` command navigates between directories. When you need to change directories, you should use this command.

To navigate to a subdirectory of the current directory, you can add an argument after `cd` with the subdirectory name. For example, if you're in the `/home/analyst` directory and want to navigate to its `projects` subdirectory, you can enter `cd projects`.

You can also navigate to any specific directory by entering the absolute file path. For example, if you're in `/home/analyst/projects`, entering `cd /home/analyst/logs` changes your current directory to `/home/analyst/logs`.

Pro Tip: You can use the relative file path and enter `cd ..` to go up one level in the file structure. For example, if the current directory is `/home/analyst/projects`, entering `cd ..` would change your working directory to `/home/analyst`.

Common commands for reading file content

The following Linux commands are useful for reading file content: `cat`, `head`, `tail`, and `less`.

cat

The `cat` command displays the content of a file. For example, entering `cat updates.txt` returns everything in the `updates.txt` file.

head

The `head` command displays just the beginning of a file, by default 10 lines. The `head` command can be useful when you want to know the basic contents of a file but don't need the full contents. Entering `head updates.txt` returns only the first 10 lines of the `updates.txt` file.

Pro Tip: If you want to change the number of lines returned by `head`, you can specify the number of lines by including `-n`. For example, if you only want to display the first five lines of the `updates.txt` file, enter `head -n 5 updates.txt`.

tail

The `tail` command does the opposite of `head`. This command can be used to display just the end of a file, by default 10 lines. Entering `tail updates.txt` returns only the last 10 lines of the `updates.txt` file.

Pro Tip: You can use `tail` to read the most recent information in a log file.

less

The `less` command returns the content of a file one page at a time. For example, entering `less updates.txt` changes the terminal window to display the contents of `updates.txt` one page at a time. This allows you to easily move forward and backward through the content.

Once you've accessed your content with the `less` command, you can use several keyboard controls to move through the file:

- `Space bar`: Move forward one page
- `b`: Move back one page
- `Down arrow`: Move forward one line
- `Up arrow`: Move back one line
- `q`: Quit and return to the previous terminal window

Key takeaways

It's important for security analysts to be able to navigate Linux and the file system of the FHS. Some key commands for navigating the file system include `pwd`, `ls`, and `cd`. Reading file content is also an important skill in the security profession. This can be done with commands such as `cat`, `head`, `tail`, and `less`.

Exemplar: Find files with Linux commands

Activity overview

Previously, you learned about Linux and how to communicate with the OS through the shell. You also learned how to use some of the core commands to navigate the Linux file system and read content from files it contains.

These are essential skills. For example, when investigating unauthorized access, you might navigate to and then read a user access report.

In this lab activity, you'll navigate a Linux file structure, locate files, and read the contents of files. You'll also need to answer a few multiple-choice questions based on the information contained in these files.

As a security analyst, it's key that you know how to navigate, manage, and analyze files remotely via a Linux shell without a graphical user interface.

This exemplar is a walkthrough of the previous Qwiklab activity, including detailed instructions and solutions. You may use this exemplar if you were unable to complete the lab and/or you need extra guidance in competing lab tasks. You may also refer to this exemplar to prepare for the graded quiz in this module.

Scenario

In this scenario, you have to locate and analyze the information of certain files located in the `/home/analyst` directory.

Here's how you'll do this: First, you'll get the information of the current working directory you're in and display the contents of the directory. Second, you'll navigate to the **`reports`** directory and list the subdirectories it contains. Third, you'll navigate to the **`users`** subdirectory and display the contents of the **`Q1_added_users.txt file`**. Finally, you'll navigate to the **`logs`** directory and display the first 10 lines of a file it contains.

To complete these tasks, you'll need to use commands that you've previously learned in this course. Well, it's time to practice what you've learned. Let's do this!

Task 1. Get the current directory information

In this task, you must use the commands you learned about to check the current working directory and list its contents.

1. Display your working directory.

The command to complete this step:

```
1 pwd
```

This will show that your current working directory is your home directory.

```
1 /home/analyst
```

2. Display the names of the files and directories in the current working directory.

The command to complete this step:

```
1 ls
```

The output should be:

```
1 logs projects reports temp
```

Which directory is your current working directory?

Answer: The lab starts with /home/analyst as your current working directory.

How many directories does the current working directory contain?

Answer: The lab starts with four subdirectories in the **`/home/analystdirectory`**, namely **`logs`**, **`notes`**, **`temp`**, and **`reports`**.

Task 2. Change directory and list the subdirectories

In this task, you must navigate to a new directory and determine the subdirectories it contains.

1. Navigate to the **`/home/analyst/reports`** directory.

The command to complete this step using a relative path:

```
1 cd reports
```

Note: The **`cd`** *command accepts absolute and relative paths. An absolute path includes all the directories from the root of the file system and starts with a* **`/`***. An alternative is a relative path, which is expressed starting from the current directory and starts without the initial* **`/`***. The above command uses a relative path.*

The command to complete this step using an absolute path:

```
1 cd /home/analyst/reports
```

2. Display the files and subdirectories in the **`/home/analyst/reports`** directory.

The command to complete this step:

```
1 ls
```

The output should be:

```
1 users
```

What is the name of the subdirectory in the /home/analyst/reports directory?

Answer: The subdirectory contained in the **`/home/analyst/reports`** directory is called **`users`**.

Task 3. Locate and read the contents of a file

In this task, you must navigate to a subdirectory and read the contents of a file it contains.

1. Navigate to the **`/home/analyst/reports/users`** directory.

The command to complete this step:

```
1 cd /home/analyst/reports/users
```

The above command uses an absolute path. You could also use a relative path as follows:

```
1 cd users
```

2. List the files in the current directory.

The command to complete this step:

```
1 ls
```

3. Display the contents of the **`Q1_added_users.txt file`**.

The command to complete this step:

```
1 cat Q1_added_users.txt
```

Note: The **`cat`** *command prints the contents of a file to the shell. You can specify the file to display using absolute or relative paths.*

The same command using an absolute path:

```
1 cat /home/analyst/reports/users/Q1_added_users.txt
```

What department does the employee with the username aezra work in?

Answer: The employee with username aezra works in the Human Resources department.

What is the employee_id of the user mreed in the Information Technology department?

Answer: The employee_id of the employee with username mreed in the Information Technology department is 1104.

Task 4. Navigate to a directory and locate a file

In this task, you must navigate to a new directory, locate a file, and examine the contents of the file.

1. Navigate to the **`/home/analyst/logs`** directory.

The command to complete this step:

```
1 cd /home/analyst/logs
```

2. Display the name of the file it contains.

The command to complete this step:

```
1 ls
```

This command will display the following output:

```
1 server_logs.txt
```

3. Display the first 10 lines of this file.

The command to complete this step:

```
1 head server_logs.txt
```

Note: The `head` *command displays just the beginning of a file, by default ten lines. You can specify how many lines to display using the* `-n` *argument, which specifies the number of lines to display.*

How many warning messages are in the first 10 lines of the server_logs.txt file?

Answer: There are three warning messages in the first 10 lines of the `server_logs.txt` file.

Conclusion

Great work!

You now have practical experience in using basic Linux Bash shell commands to

- navigate directory structures with the `cd` command,
- display the current working directory with the `pwd` command,
- list the contents of a directory with the `ls` command, and
- display the contents of files with the `cat` and `head` commands.

Navigating through directories and reading file contents are fundamental skills that you'll often use when communicating through the shell.

Filter content in Linux

In this reading, you'll continue exploring Linux commands, which can help you filter for the information you need. You'll learn a new Linux command, `find`, which can help you search files and directories for specific information.

Filtering for information

You previously explored how filtering for information is an important skill for security analysts. Filtering is selecting data that match a certain condition. For example, if you had a virus in your system that only affected the `.txt` files, you could use filtering to find these files quickly. Filtering allows you to search based on specific criteria, such as file extension or a string of text.

grep

The grep command searches a specified file and returns all lines in the file containing a specified string or text. The grep command commonly takes two arguments: a specific string to search for and a specific file to search through.

For example, entering grep OS updates.txt returns all lines containing OS in the updates.txt file. In this example, OS is the specific string to search for, and updates.txt is the specific file to search through.

Let's look at another example: grep error time_logs.txt. Here grep is used to search for the text pattern. error is the term you are looking for in the time_logs.txt file. When you run this command, grep will scan the time_logs.txt file and print only the lines containing the word error.

Piping

The pipe command is accessed using the pipe character (`|`). Piping sends the standard output of one command as standard input to another command for further processing. As a reminder, standard output is information returned by the OS through the shell, and standard input is information received by the OS via the command line.

The pipe character (`|`) is located in various places on a keyboard. On many keyboards, it's located on the same key as the backslash character (`\`). On some keyboards, the `|` can look different and have a small space through the middle of the line. If you can't find the `|`, search online for its location on your particular keyboard.

When used with **`grep`**, the pipe can help you find directories and files containing a specific word in their names. For example, **`ls /home/analyst/reports | grep users`** returns the file and directory names in the **`reports`** directory that contain **`users`**. Before the pipe, **`ls`** indicates to list the names of the files and directories in **`reports`**. Then, it sends this output to the command after the pipe. In this case, **`grep users`** returns all of the file or directory names containing **`users`** from the input it received.

Note: Piping is a general form of redirection in Linux and can be used for multiple tasks other than filtering. You can think of piping as a general tool that you can use whenever you want the output of one command to become the input of another command.

find

The **`find`** command searches for directories and files that meet specified criteria. There's a wide range of criteria that can be specified with **`find`**. For example, you can search for files and directories that

- Contain a specific string in the name,
- Are a certain file size, or
- Were last modified within a certain time frame.

When using **`find`**, the first argument after **`find`** indicates where to start searching. For example, entering **`find /home/analyst/projects`** searches for everything starting at the **`projects`** directory.

After this first argument, you need to indicate your criteria for the search. If you don't include a specific search criteria with your second argument, your search will likely return a lot of directories and files.

Specifying criteria involves options. Options modify the behavior of a command and commonly begin with a hyphen (`-`).

-name and -iname

One key criteria analysts might use with `find` is to find file or directory names that contain a specific string. The specific string you're searching for must be entered in quotes after the `-name` or `-iname` options. The difference between these two options is that `-name` is case-sensitive, and `-iname` is not.

For example, you might want to find all files in the `projects` directory that contain the word "log" in the file name. To do this, you'd enter `find /home/analyst/projects -name "*log*"`. You could also enter `find /home/analyst/projects -iname "*log*"`.

In these examples, the output would be all files in the `projects` directory that contain `log` surrounded by zero or more characters. The `"*log*"` portion of the command is the search criteria that indicates to search for the string "log". When `-name` is the option, files with names that include `Log` or `LOG`, for example, wouldn't be returned because this option is case-sensitive. However, they would be returned when `-iname` is the option.

Note: An asterisk (`*`) is used as a wildcard to represent zero or more unknown characters.

-mtime

Security analysts might also use `find` to find files or directories last modified within a certain time frame. The `-mtime` option can be used for this search. For example, entering `find /home/analyst/projects -mtime -3` returns all files and directories in the `projects` directory that have been modified within the past three days.

The `-mtime` option search is based on days, so entering `-mtime +1` indicates all files or directories last modified more than one day ago, and entering `-mtime -1` indicates all files or directories last modified less than one day ago.

Note: The option `-mmin` can be used instead of `-mtime` if you want to base the search on minutes rather than days.

Key takeaways

Filtering for information using Linux commands is an important skill for security analysts so that they can customize data to fit their needs. Three key Linux commands for this are `grep`, piping (`|`), and `find`. These commands can be used to navigate and filter for information in the file system.

- Consider the privacy and security implications of using AI. Consider how using AI tools may affect the security of other people or organizations.

Exemplar: Filter with grep

Activity overview

Previously, you learned about tools that you can use to filter information in Linux. You're also familiar with the basic commands to navigate the Linux file system by now.

In this lab activity, you'll use the `grep` command and piping to search for files and to return specific information from files.

As a security analyst, it's key to know how to find the information you need. The ability to search for specific strings can help you locate what you need more efficiently.

Scenario

In this scenario, you need to obtain information contained in server log and user data files. You also need to find files with specific names.

Here's how you'll do this: First, you'll navigate to the `logs` directory and return the error messages in the `server_logs.txt` file. Next, you'll navigate to the `users` directory and search for files that contain a specific string in their names. Finally, you'll search for information contained in user files.

With that in mind, you're ready to practice what you've learned.

Task 1. Search for error messages in a log file

In this task, you must navigate to the `/home/analyst/logs` directory and report on the error messages in the `server_logs.txt` file. You'll do this by using grep to search the file and output only the entries that are for errors.

1. Navigate to the `/home/analyst/logs` directory.

The command to complete this step:

```
1 cd logs
```

2. Use `grep` to filter the `server_logs.txt` file, and return all lines containing the text string `error`.

Note: If you enter a command incorrectly and it fails to return to the command-line prompt, you can press CTRL+C to stop the process and force the shell to return to the command-line prompt.

The command to complete this step:

```
1 grep error server_logs.txt
```

This grep command will filter `server_logs.txt` file, and return a list of the lines that match the text string `error`.

Note: The first argument passed to `grep` *is the string you're searching for, and the second argument is the name of the file you're searching through.*

How many error lines are there in the server_logs.txt file?

Answer: There are six entries in the server_logs.txt file that include the error string.

Task 2. Find files containing specific strings

In this task, you must navigate to the `/home/analyst/reports/users` directory and use the correct Linux commands and arguments to search for user data files that contain a specific string in their names.

1. Navigate to the `/home/analyst/reports/users` directory.

The command to complete this step:

```
1 cd /home/analyst/reports/users
```

2. Using the pipe character (`|`), pipe the output of the `ls` command to the `grep` command to list only the files containing the string `Q1` in their names.

The command to complete this step:

```
1 ls | grep Q1
```

How many files in the /home/analyst/reports/users subdirectory contain “Q1” in their names?

Answer: There are three files in the `reports/users` directory that have `Q1` in their names.

Note: Piping sends the standard output of one command to the standard input of another command for further processing. In the example, the output of the `grep` *command is piped to the* `ls` *command and the output displayed in the shell.*

3. List the files that contain the word `access` in their names.

The command to complete this step:

```
1 ls | grep access
```

How many files in the /home/analyst/reports/users directory contain “access” in their names?

Answer: There are four files in the reports/users directory that have the text string access in their names.

Task 3. Search more file contents

In this task, you must search for information contained in user files and report on users that were added and deleted from the system.

1. Display the files in the **`/home/analyst/reports/users`** directory.

The command to complete this step:

```
1 ls
```

2. Search the **`Q2_deleted_users.txt`** file for the username **`jhill`**.

The command to complete this step:

```
1 grep jhill Q2_deleted_users.txt
```

Did you find the username jhill in the Q2_deleted_users.txt file?

Answer: Yes, the user **`jhill`** is listed in the **`Q2_deleted_users.txt`** file.

3. Search the **`Q4_added_users.txt`** file to list the users who were added to the **`Human Resources`** department.

The command to complete this step:

```
1 grep "Human Resources" Q4_added_users.txt
```

Manage directories and files

Previously, you explored how to manage the file system using Linux commands. The following commands were introduced: `mkdir`, `rmdir`, `touch`, `rm`, `mv`, and `cp`. In this reading, you'll review these commands, the nano text editor, and learn another way to write to files.

Creating and modifying directories

mkdir

The `mkdir` command creates a new directory. Like all of the commands presented in this reading, you can either provide the new directory as the absolute file path, which starts from the root, or as a relative file path, which starts from your current directory.

For example, if you want to create a new directory called `network` in your `/home/analyst/logs` directory, you can enter `mkdir /home/analyst/logs/network` to create this new directory. If you're already in the `/home/analyst/logs` directory, you can also create this new directory by entering `mkdir network`.

Pro Tip: You can use the `ls` command to confirm the new directory was added.

rmdir

The `rmdir` command removes, or deletes, a directory. For example, entering `rmdir /home/analyst/logs/network` would remove this empty directory from the file system.

Note: The `rmdir` command cannot delete directories with files or subdirectories inside. For example, entering `rmdir /home/analyst` returns an error message.

Creating and modifying files

touch and rm

The `touch` command creates a new file. This file won't have any content inside. If your current directory is `/home/analyst/reports`, entering `touch permissions.txt` creates a new file in the `reports` subdirectory called `permissions.txt`.

The `rm` command removes, or deletes, a file. This command should be used carefully because it's not easy to recover files deleted with `rm`. To remove the permissions file you just created, enter `rm permissions.txt`.

Pro Tip: You can verify that `permissions.txt` was successfully created or removed by entering `ls`.

mv and cp

You can also use `mv` and `cp` when working with files. The `mv` command moves a file or directory to a new location, and the `cp` command copies a file or directory into a new location. The first argument after `mv` or `cp` is the file or directory you want to move or copy, and the second argument is the location you want to move or copy it to.

To move `permissions.txt` into the `logs` subdirectory, enter `mv permissions.txt /home/analyst/logs`. Moving a file removes the file from its original location. However, copying a file doesn't remove it from its original location. To copy `permissions.txt` into the `logs` subdirectory while also keeping it in its original location, enter `cp permissions.txt /home/analyst/logs`.

Note: The `mv` command can also be used to rename files. To rename a file, pass the new name in as the second argument instead of the new location. For example, entering `mv permissions.txt perm.txt` renames the `permissions.txt` file to `perm.txt`.

nano text editor

nano is a command-line file editor that is available by default in many Linux distributions. Many beginners find it easy to use, and it's widely used in the security profession. You can perform multiple basic tasks in nano, such as creating new files and modifying file contents.

To open an existing file in nano from the directory that contains it, enter `nano` followed by the file name. For example, entering `nano permissions.txt` from the `/home/analyst/reports` directory opens a new nano editing window with the `permissions.txt` file open for editing. You can also provide the absolute file path to the file if you're not in the directory that contains it.

You can also create a new file in nano by entering `nano` followed by a new file name. For example, entering `nano authorized_users.txt` from the `/home/analyst/reports` directory creates the `authorized_users.txt` file within that directory and opens it in a new nano editing window.

Since there isn't an auto-saving feature in nano, it's important to save your work before exiting. To save a file in nano, use the keyboard shortcut `Ctrl + O`. You'll be prompted to confirm the file name before saving. To exit out of nano, use the keyboard shortcut `Ctrl + X`.

Note: Vim and Emacs are also popular command-line text editors.

Standard output redirection

There's an additional way you can write to files. Previously, you learned about standard input and standard output. Standard input is information received by the OS via the command line, and standard output is information returned by the OS through the shell.

You've also learned about piping. Piping sends the standard output of one command as standard input to another command for further processing. It uses the pipe character (`|`).

In addition to the pipe (`|`), you can also use the right angle bracket (`>`) and double right angle bracket (`>>`) operators to redirect standard output.

When used with `echo`, the `>` and `>>` operators can be used to send the output of `echo` to a specified file rather than the screen. The difference between the two is that `>` overwrites your existing file, and `>>` adds your content to the end of the existing file instead of overwriting it. The `>` operator should be used carefully, because it's not easy to recover overwritten files.

When you're inside the directory containing the `permissions.txt` file, entering `echo "last updated date" >> permissions.txt` adds the string "last updated date" to the file contents. Entering `echo "time" > permissions.txt` after this command overwrites the entire file contents of `permissions.txt` with the string "time".

Note: Both the `>` and `>>` operators will create a new file if one doesn't already exist with your specified name.

Key takeaways

Knowing how to manage the file system in Linux is an important skill for security analysts. Useful commands for this include: `mkdir`, `rmdir`, `touch`, `rm`, `mv`, and `cp`. When security analysts need to write to files, they can use the nano text editor, or the `>` and `>>` operators.

Exemplar: Manage files with Linux commands

Activity overview

In this lab activity, you'll use Linux commands to modify a directory structure and the files it contains.

You'll also use the nano text editor to add text to a file.

You previously learned that directories help you organize subdirectories and files in Linux. As a security analyst, creating, removing, and editing directories and files are core tasks you'll need to perform to help you to manage data.

When data is well organized, you can more easily detect issues and keep data safe.

With that in mind, you’re now ready to practice what you've learned.

This exemplar is a walkthrough of the previous Qwiklab activity, including detailed instructions and solutions. You may use this exemplar if you were unable to complete the lab and/or you need extra guidance in competing lab tasks. You may also refer to this exemplar to prepare for the graded quiz in this module.

Scenario

In this scenario, you need to ensure that the `/home/analyst` directory is properly organized.

You have to make a few changes to the `/home/analyst` directory and the files it contains.

You also have to edit a file to record the changes or updates you make to the directory.

When you start, the `/home/analyst` directory contains the following subdirectories and files:

```
home
└── analyst
    ├── notes
    │   ├── Q3patches.txt
    │   └── tempnotes.txt
    ├── reports
    │   ├── Q1patches.txt
    │   └── Q2patches.txt
    └── temp
```

You need to modify the /home/analyst directory to the following directory and file structure:

```
1 home
2  └── analyst
3    ├── logs
4    ├── notes
5    │   └── tasks.txt
6    └── reports
7        ├── Q1patches.txt
8        └── Q2patches.txt
9        └── Q3patches.txt
```

Here's how you'll do this: First, you'll create a new subdirectory called logs in the **`/home/analyst`** directory. Next, you'll remove the temp subdirectory. Then, you'll move the **`Q3patches.txt`** file to the **`reports`** subdirectory and delete the **`tempnotes.txt`** file. Finally, you'll create a new **`.txt`** file called **`tasks`** in the **`notes`** subdirectory and add a note to the file describing the tasks you've performed.

You'll need to use the commands learned in the video lesson to complete these steps.

This might sound like quite a number of tasks to perform, but you'll be guided on how to do this.

Task 1. Create a new directory

First, you must create a dedicated subdirectory called **logs**, which will be used to store all future log files.

1. Create a new subdirectory called **logs** in the **/home/analyst** directory.

The command to complete this step:

```
1 mkdir logs
```

2. List the contents of the **/home/analyst** directory to confirm that you've successfully created the new **logs** subdirectory.

The command to complete this step:

```
1 ls
```

The output should list the original three directories and the new **logs** subdirectory:

```
1 logs notes reports temp
```

Task 2. Remove a directory

Next, you must remove the **temp** directory, as you'll no longer be placing items in it.

1. Remove the **/home/analyst/temp** directory.

The command to complete this step:

```
1 rmdir logs
```

2. List the contents of the **/home/analyst** directory to confirm that you have removed the **temp** subdirectory.

The command to complete this step:

```
1 ls
```

The **temp** directory should no longer be listed:

```
1 logs reports temp
```

Task 3. Move a file

The **Q3patches.txt** file contains notes taken on third-quarter patches and is now in the correct reporting format.

You must move the **Q3patches.txt** file from the **notes** directory to the **reports** directory.

1. Navigate to the **/home/analyst/notes** directory.

The command to complete this step:

```
1 cd /home/analyst/notes
```

The previous command used the absolute path, you could use the relative path as follows:

```
1 cd notes
```

2. Move the **Q3patches.txt** file from the **/home/analyst/notes** directory to the **/home/analyst/reports** directory.

The command to complete this step:

```
1 mv Q3patches.txt /home/analyst/reports/
```

3. List the contents of the **/home/analyst/reports** directory to confirm that you have moved the file successfully.

The command to complete this step:

```
1 ls /home/analyst/reports
```

When you list the contents of the **`reports`** directory, it should show that three quarterly report files are now in the **`reports`** directory:

```
1 Q1patches.txt Q2patches.txt Q3patches.txt
```

Task 4. Remove a file

Next, you must delete an unused file called **`tempnotes.txt`** from the **`/home/analyst/notes`** directory.

1. Remove the **`tempnotes.txt`** file from the **`/home/analyst/notes`** directory.

The command to complete this step:

```
1 rm tempnotes.txt
```

2. List the contents of the **`/home/analyst/notes`** directory to confirm that you've removed the file successfully.

The command to complete this step:

```
1 ls
```

Task 5. Create a new file

Now, you must create a file named **`tasks.txt`** in the **`/home/analyst/notes`** directory that you'll use to document completed tasks.

1. Use the **`touch`** command to create an empty file called **`tasks.txt`** in the **`/home/analyst/notes`** directory.

The command to complete this step:

```
1 touch task.txt
```

2. List the contents of the **`/home/analyst/notes`** directory to confirm that you have created a new file.

The command to complete this step:

```
1 ls
```

A file called **`tasks.txt`** should now exist in the notes directory:

```
1 task.txt
```

Task 6. Edit a file

Finally, you must use the nano text editor to edit the **`tasks.txt`** file and add a note describing the tasks you've completed.

1. Using the nano text editor, open the **`tasks.txt`** file that is located in the **`/home/analyst/notes`** directory.

The command to complete this step:

```
1 nano task.txt
```

Note: This action changes the shell from the normal Bash interface to the nano text editor interface.

2. Copy and paste the following text into the text input area of the nano editor:

```
1 Completed tasks

2  1. Managed file structure in /home/analyst
```

3. Press CTRL+X to exit the nano text editor.

This triggers a prompt asking Save modified bufferer?

4. Press Y to confirm that you want to save the new data to your file. (Answering "no" will discard changes.)

5. Press ENTER to confirm that File Name to Write is tasks.txt.

Note: The recommended sequence of commands for saving a file with the nano text editor is to use CTRL+O to tell nano to save the file and then use CTRL+X to exit immediately.

In this web-based lab environment, the CTRL+O command is intercepted by your web browser and is interpreted as a request to save the web page. The sequence used here is a commonly used alternative that achieves the same end result.

6. Use the **`clear`** command to clear the Bash shell window and remove any traces of the nano text input area.

The command to complete this step:

Note: Most Bash shells typically handle the screen cleanup after you exit nano. In this lab environment, nano sometimes leaves some text clutter around the edges of the screen that the **`clear`** *command cleans up for you.*

7. Display the contents of the **`tasks.txt`** file to confirm that it contains the updated task details.

```
1 cat task.txt
```

This file should now contain the contents of the **`tasks.txt`** file that you added and saved in previous steps:

```
1 Completed tasks

2  1. Managed file structure in /home/analyst
```

Conclusion

Great work!

You now have practical experience in using basic Linux Bash shell commands to

- create and remove directories,
- copy, move, and remove files, and
- edit files with the nano text editor.

You're well on your way to managing directories and files in a Linux environment!

Permission commands

Previously, you explored file permissions and the commands that you can use to display and change them. In this reading, you'll review these concepts and also focus on an example of how these commands work together when putting the principle of least privilege into practice.

Reading permissions

In Linux, permissions are represented with a 10-character string. Permissions include:

- read: for files, this is the ability to read the file contents; for directories, this is the ability to read all contents in the directory including both files and subdirectories

- write: for files, this is the ability to make modifications on the file contents; for directories, this is the ability to create new files in the directory
- execute: for files, this is the ability to execute the file if it's a program; for directories, this is the ability to enter the directory and access its files

These permissions are given to these types of owners:

- user: the owner of the file
- group: a larger group that the owner is a part of
- other: all other users on the system

Each character in the 10-character string conveys different information about these permissions. The following table describes the purpose of each character:

Character	Example	Meaning
1st	drwxrwxrwx	file type • `d` for directory

		• `-` for a regular file
2nd	drwxrwxrwx	read permissions for the user • `r` if the user has read permissions • `-` if the user lacks read permissions
3rd	drwxrwxrwx	write permissions for the user • `w` if the user has write permissions • `-` if the user lacks write permissions
4th	drwxrwxrwx	execute permissions for the user

		• `x` if the user has execute permissions • `-` if the user lacks execute permissions
5th	drwxrwxrwx	read permissions for the group • `r` if the group has read permissions • `-` if the group lacks read permissions
6th	drwxrwxrwx	write permissions for the group • `w` if the group has write permissions • `-` if the group lacks write permissions

7th	drwxrwxrwx	execute permissions for the group • `x` if the group has execute permissions • `-` if the group lacks execute permissions
8th	drwxrwxrwx	read permissions for other • `r` if the other owner type has read permissions • `-` if the other owner type lacks read permissions
9th	drwxrwxrwx	write permissions for other • `w` if the other owner type has write permissions

		• `-` if the other owner type lacks write permissions
10th	drwxrwxrwx	execute permissions for other • `x` if the other owner type has execute permissions • `-` if the other owner type lacks execute permissions

Exploring existing permissions

You can use the `ls` command to investigate who has permissions on files and directories. Previously, you learned that `ls` displays the names of files in directories in the current working directory.

There are additional options you can add to the `ls` command to make your command more specific. Some of these options provide details about permissions. Here are a few important `ls` options for security analysts:

- `ls -a`: Displays hidden files. Hidden files start with a period (`.`) at the beginning.
- `ls -l`: Displays permissions to files and directories. Also displays other additional information, including owner name, group, file size, and the time of last modification.
- `ls -la`: Displays permissions to files and directories, including hidden files. This is a combination of the other two options.

Changing permissions

The principle of least privilege is the concept of granting only the minimal access and authorization required to complete a task or function. In other words, users should not have privileges that are beyond what is necessary. Not following the principle of least privilege can create security risks.

The `chmod` command can help you manage this authorization. The `chmod` command changes permissions on files and directories.

Using chmod

The `chmod` command requires two arguments. The first argument indicates how to change permissions, and the second argument indicates the file or directory that you want to change permissions for. For example, the following command would add all permissions to `login_sessions.txt`:

```
chmod u+rwx,g+rwx,o+rwx login_sessions.txt
```

If you wanted to take all the permissions away, you could use

```
chmod u-rwx,g-rwx,o-rwx login_sessions.txt
```

Another way to assign these permissions is to use the equals sign (`=`) in this first argument. Using `=` with `chmod` sets, or assigns, the permissions exactly as specified. For example, the following command would set read permissions for `login_sessions.txt` for user, group, and other:

```
chmod u=r,g=r,o=r login_sessions.txt
```

This command overwrites existing permissions. For instance, if the user previously had write permissions, these write permissions are removed after you specify only read permissions with `=`.

The following table reviews how each character is used within the first argument of `chmod`:

Character	Description
`u`	indicates changes will be made to user permissions
`g`	indicates changes will be made to group permissions
`o`	indicates changes will be made to other permissions
`+`	adds permissions to the user, group, or other
`-`	removes permissions from the user, group, or other

`=`	assigns permissions for the user, group, or other

Note: When there are permission changes to more than one owner type, commas are needed to separate changes for each owner type. You should not add spaces after those commas.

The principle of least privilege in action

As a security analyst, you may encounter a situation like this one: There's a file called `bonuses.txt` within a compensation directory. The owner of this file is a member of the Human Resources department with a username of `hrrep1`. It has been decided that `hrrep1` needs access to this file. But, since this file contains confidential information, no one else in the `hr` group needs access.

You run `ls -l` to check the permissions of files in the compensation directory and discover that the permissions for `bonuses.txt` are `-rw-rw----`. The group owner type has read and write permissions that do not align with the principle of least privilege.

To remedy the situation, you input `chmod g-rw bonuses.txt`. Now, only the user who needs to access this file to carry out their job responsibilities can access this file.

Key takeaways

Managing directory and file permissions may be a part of your work as a security analyst. Using `ls` with the `-l` and `-la` options allows you to investigate directory and file permissions. Using `chmod` allows you to change user permissions and ensure they are aligned with the principle of least privilege.

Responsible use of sudo

Previously, you explored authorization, authentication, and Linux commands with `sudo`, `useradd`, and `userdel`. The `sudo` command is important for security analysts because it allows users to have elevated permissions without risking the system by running commands as the root user. You'll continue exploring authorization, authentication, and Linux commands in this reading and learn two more commands that can be used with `sudo`: `usermod` and `chown`.

Responsible use of sudo

To manage authorization and authentication, you need to be a root user, or a user with elevated privileges to modify the system. The root user can also be called the "super user." You become a root user by logging in as the root user. However, running commands as the root user is not recommended in Linux because it can create security

risks if malicious actors compromise that account. It's also easy to make irreversible mistakes, and the system can't track who ran a command. For these reasons, rather than logging in as the root user, it's recommended you use `sudo` in Linux when you need elevated privileges.

The `sudo` command temporarily grants elevated permissions to specific users. The name of this command comes from "super user do." Users must be given access in a configuration file to use `sudo`. This file is called the "sudoers file." Although using `sudo` is preferable to logging in as the root user, it's important to be aware that users with the elevated permissions to use `sudo` might be more at risk in the event of an attack.

You can compare this to a hotel with a master key. The master key can be used to access any room in the hotel. There are some workers at the hotel who need this key to perform their work. For example, to clean all the rooms, the janitor would scan their ID badge and then use this master key. However, if someone outside the hotel's network gained access to the janitor's ID badge and master key, they could access any room in the hotel. In this example, the janitor with the master key represents a user using `sudo` for elevated privileges. Because of the dangers of `sudo`, only users who really need to use it should have these permissions.

Additionally, even if you need access to `sudo`, you should be careful about using it with only the commands you need and nothing more. Running commands with `sudo` allows users to bypass the typical security controls that are in place to prevent elevated access to an attacker.

Note: Be aware of `sudo` if copying commands from an online source. It's important you don't use `sudo` accidentally.

Authentication and authorization with sudo

You can use `sudo` with many authentication and authorization management tasks. As a reminder, authentication is the process of verifying who someone is, and authorization is the concept of granting access to specific resources in a system. Some of the key commands used for these tasks include the following:

useradd

The `useradd` command adds a user to the system. To add a user with the username of `fgarcia` with `sudo`, enter `sudo useradd fgarcia`. There are additional options you can use with `useradd`:

- `-g`: Sets the user's default group, also called their primary group
- `-G`: Adds the user to additional groups, also called supplemental or secondary groups

To use the `-g` option, the primary group must be specified after `-g`. For example, entering `sudo useradd -g security fgarcia` adds `fgarcia` as a new user and assigns their primary group to be `security`.

To use the `-G` option, the supplemental group must be passed into the command after `-G`. You can add more than one supplemental group at a time with the `-G` option. Entering `sudo useradd -G finance,admin fgarcia` adds `fgarcia` as a new user and adds them to the existing `finance` and `admin` groups.

usermod

The `usermod` command modifies existing user accounts. The same `-g` and `-G` options from the `useradd` command can be used with `usermod` if a user already exists.

To change the primary group of an existing user, you need the `-g` option. For example, entering `sudo usermod -g executive fgarcia` would change `fgarcia`'s primary group to the `executive` group.

To add a supplemental group for an existing user, you need the `-G` option. You also need a `-a` option, which appends the user to an existing group and is only used with the `-G` option. For example, entering `sudo usermod -a -G marketing fgarcia` would add the existing `fgarcia` user to the supplemental `marketing` group.

Note: When changing the supplemental group of an existing user, if you don't include the `-a` option, `-G` will replace any existing supplemental groups with the groups specified after `usermod`. Using `-a` with `-G` ensures that the new groups are added but existing groups are not replaced.

There are other options you can use with `usermod` to specify how you want to modify the user, including:

- `-d`: Changes the user's home directory.
- `-l`: Changes the user's login name.
- `-L`: Locks the account so the user can't log in.

The option always goes after the `usermod` command. For example, to change `fgarcia`'s home directory to `/home/garcia_f`, enter `sudo usermod -d /home/garcia_f fgarcia`. The option `-d` directly follows the command `usermod` before the other two needed arguments.

userdel

The `userdel` command deletes a user from the system. For example, entering `sudo userdel fgarcia` deletes `fgarcia` as a user. Be careful before you delete a user using this command.

The `userdel` command doesn't delete the files in the user's home directory unless you use the `-r` option. Entering `sudo userdel -r fgarcia` would delete `fgarcia` as a user and delete all files in their home directory. Before deleting any user files, you should ensure you have backups in case you need them later.

Note: Instead of deleting the user, you could consider deactivating their account with `usermod -L`. This prevents the user from logging in while still giving you access to their account and associated permissions. For example, if a user left an organization, this option would allow you to identify which files they have ownership over, so you could move this ownership to other users.

chown

The `chown` command changes ownership of a file or directory. You can use `chown` to change user or group ownership. To change the user owner of the `access.txt` file to `fgarcia`, enter `sudo chown fgarcia access.txt`. To change the group owner of `access.txt` to `security`, enter `sudo chown :security access.txt`. You must enter a colon (`:`) before `security` to designate it as a group name.

Similar to `useradd`, `usermod`, and `userdel`, there are additional options that can be used with `chown`.

Key takeaways

Authentication is the process of a user verifying their identity, and authorization is the process of determining what they have access to. You can use the `sudo` command to temporarily run commands with elevated privileges to complete authentication and authorization management tasks. Specifically, `useradd`, `userdel`, `usermod`, and `chown` can be used to manage users and file ownership.

Linux resources

Previously, you were introduced to the Linux community and some resources that exist to help Linux users. Linux has many options available to give users the information they need. This reading will review these resources. When you're aware of the resources available to you, you can continue to learn Linux independently. You can also discover even more ways that Linux can support your work as a security analyst.

Linux community

Linux has a large online community, and this is a huge resource for Linux users of all levels. You can likely find the answers to your questions with a simple online search. Troubleshooting issues by searching and reading online is an effective way to discover how others approached your issue. It's also a great way for beginners to learn more about Linux.

The UNIX and Linux Stack Exchange is a trusted resource for troubleshooting Linux issues. The Unix and Linux Stack Exchange is a question and answer website where community members can ask and answer questions about Linux. Community members vote on answers, so the higher quality answers are displayed at the top. Many of the questions are related to specific topics from advanced users, and the topics might help you troubleshoot issues as you continue using Linux.

Integrated Linux support

Linux also has several commands that you can use for support.

man

The `man` command displays information on other commands and how they work. It's short for "manual." To search for information on a command, enter the command after `man`. For example, entering `man chown` returns detailed information about `chown`, including the various options you can use with it. The output of the `man` command is also called a "man page."

apropos

The `apropos` command searches the man page descriptions for a specified string. Man pages can be lengthy and difficult to search through if you're looking for a specific keyword. To use `apropos`, enter the keyword after `apropos`.

You can also include the `-a` option to search for multiple words. For example, entering `apropos -a graph editor` outputs man pages that contain both the words "graph" and "editor" in their descriptions.

whatis

The `whatis` command displays a description of a command on a single line. For example, entering `whatis nano` outputs the description of `nano`. This command is useful when you don't need a detailed description, just a general idea of the command. This might be as a reminder. Or, it might be after you discover a new command through a colleague or online resource and want to know more.

Key takeaways

There are many resources available for troubleshooting issues or getting support for Linux. Linux has a large global community of users who ask and answer questions on online resources, such as the Unix and Linux Stack Exchange. You can also use integrated support commands in Linux, such as `man`, `apropos`, and `whatis`.

Resources for more information

There are many resources available online that can help you learn new Linux concepts, review topics, or ask and answer questions with the global Linux community. The Unix and Linux Stack Exchange is one example, and you can search online to find others.

Reference guide: Linux

The Linux reference guide contains key Linux commands security professionals use to perform basic job duties. The reference guide is divided into six different categories of useful Linux commands for security-related tasks:

- Navigate the file system
- Read files
- Manage the file system
- Filter content
- Manage users and their permissions
- Get help in Linux

Within each category, commands are organized alphabetically.

Access and save the guide

You can save a copy of this guide for future reference. You can use it as a resource for additional practice or in your future professional projects.

To access a downloadable version of this course item, click the following link and select *Use Template*.

Reference guide: Linux

OR

If you don’t have a Google account, you can download the item directly from the following attachment.

https://www.coursera.org/api/rest/v1/asset/download/pdf/81hFzUEIRj6pVYXS1nU_Nw?pageStart=&pageEnd=

Glossary terms from module 3

Terms and definitions from Course 4, Module 3

Absolute file path: The full file path, which starts from the root

Argument (Linux): Specific information needed by a command

Authentication: The process of verifying who someone is

Authorization: The concept of granting access to specific resources in a system

Bash: The default shell in most Linux distributions

Command: An instruction telling the computer to do something

File path: The location of a file or directory

Filesystem Hierarchy Standard (FHS): The component of the Linux OS that organizes data

Filtering: Selecting data that match a certain condition

nano: A command-line file editor that is available by default in many Linux distributions

Options: Input that modifies the behavior of a command

Permissions: The type of access granted for a file or directory

Principle of least privilege: The concept of granting only the minimal access and authorization required to complete a task or function

Relative file path: A file path that starts from the user's current directory

Root directory: The highest-level directory in Linux

Root user (or superuser): A user with elevated privileges to modify the system

Standard input: Information received by the OS via the command line

Standard output: Information returned by the OS through the shell

Module 4

SQL filtering versus Linux filtering

In this reading, you'll explore the differences between the two tools as they relate to filtering. You'll also learn that one way to access SQL is through the Linux command line.

Accessing SQL

There are many interfaces for accessing SQL and many different versions of SQL. One way to access SQL is through the Linux command line.

To access SQL from Linux, you need to type in a command for the version of SQL that you want to use. For example, if you want to access SQLite, you can enter the command sqlite3 in the command line.

After this, any commands typed in the command line will be directed to SQL instead of Linux commands.

Differences between Linux and SQL filtering

Although both Linux and SQL allow you to filter through data, there are some differences that affect which one you should choose.

Purpose

Linux filters data in the context of files and directories on a computer system. It's used for tasks like searching for specific files, manipulating file permissions, or managing processes.

SQL is used to filter data within a database management system. It's used for querying and manipulating data stored in tables and retrieving specific information based on defined criteria.

Syntax

Linux uses various commands and command-line options specific to each filtering tool. Syntax varies depending on the tool and purpose. Some examples of Linux commands are find, sed, cut, e grep

SQL uses the Structured Query Language (SQL), a standardized language with specific keywords and clauses for filtering data across different SQL databases. Some examples of SQL keywords and clauses are WHERE, SELECT, JOIN

Structure

SQL offers a lot more structure than Linux, which is more free-form and not as tidy.

For example, if you wanted to access a log of employee log-in attempts, SQL would have each record separated into columns. Linux would print the data as a line of text without this organization. As a result, selecting a specific column to analyze would be easier and more efficient in SQL.

In terms of structure, SQL provides results that are more easily readable and that can be adjusted more quickly than when using Linux.

Joining tables

Some security-related decisions require information from different tables. SQL allows the analyst to join multiple tables together when returning data. Linux doesn’t have that same functionality; it doesn’t allow data to be connected to other information on your computer. This is more restrictive for an analyst going through security logs.

Best uses

As a security analyst, it's important to understand when you can use which tool. Although SQL has a more organized structure and allows you to join tables, this doesn't mean that there aren't situations that would require you to filter data in Linux.

A lot of data used in cybersecurity will be stored in a database format that works with SQL. However, other logs might be in a format that is not compatible with SQL. For instance, if the data is stored in a text file, you cannot search through it with SQL. In those cases, it is useful to know how to filter in Linux.

Key takeaways

Linux filtering focuses on managing files and directories on a system, while SQL filtering focuses on structured data manipulation within databases. To work with SQL, you can access it from multiple different interfaces, such as the Linux command line. Both SQL and Linux allow you to filter for specific data, but SQL offers the advantages of structuring the data and allowing you to join data from multiple tables.

Query a database

In this reading, you'll review those basic SQL queries and learn a new keyword that will help you organize your output. You'll also learn about the `Chinook` database, which this course uses for queries in readings and quizzes.

Basic SQL query

There are two essential keywords in any SQL query: `SELECT` and `FROM`. You will use these keywords every time you want to query a SQL database. Using them together helps SQL identify what data you need from a database and the table you are returning it from.

The video demonstrated this SQL query:

```
SELECT employee_id, device_id

FROM employees;
```

In readings and quizzes, this course uses a sample database called the `Chinook` database to run queries. The `Chinook` database includes data that might be created at a digital media company. A security analyst employed by this company might need to query this data. For example, the database contains eleven tables, including an `employees` table, a `customers` table, and an `invoices` table. These tables include data such as names and addresses.

SELECT

The `SELECT` keyword indicates which columns to return. For example, you can return the `customerid` column from the `Chinook` database with

```
SELECT customerid
```

You can also select multiple columns by separating them with a comma. For example, if you want to return both the `customerid` and `city` columns, you should write `SELECT customerid, city`.

If you want to return all columns in a table, you can follow the `SELECT` keyword with an asterisk (`*`). The first line in the query will be `SELECT *`.

Note: Although the tables you're querying in this course are relatively small, using `SELECT *` may not be advisable when working with large databases and tables; in those cases, the final output may be difficult to understand and might be slow to run.

FROM

The `SELECT` keyword always comes with the `FROM` keyword. `FROM` indicates which table to query. To use the `FROM` keyword, you should write it after the `SELECT` keyword, often on a new line, and follow it with the name of the table you're querying. If you want to return all columns from the `customers` table, you can write:

```
SELECT *
FROM customers;
```

When you want to end the query here, you put a semicolon (`;`) at the end to tell SQL that this is the entire query.

Note: Line breaks are not necessary in SQL queries, but are often used to make the query easier to understand. If you prefer, you can also write the previous query on one line as

```
SELECT * FROM customers;
```

ORDER BY

Database tables are often very complicated, and this is where other SQL keywords come in handy. `ORDER BY` is an important keyword for organizing the data you extract from a table.

`ORDER BY` sequences the records returned by a query based on a specified column or columns. This can be in either ascending or descending order.

Sorting in ascending order

To use the `ORDER BY` keyword, write it at the end of the query and specify a column to base the sort on.

Sorting in descending order

You can also use the `ORDER BY` with the `DESC` keyword to sort in descending order. The `DESC` keyword is short for "descending" and tells SQL to sort numbers from largest to smallest, or alphabetically from Z to A. This can be done by following `ORDER BY` with the `DESC` keyword.

Sorting based on multiple columns

You can also choose multiple columns to order by. For example, you might first choose the `country` and then the `city` column. SQL then sorts the output by `country`, and for rows with the same `country`, it sorts them based on `city`.

Key takeaways

`SELECT` and `FROM` are important keywords in SQL queries. You use `SELECT` to indicate which columns to return and `FROM` to indicate which table to query. You can also include `ORDER BY` in your query to organize the output. These foundational SQL skills will support you as you move into more advanced queries.

Resources for completing SQL labs

This course features hands-on lab activities where you'll have the opportunity to practice using SQL queries in the terminal. You'll use a platform called Qwiklabs to complete these labs. In this reading, you'll learn how to use Qwiklabs.

This reading first provides a section on how to use Qwiklabs, which includes details on how to launch a lab, how to interact within the Qwiklabs environment, and how to end a lab. This is followed by another section on helpful navigation tips and keyboard shortcuts; these may be useful when working in the terminal.

Note: You will not launch Qwiklabs directly from this reading and instead will do this through lab activities and exemplars that you encounter throughout the course.

How to use Qwiklabs

Launching Qwiklabs

When you select a lab, you start from a Coursera page. You will need to click Launch App on that page. After you click Launch App, a new tab will open with a Qwiklabs page that contains instructions for that particular lab.

Start Lab button

On the Qwiklabs page, you must click Start Lab to open a temporary terminal. The instructions for the lab will move to the right side of the screen.

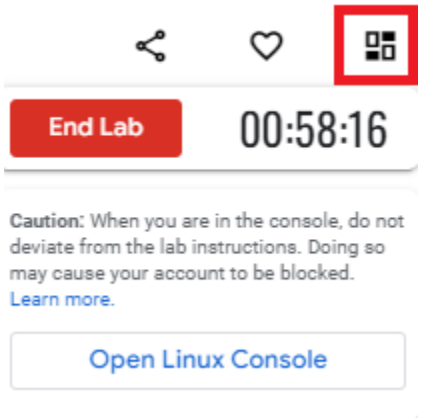
Read the instructions and complete all the tasks in the lab by entering commands in the terminal.

Note: It may take a moment for the terminal to start.

Lab control dialog box

After you click Start Lab, the lab control dialog box opens. It contains the End Lab button, the timer, and the Open Linux Console button.

You can hide or unhide the dialog box by clicking the following icon in the red box:



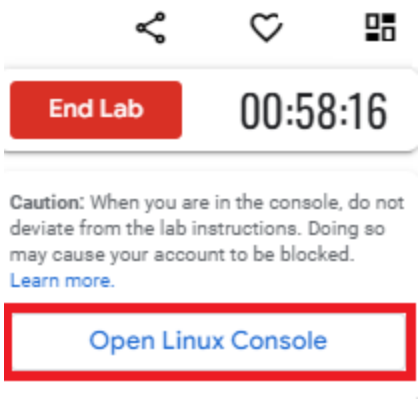
The timer

The timer starts when the terminal has loaded. The timer keeps track of the amount of time you have left to complete a lab. The timer counts down until it reaches 00:00:00. When it does, your temporary terminal and resources are deleted.

You will have ample time to complete the labs. But, stay focused on completing the tasks to ensure you use your time well.

Open Linux Console button

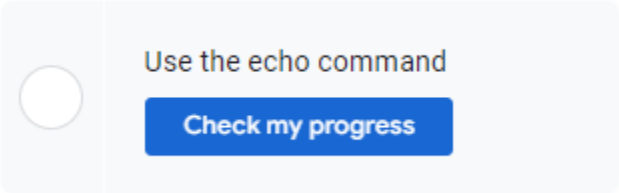
When you click the button to Open Linux Console, the terminal opens in a new browser window:



Use this feature if you want a full-screen view of the terminal. You can close this window at any time. Closing the window does not end your lab, and you can continue working in the terminal in the original tab.

Check progress

You can check your progress by clicking Check my progress at the end of each task.



If you haven't yet completed a task, you'll receive hints on what you must do to complete it.

You can click Check my progress whenever you want to check the completion status of a task or receive a hint.

Using copy/paste commands

The first time you try to use copy or paste keyboard shortcuts (such as CTRL + C), you'll receive a pop-up requesting permission to use your device's clipboard: "googlecoursera.qwiklabs.com wants to see text and images copied to the clipboard." Please click Allow if you would like to be able to use these shortcuts in the Qwiklabs platform. If you choose not to allow Qwiklabs access to your clipboard, you cannot use keyboard shortcuts but you can still complete the lab.

Code block

Certain steps may include a code block. Click the copy button to copy the code provided and then paste it into the terminal.

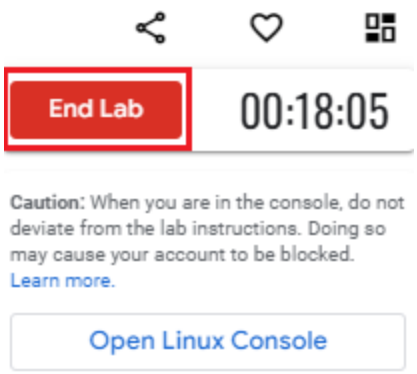
To paste code or other text content that you have copied from the instructions into the terminal, activate the terminal by clicking anywhere inside it. The terminal is active when the cursor in the terminal changes from a static empty outline to a flashing solid block.

Once the terminal is active, use the keyboard shortcut CTRL + V (hold down the CTRL key and press the V key) to insert the copied text into the terminal at the location of the flashing cursor.

Scrolling

In certain situations, you may want to scroll within the terminal window. To do so, use the scroll wheel on your mouse or the touchpad of your computer.

End Lab button



Finally, click End Lab when you've completed the tasks in the lab.

Note: Don't click End Lab until you're finished; you'll lose access to the work you've done throughout the lab.

Tracking progress on Coursera

If you complete a lab but your progress hasn't been tracked on Coursera, you may need to refresh the page for your progress to be registered. Once you complete the lab and refresh the page, the green check mark should appear.

Helpful navigation tips and keyboard shortcuts

The following contains a list of navigation tips and keyboard shortcuts you may find useful when completing your SQL labs. Your cursor must be in the terminal window to use these navigation tips and keyboard shortcuts.

- **`CTRL + C`**: Terminates a command that is currently running; from the instructions portion of Qwiklabs, you can use **`CTRL + C`** to copy, but within the terminal, it will only terminate a command and if one isn't running, it will exit out of the MariaDB shell; if you unintentionally exit, you can reconnect by running the **`sudo mysql organization`** command
- **`CTRL + V`**: Pastes text
- **`CTRL + L`**: Clears the terminal screen; within MariaDB, you must use **`CTRL + L`** and cannot use **`clear`**
- **`\c + Enter`**: Clears the current input
- **`CTRL + A`**: Sets your cursor at the beginning of a command
- **`CTRL + E`**: Sets your cursor at the end of a command
- **`Left arrow`** key: Moves left within a command
- **`Right arrow`** key: Moves right within a command
- **`Up arrow`** key: Provides the last command you entered into the command line; can be entered multiple times to go through multiple commands from the command history
- **`Down arrow`** key: Provides the next command in the command history; must be after using the **`up arrow`** key
- **`Tab`** key: Provides available suggestions for completing your text

Note: If you unintentionally exit the **`organization`** database in the MariaDB shell, you can reconnect by running the **`sudo mysql organization`** command.

Key takeaways

Knowing how to navigate Qwiklabs will be useful as you complete the labs throughout this course. These labs can help you practice what you've learned in an interactive environment.

More on filters with AND, OR, and NOT

Previously, you explored how to add filters containing the `AND`, `OR`, and `NOT` operators to your SQL queries. In this reading, you'll continue to explore how these operators can help you refine your queries.

Logical operators

`AND`, `OR`, and `NOT` allow you to filter your queries to return the specific information that will help you in your work as a security analyst. They are all considered logical operators.

AND

First, `AND` is used to filter on two conditions. `AND` specifies that both conditions must be met simultaneously.

As an example, a cybersecurity concern might affect only those customer accounts that meet both the condition of being handled by a support representative with an ID of 5

and the condition of being located in the USA. To find the names and emails of those specific customers, you should place the two conditions on either side of the `AND` operator in the `WHERE` clause:

```
1
2
3
```

Reset

Running this query returns four rows of information about the customers. You can use this information to contact them about the security concern.

OR

The `OR` operator also connects two conditions, but `OR` specifies that either condition can be met. It returns results where the first condition, the second condition, or both are met.

For example, if you are responsible for finding all customers who are either in the USA or Canada so that you can communicate information about a security update, you can use an `OR` operator to find all the needed records. As the following query demonstrates, you should place the two conditions on either side of the `OR` operator in the `WHERE` clause:

```
1
2
3
```

Reset

The query returns all customers in either the US or Canada.

Note: Even if both conditions are based on the same column, you need to write out both full conditions. For instance, the query in the previous example contains the filter `WHERE country = 'Canada' OR country = 'USA'`.

NOT

Unlike the previous two operators, the `NOT` operator only works on a single condition, and not on multiple ones. The `NOT` operator negates a condition. This means that SQL returns all records that don't match the condition specified in the query.

For example, if a cybersecurity issue doesn't affect customers in the USA but might affect those in other countries, you can return all customers who are not in the USA. This would be more efficient than creating individual conditions for all of the other countries. To use the `NOT` operator for this task, write the following query and place `NOT` directly after `WHERE`:

```
1

2

3
```

Reset

SQL returns every entry where the customers are not from the USA.

Pro tip: Another way of finding values that are not equal to a certain value is by using the `<>` operator or the `!=` operator. For example, `WHERE country <> 'USA'` and `WHERE country != 'USA'` are the same filters as `WHERE NOT country = 'USA'`.

Combining logical operators

Logical operators can be combined in filters. For example, if you know that both the USA and Canada are not affected by a cybersecurity issue, you can combine operators to return customers in all countries besides these two. In the following query, `NOT` is

placed before the first condition, it's joined to a second condition with `AND`, and then `NOT` is also placed before that second condition. You can run it to explore what it returns:

```
1

2

3
```

Reset

Key takeaways

Logical operators allow you to create more specific filters that target the security-related information you need. The `AND` operator requires two conditions to be true simultaneously, the `OR` operator requires either one or both conditions to be true, and the `NOT` operator negates a condition. Logical operators can be combined together to create even more specific queries.

Portfolio Activity Exemplar: Apply filters to SQL queries

Here is a completed exemplar along with an explanation of how the exemplar fulfills the expectations for the activity.

Completed Exemplar

To review the exemplar for this course item, click the following link and select *Use Template*.

Link to exemplar: Apply filters to SQL queries

OR

If you don’t have a Google account, you can download the exemplar directly from the following attachment.

Assessment of Exemplar

Compare the exemplar to your completed activity. Review your work using each of the criteria in the exemplar. What did you do well? Where can you improve? Use your answers to these questions to revise your project as needed and guide you as you continue to progress through the certificate program.

Note: The exemplar represents one possible way to complete the Apply filters to SQL queries portfolio activity. Yours will likely differ in certain ways. What's important is that you understand how to use SQL queries to apply filters.

The exemplar uses details from the given scenario and includes the following:

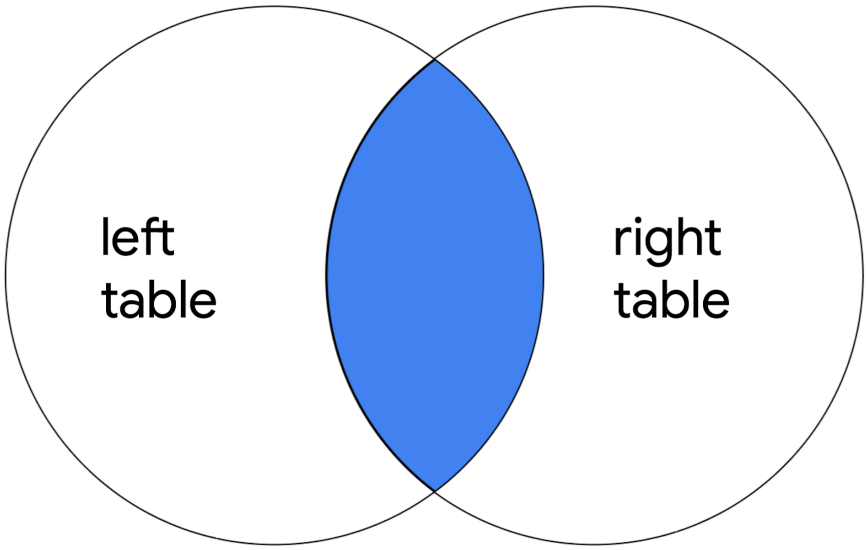
- Screenshots of queries or typed versions of the queries
- Explanations of queries
- A project description at the beginning
- A summary at the end
- Details on using `LIKE` to search for a pattern
- Details on filtering for dates and times
- Details on using `AND` and `OR` to filter on multiple conditions
- Details on using `NOT` in filters

Compare types of joins

Previously, you explored SQL joins and how to use them to join data from multiple tables when these tables share a common column. You also examined how there are different types of joins, and each of them returns different rows from the tables being joined. In this reading, you'll review these concepts and more closely analyze the syntax needed for each type of join.

Inner joins

The first type of join that you might perform is an inner join. `INNER JOIN` returns rows matching on a specified column that exists in more than one table.



It only returns the rows where there is a match, but like other types of joins, it returns all specified columns from all joined tables. For example, if the query joins two tables with `SELECT *`, all columns in both of the tables are returned.

Note: If a column exists in both of the tables, it is returned twice when `SELECT *` is used.

The syntax of an inner join

To write a query using `INNER JOIN`, you can use the following syntax:

```
SELECT *

FROM employees

INNER JOIN machines ON employees.device_id = machines.device_id;
```

You must specify the two tables to join by including the first or left table after `FROM` and the second or right table after `INNER JOIN`.

After the name of the right table, use the `ON` keyword and the `=` operator to indicate the column you are joining the tables on. It's important that you specify both the table and column names in this portion of the join by placing a period (`.`) between the table and the column.

In addition to selecting all columns, you can select only certain columns. For example, if you only want the join to return the `username`, `operating_system` and `device_id` columns, you can write this query:

```
SELECT username, operating_system, employees.device_id

FROM  employees

INNER JOIN machines ON employees.device_id = machines.device_id;
```

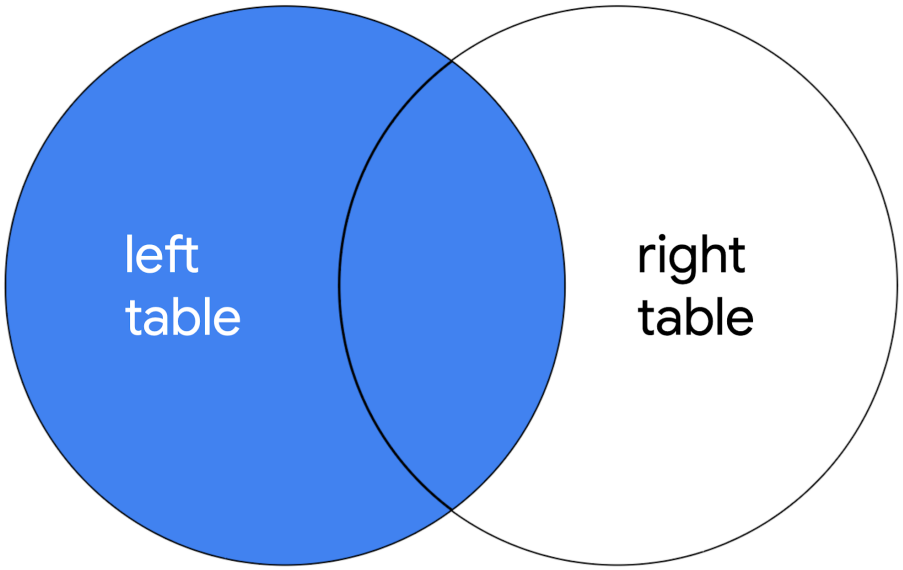
Note: In the example query, `username` and `operating_system` only appear in one of the two tables, so they are written with just the column name. On the other hand, because `device_id` appears in both tables, it's necessary to indicate which one to return by specifying both the table and column name (`employees.device_id`).

Outer joins

Outer joins expand what is returned from a join. Each type of outer join returns all rows from either one table or both tables.

Left joins

When joining two tables, `LEFT JOIN` returns all the records of the first table, but only returns rows of the second table that match on a specified column.



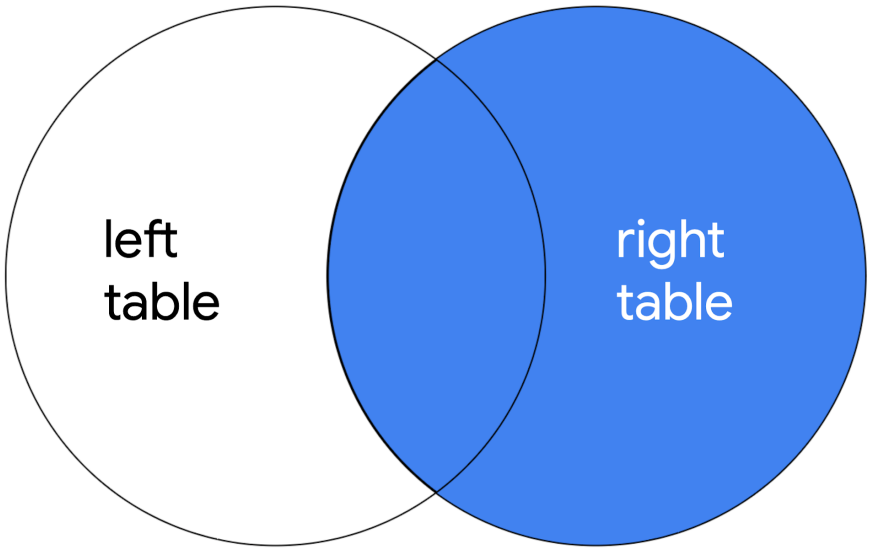
The syntax for using `LEFT JOIN` is demonstrated in the following query:

```
SELECT *

FROM employees

LEFT JOIN machines ON employees.device_id = machines.device_id;
```

As with all joins, you should specify the first or left table as the table that comes after `FROM` and the second or right table as the table that comes after `LEFT JOIN`. In the example query, because `employees` is the left table, all of its records are returned. Only records that match on the `device_id` column are returned from the right table, `machines`.

Right joins

When joining two tables, `RIGHT JOIN` returns all of the records of the second table, but only returns rows from the first table that match on a specified column.



The following query demonstrates the syntax for `RIGHT JOIN`:

```
SELECT *

FROM employees

RIGHT JOIN machines ON employees.device_id = machines.device_id;
```

`RIGHT JOIN` has the same syntax as `LEFT JOIN`, with the only difference being the keyword `RIGHT JOIN` instructs SQL to produce different output. The query returns all records from `machines`, which is the second or right table. Only matching records are returned from `employees`, which is the first or left table.

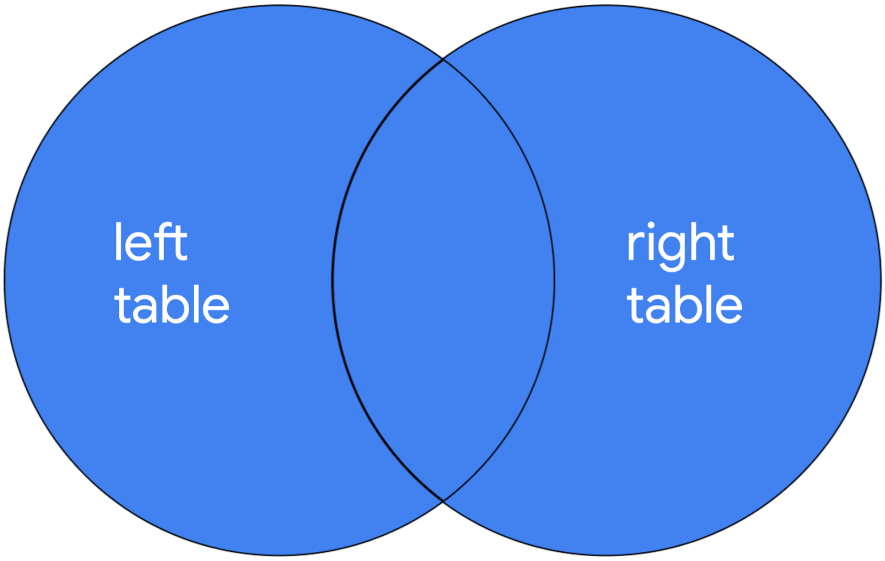
Note: You can use `LEFT JOIN` and `RIGHT JOIN` and return the exact same results if you use the tables in reverse order. The following `RIGHT JOIN` query returns the exact same result as the `LEFT JOIN` query demonstrated in the previous section:

```
SELECT *

FROM machines

RIGHT JOIN employees ON employees.device_id = machines.device_id;
```

All that you have to do is switch the order of the tables that appear before and after the keyword used for the join, and you will have swapped the left and right tables.

Full outer joins

`FULL OUTER JOIN` returns all records from both tables. You can think of it as a way of completely merging two tables.



You can review the syntax for using `FULL OUTER JOIN` in the following query:

```
SELECT *

FROM employees

FULL OUTER JOIN machines ON employees.device_id =
machines.device_id;
```

The results of a `FULL OUTER JOIN` query include all records from both tables. Similar to `INNER JOIN`, the order of tables does not change the results of the query.

Key takeaways

When working in SQL, there are multiple ways to join tables. All joins return the records that match on a specified column. `INNER JOIN` will return only these records. Outer joins also return all other records from one or both of the tables. `LEFT JOIN` returns all records from the first or left table, `RIGHT JOIN` returns all records from the second or right table, and `FULL OUTER JOIN` returns all records from both tables.

Continuous learning in SQL

You've explored a lot about SQL, including applying filters to SQL queries and joining multiple tables together in a query. There's still more that you can do with SQL. This reading will explore an example of something new you can add to your SQL toolbox: aggregate functions. You'll then focus on how you can continue learning about this and other SQL topics on your own.

Aggregate functions

In SQL, aggregate functions are functions that perform a calculation over multiple data points and return the result of the calculation. The actual data is not returned.

There are various aggregate functions that perform different calculations:

- `COUNT` returns a single number that represents the number of rows returned from your query.
- `AVG` returns a single number that represents the average of the numerical data in a column.
- `SUM` returns a single number that represents the sum of the numerical data in a column.

Aggregate function syntax

To use an aggregate function, place the keyword for it after the `SELECT` keyword, and then in parentheses, indicate the column you want to perform the calculation on.

For example, when working with the `customers` table, you can use aggregate functions to summarize important information about the table. If you want to find out how many customers there are in total, you can use the `COUNT` function on any column, and SQL will return the total number of records, excluding `NULL` values. You can run this query and explore its output:

1

2

Reset

The result is a table with one column titled `COUNT(firstname)` and one row that indicates the count.

If you want to find the number of customers from a specific country, you can add a filter to your query:

```
1

2

3
```

Reset

With this filter, the count is lower because it only includes the records where the `country` column contains a value of `'USA'`.

There are a lot of other aggregate functions in SQL. The syntax of placing them after `SELECT` is exactly the same as the `COUNT` function.

Continuing to learn SQL

SQL is a widely used querying language, with many more keywords and applications. You can continue to learn more about aggregate functions and other aspects of using SQL on your own.

Most importantly, approach new tasks with curiosity and a willingness to find new ways to apply SQL to your work as a security analyst. Identify the data results that you need and try to use SQL to obtain these results.

Fortunately, SQL is one of the most important tools for working with databases and analyzing data, so you'll find a lot of support in trying to learn SQL online. First, try searching for the concepts you've already learned and practiced to find resources that have accurate easy-to-follow explanations. When you identify these resources, you can use them to extend your knowledge.

Continuing your practical experience with SQL is also important. You can also search for new databases that allow you to perform SQL queries using what you've learned.

Key takeaways

Aggregate functions like `COUNT`, `SUM`, and `AVG` allow you to work with SQL in new ways. There are many other additional aspects of SQL that could be useful to you as an analyst. By continuing to explore SQL on your own, you can expand the ways you can apply SQL in a cybersecurity context.

Reference guide: SQL

The SQL reference guide contains keywords for SQL queries. Security analysts can use these keywords to query databases and find data to support security-related decisions. The reference guide is divided into four different categories of SQL keywords for security-related tasks:

- Query a database
- Apply filters to SQL queries
- Join tables
- Perform calculations

Within each category, commands are organized alphabetically.

Access and save the guide

You can save a copy of this guide for future reference. You can use it as a resource for additional practice or in your future professional projects.

To access a downloadable version of this course item, click the following link and select *Use Template*.

Reference guide: SQL

OR

If you don’t have a Google account, you can download the item directly from the following attachment.

https://www.coursera.org/api/rest/v1/asset/download/pdf/UkgRanrLR_Ki2e1zVCKYDA?pageStart=&pageEnd=

Glossary terms from module 4

Terms and definitions from Course 4, Module 4

Database: An organized collection of information or data

Date and time data: Data representing a date and/or time

Exclusive operator: An operator that does not include the value of comparison

Filtering: Selecting data that match a certain condition

Foreign key: A column in a table that is a primary key in another table

Inclusive operator: An operator that includes the value of comparison

Log: A record of events that occur within an organization's systems

Numeric data: Data consisting of numbers

Operator: A symbol or keyword that represents an operation

Primary key: A column where every row has a unique entry

Query: A request for data from a database table or a combination of tables

Relational database: A structured database containing tables that are related to each other

String data: Data consisting of an ordered sequence of characters

SQL (Structured Query Language): A programming language used to create, interact with, and request information from a database

Syntax: The rules that determine what is correctly structured in a computing language

Wildcard: A special character that can be substituted with any other character

Course 4 glossary

We've covered a lot of terms—some of which you may have already known, and some of which are new. To make it easy to remember what a word means, we created this glossary of terms and definitions.

To use the glossary for this course item, click the link below and select "Use Template."

Link to glossary: Course 4 Glossary

OR

If you don't have a Google account, you can download the glossary directly from the attachment below

https://d3c33hcgiwev3.cloudfront.net/R81C_91rT--Dh0uAm7IswA_2fd46480e31e45f1b8ae9a0b117addf1_Course-4-glossary.docx?Expires=1762560000&Signature=g4gv~4Z-~DC32zCQhANOtFIMngjAHRUtYmFcwU3jDtpI-kLsu776pUp1~L21AYxUIvvw2FOMfkJW~JV4DTWBthjIDfPvlExKrIRsIGh7Q6TrXNXzEGhGvMl5KepyU~XffpDFrH6hpCsIFLJwp~lW4uTLnX19XbCRVtkD5srl4DY_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A