4.1 Linux basics

4.1.1 What is Linux

Linux is an operating system that was created in 1991. Linux is open source, fast, reliable, and small. It requires very little hardware resources to run and is highly customizable. Unlike other operating systems such as Windows and Mac OS X, Linux was created, and is currently maintained by, a community of programmers. Linux is part of several platforms and can be found on devices anywhere from "wristwatches to supercomputers".

Another important aspect of Linux is that it is designed to be connected to the network, which makes it much simpler to write and use network-based applications. Because Linux is open source, any person or company can get the kernel's source code, inspect it, modify it, and re-compile it at will. They are also allowed to redistribute the program with or without charges.

A Linux distribution is the term used to describe packages created by different organizations. Linux distributions (or distros) include the Linux kernel with customized tools and software packages. While some of these organizations may charge for their Linux distribution support (geared towards Linux-based businesses), the majority of them also offer their distribution for free without support. Debian, Red Hat, Ubuntu, CentOS, and SUSE are just a few examples of Linux distributions.

4.1.2 The Value of Linux

Linux is often the operating system of choice in the Security Operations Center (SOC). These are some of the reasons to choose Linux:

- Linux is open source - Any person can acquire Linux at no charge and modify it to fit specific needs. This flexibility allows analysts and administrators to tailor-build an operating system specifically for security analysis.
- The Linux CLI is very powerful - While a GUI makes many tasks easier to perform, it adds complexity and requires more computer resources to run. The Linux Command Line Interface (CLI) is extremely powerful and enables analysts to perform tasks not only directly on a terminal, but also remotely.
- The user has more control over the OS - The administrator user in Linux, known as the root user, or superuser, has absolute power over the computer. Unlike other operating systems, the root user can modify any aspect of the computer with a few keystrokes. This ability is especially valuable when working with low level functions such as the network stack. It allows the root user to have precise control over the way network packets are handled by the operating system.
- It allows for better network communication control - Control is an inherent part of Linux. Because the OS can be adjusted in practically every aspect, it is a great platform for creating network applications. This is the same reason that many great network-based software tools are available for Linux only.

4.1.3 Linux in SOC

The flexibility provided by Linux is a great feature for the SOC. The entire operating system can be tailored to become the perfect security analysis platform. For example, administrators can add only the necessary packages to the OS, making it lean and efficient. Specific software tools can be installed and configured to work in conjunction, allowing administrators to build a customized computer that fits perfectly in the workflow of a SOC. Sguil, which is the cybersecurity analyst console in a special version of Linux called Security Onion.

SOC Tool	Description
Network packet capture software	• A crucial tool for a SOC analyst as it makes it possible to observe and understand every detail of a network transaction. • Wireshark is a popular packet capture tool.
Malware analysis tools	These tools allow analysts to safely run and observe malware execution without the risk of compromising the underlying system.
Intrusion detection systems (IDSs)	• These tools are used for real-time traffic monitoring and inspection. • If any aspect of the currently flowing traffic matches any of the established rules, a pre-defined action is taken.
Firewalls	This software is used to specify, based on pre-defined rules, whether traffic is allowed to enter or leave a network or device.
Log managers	• Log files are used to record events. • Because a network can generate a very large number of log entries, log manager software is employed to facilitate log monitoring.
Security information and event management (SIEM)	SIEMs provide real-time analysis of alerts and log entries generated by network appliances such as IDSs and firewalls.
Ticketing systems	Task ticket assignment, editing, and recording is done through a ticket management system. Security alerts are often assigned to analysts through a ticketing system.

4.1.4 Linux tools

In addition to SOC-specific tools, Linux computers that are used in the SOC often contain penetration testing tools. Also known as PenTesting, a penetration test is the process of looking for vulnerabilities in a network or computer by attacking it. Packet generators, port scanners, and proof-of-concept exploits are examples of PenTesting tools.

Kali Linux is a Linux distribution groups many penetration tools together in a single Linux distribution. Kali contains a great selection of tools.

4.2 Working in Linux shell

4.2.1 Linux Shell

In Linux, the user communicates with the OS by using the CLI or the GUI. Linux often starts in the GUI by default. This hides the CLI from the user. One way to access the CLI from the GUI is through a terminal emulator application. These applications provide user access to the CLI and are often named as some variation of the word "terminal". In Linux, popular terminal emulators are Terminator, eterm, xterm, konsole, and gnome-terminal.

4.2.2 Basic Commands

Linux commands are programs created to perform a specific task. Use the **man** command (short for manual) to obtain documentation about commands. As an example, **man ls** provides documentation about the **ls** command from the user manual.

Command	Description
mv	Moves or renames files and directories
chmod	Modifies file permissions
chown	Changes the ownership of a file
dd	Copies data from an input to an output
pwd	Displays the name of the current directory
ps	Lists the processes that are currently running in the system
su	Simulates a login as another user or to become a superuser
sudo	Runs a command as a super user, by default, or another named user
grep	Used to search for specific strings of characters within a file or other command outputs. To search through the output of a previous command, **grep** must be piped at the end of the previous command.
ifconfig	Used to display or configure network card related information. If issued without parameters, **ifconfig** will display the current network card(s) configuration. Note: While still widely in use, this command is deprecated. Use **ip address** instead.
apt-get	Used to install, configure and remove packages on Debian and its derivatives. Note: **apt-get** is a user-friendly command line front-end for **dpkg**, Debian's package manager. The combo **dpkg** and **apt-get** is the default package manager system in all Debian Linux derivatives, including Raspbian.

iwconfig	Used to display or configure wireless network card related information. Similar to **ifconfig**, **iwconfig** will display wireless information when issued without parameters.
shutdown	Shuts down the system, **shutdown** can be instructed to perform a number of shut down related tasks, including restart, halt, put to sleep or kick out all currently connected users.
passwd	Used to change the password. If no parameters are provided, **passwd** changes the password for the current user.
cat	Used to list the contents of a file and expects the file name as the parameter. The **cat** command is usually used on text files.
man	Used to display the documentation for a specific command.

4.2.3 File and Directory Commands

Command	Description
ls	Displays the files inside a directory
cd	Changes the current directory
mkdir	Creates a directory under the current directory
cp	Copies files from source to destination
mv	Moves files to a different directory
rm	Removes files
grep	Searches for specific strings of characters within a file or other commands outputs
cat	Lists the contents of a file and expects the file name as the parameter

4.2.4 Working wit Text

Linux has many different text editors, with various features and functions. Some text editors include graphical interfaces while others are command-line only tools. Each text editor includes a feature set designed to support a specific type of task. Some text editors focus on the programmer and include features such as syntax highlighting, brackets and parenthesis check, and other programming-focused features.

While graphical text editors are convenient and easy to use, command line-based text editors are very important for Linux users. The main benefit of command-line-based text editors is that they allow for text file editing from a remote computer.

Consider the following scenario: a user must perform administrative tasks on a Linux computer but is not sitting in front of that computer. Using SSH, the user starts a remote shell to the remote computer. Under the text-based remote shell, the graphical interface is not available, which makes it impossible to rely on tools such as graphical text editors. In this type of situation, text-based programs are crucial.

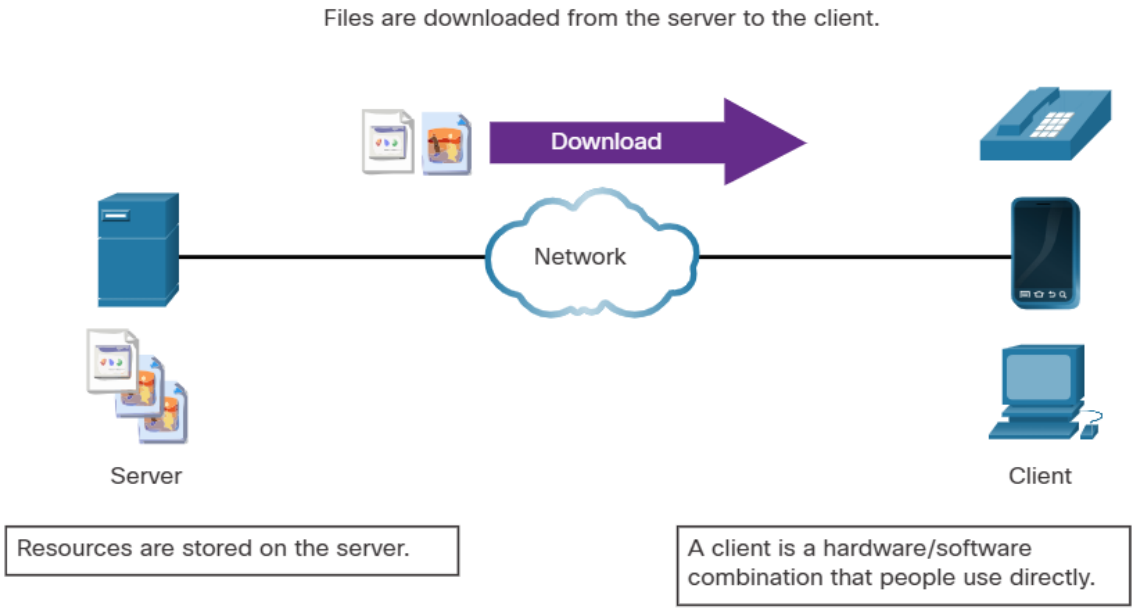
4.2.5 Importance of Text file in Linux

In Linux, everything is treated as a file. This includes the memory, the disks, the monitor, and the directories. For example, from the operating system standpoint, showing information on the display means to write to the file that represents the display device. It should be no surprise that the computer itself is configured through files. Known as configuration files, they are usually text files used to store adjustments and settings for specific applications or services. Practically everything in Linux relies on configuration files to work. Some services have not one, but several configuration files.

4.3 Linux Server and clients

4.3.1 Introduction to client-server communications

Servers are computers with software installed that enables them to provide services to clients across the network. There are many types of services. Some provide external resources such as files, email messages, or web pages to clients upon request. Other services run maintenance tasks such as log management, memory management, disk scanning, and more. Each service requires separate server software. For example, the server in the figure uses file server software to provide clients with the ability to retrieve and submit files.



4.3.2 Servers, Service and Ports

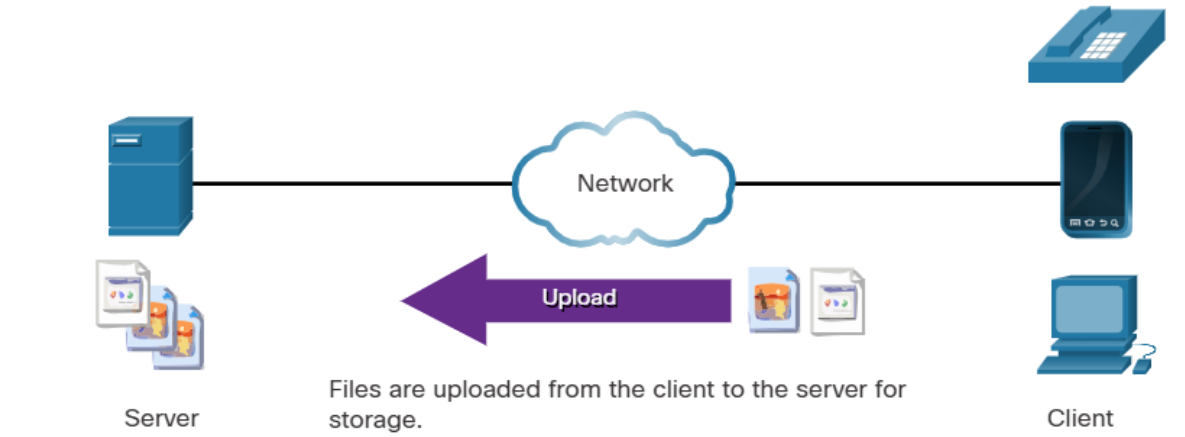
In order that a computer can be the server for multiple services, ports are used. A port is a reserved network resource used by a service. A server is said to be “listening” on a port when it has associated itself to that port.

While the administrator can decide which port to use with any given service, many clients are configured to use a specific port by default. It is common practice to leave the service running in its default port. The table lists a few commonly used ports and their services. These are also called “well-known ports”.

Port	Description
20/21	File Transfer Protocol (FTP)
22	Secure Shell (SSH)
23	Telnet remote login service
25	Simple Mail Transfer Protocol (SMTP)
53	Domain Name System (DNS)
67/68	Dynamic Host Configuration Protocol (DHCP)
69	Trivial File Transfer Protocol (TFTP)
80	Hypertext Transfer Protocol (HTTP)
110	Post Office Protocol version 3 (POP3)
123	Network Time Protocol (NTP)
143	Internet Message Access Protocol (IMAP)
161/162	Simple Network Management Protocol (SNMP)
443	HTTP Secure (HTTPS)

4.3.3 Clients

Clients are programs or applications designed to communicate with a specific type of server. Also known as client applications, clients use a well-defined protocol to communicate with the server. Web browsers are web clients that are used to communicate with web servers through the Hyper Text Transfer Protocol (HTTP) on port 80. The File Transfer Protocol (FTP) client is software used to communicate with an FTP server. The figure shows a client uploading files to a server.



4.4 Basic Server Administrator

4.4.1 Service Configuration files

In Linux, services are managed using configuration files. Common options in configuration files are port number, location of the hosted resources, and client authorization details. When the service starts, it looks for its configuration files, loads them into memory, and adjusts itself according to the settings in the files. Configuration file modifications often require restarting the service before the changes take effect.

Because services often require superuser privileges to run, service configuration files often require superuser privileges to edit.

The command output shows a portion of the configuration file for Nginx, which is a lightweight web server for Linux.

```
[analyst@secOps ~]$ cat /etc/nginx/nginx.conf
#user html;
worker_processes  1;
#error_log  logs/error.log;
#error_log  logs/error.log  notice;
#error_log  logs/error.log  info;
#pid        logs/nginx.pid;
events {
    worker_connections  1024;
}
http {
    include       mime.types;
    default_type  application/octet-stream;
```

```
    #log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
    #                  '$status $body_bytes_sent "$http_referer" '
    #                  '"$http_user_agent" "$http_x_forwarded_for"';
    #access_log  logs/access.log  main;
```

The next command output shows the configuration file for the network time protocol (NTP).

```
[analyst@secOps ~]$ cat /etc/ntp.conf
# Please consider joining the pool:
#
# http://www.pool.ntp.org/join.html
#
# For additional information see:
# - https://wiki.archlinux.org/index.php/Network_Time_Protocol_daemon
# - http://support.ntp.org/bin/view/Support/GettingStarted
# - the ntp.conf man page
# Associate to Arch's NTP pool
server 0.arch.pool.ntp.org
server 1.arch.pool.ntp.org
server 2.arch.pool.ntp.org
server 3.arch.pool.ntp.org
# By default, the server allows:
# - all queries from the local host
# - only time queries from remote hosts, protected by rate limiting and kod
restrict default kod limited nomodify nopeer noquery notrap
restrict 127.0.0.1
restrict ::1
# Location of drift file
[analyst@secOps ~]$
```

The last command output shows the configuration file for Snort, a Linux-based intrusion detection system (IDS).

```
[analyst@secOps ~]$ cat /etc/snort/snort.conf
#--------------------------------------------------
#   VRT Rule Packages Snort.conf
#
#   For more information visit us at:
# http://www.snort.org                Snort Website
# http://vrt-blog.snort.org/ Sourcefire VRT Blog
#
# Mailing list Contact:   snort-sigs@lists.sourceforge.net
# False Positive reports: fp@sourcefire.com
# Snort bugs:             bugs@snort.org
#
# Compatible with Snort Versions:
# VERSIONS : 2.9.9.0
#
```

```
# Snort build options:
# OPTIONS : --enable-gre --enable-mpls --enable-targetbased --enable-ppm --enable-perfprofiling --
enable-zlib --enable-active-response --enable-normalizer --enable-reload --enable-react --enable-
flexresp3
<output omitted>
##################################################
# Step #1: Set the network variables.  For more information, see README.variables
##################################################
# Setup the network addresses you are protecting
###ipvar HOME_NET any
###ipvar HOME_NET [192.168.0.0/24,192.168.1.0/24]
ipvar HOME_NET [209.165.200.224/27]
# Set up the external network addresses. Leave as "any" in most situations
ipvar EXTERNAL_NET any
```

There is no rule for a configuration file format; it is the choice of the service's developer. However, the **option = value** format is often used. For example, in the last command output, the variable **ipvar** is configured with several options. The first option, HOME_NET, has the value 209.165.200.224/27. The hash character (**#**) is used to indicate comments.

4.4.2 Hardening Devices

Device hardening involves implementing proven methods of securing the device and protecting its administrative access. Some of these methods involve maintaining passwords, configuring enhanced remote login features, and implementing secure login with SSH. Defining administrative roles in terms of access is another important aspect of securing infrastructure devices because not all information technology personnel should have the same level of access to the infrastructure devices.

Depending on the Linux distribution, many services are enabled by default. Some of these features are enabled for historical reasons but are no longer required. Stopping such services and ensuring they do not automatically start at boot time is another device hardening technique.

OS updates are also extremely important to maintaining a hardened device. New vulnerabilities are discovered every day. OS developers create and issue fixes and patches regularly. An up-to-date computer is less likely to be compromised.

The following are basic best practices for device hardening.

- Ensure physical security
- Minimize installed packages
- Disable unused services
- Use SSH and disable the root account login over SSH
- Keep the system updated
- Disable USB auto-detection
- Enforce strong passwords
- Force periodic password changes
- Keep users from re-using old passwords

Many other steps exist and are often service or application-dependent.

4.4.3 Monitoring Services Logs

Log files are the records that a computer stores to keep track of important events. Kernel, services, and application events are all recorded in log files. It is very important for an administrator to periodically review the logs of a computer to keep it healthy. By monitoring Linux log files, an administrator gains a clear picture of the computer's performance, security status, and any underlying issues. Log file analysis allows an administrator to guard against upcoming issues before they occur.

In Linux, log files can be categorized as:

- Application logs
- Event logs
- Service logs
- System logs

Some logs contain information about daemons that are running in the Linux system. A daemon is a background process that runs without the need for user interaction. For example, the System Security Services Daemon (SSSD) manages remote access and authentication for single sign-on capabilities.

The table lists a few popular Linux log files and their functions

Linux Log File	Description
/var/log/messages	• This directory contains generic computer activity logs. • It is mainly used to store informational and non-critical system messages. • In Debian-based computers, /var/log/syslog directory serves the same purpose.
/var/log/auth.log	• This file stores all authentication-related events in Debian and Ubuntu computers. • Anything involving the user authorization mechanism can be found in this file.
/var/log/secure	• This directory is used by RedHat and CentOS computers instead of /var/log/auth.log. • It also tracks sudo logins, SSH logins, and other errors logged by SSSD.
/var/log/boot.log	• This file stores boot-related information and messages logged during the computer startup process.
/var/log/dmesg	• This directory contains kernel ring buffer messages. • Information related to hardware devices and their drivers is recorded here. • It is very important because, due to their low-level nature, logging systems such as syslog are not running when these events take place and therefore are often unavailable to the administrator in real-time.
/var/log/kern.log	• This file contains information logged by the kernel.
/var/log/cron	• Cron is a service used to schedule automated tasks in Linux and this directory stores its events. • Whenever a scheduled task (also called a cron job) runs, all its relevant information including execution status and error messages are stored here.
/var/log/mysqld.log or **/var/log/mysql.log**	• This is the MySQL log file. • All debug, failure and success messages related to the mysqld process and mysqld_safe daemon are logged here. • RedHat, CentOS and Fedora Linux distributions store MySQL logs under /var/log/mysqld.log, while Debian and Ubuntu maintain the log in /var/log/mysql.log file.

4.5 The Linux File System

4.5.1 file System types in linux

Linux File System	Description
ext2 (second extended file system)	• ext2 was the default file system in several major Linux distributions until supplanted by ext3. • Almost fully compatible with ext2, ext3 also supports journaling (see below). • ext2 is still the file system of choice for flash-based storage media because its lack of a journal increases performance and minimizes the number of writes. • Because flash memory devices have a limited number of write operations, minimizing write operations increases the device's lifetime. • However, contemporary Linux kernels also support ext4, an even more modern file system, with better performance and which can also operate in a journal-less mode.
ext3 (third extended file system)	• ext3 is a journaled file system designed to improve the existing ext2 file system. • A journal, the main feature added to ext3, is a technique used to minimize the risk of file system corruption in the event of sudden power loss. • The file systems keeps a log (or journal) of all the file system changes about to be made. • If the computer crashes before the change is complete, the journal can be used to restore or correct any eventual issues created by the crash. • The maximum file size in ext3 file systems is 32 TB.
ext4 (fourth extended file system)	• Designed as a successor of ext3, ext4 was created based on a series of extensions to ext3. • While the extensions improve the performance of ext3 and increase supported file sizes, Linux kernel developers were concerned about stability issues and were opposed to adding the extensions to the stable ext3. • The ext3 project was split in two; one kept as ext3 and its normal development and the other, named ext4, incorporated the mentioned extensions.

NFS (Network File System)	• NFS is a network-based file system, allowing file access over the network. • From the user standpoint, there is no difference between accessing a file stored locally or on another computer on the network. • NFS is an open standard which allows anyone to implement it.
CDFS (Compact Disc File System)	CDFS was created specifically for optical disk media.
Swap File System	• The swap file system is used by Linux when it runs out of RAM. • Technically, it is a swap partition that does not have a specific file system, but it is relevant to the file system discussion. • When this happens, the kernel moves inactive RAM content to the swap partition on the disk. • While swap partitions (also known as swap space) can be useful to Linux computers with a limited amount of memory, they should not be considered as a primary solution. • Swap partition is stored on disk which has much lower access speeds than RAM.
HFS Plus or HFS+ (Hierarchical File System Plus)	• A file system used by Apple in its Macintosh computers. • The Linux kernel includes a module for mounting HFS+ for read-write operations.
APFS (Apple File System)	An updated file system that is used by Apple devices. It provides strong encryption and is optimized for flash and solid-state drives.
Master Boot Record (MBR)	• Located in the first sector of a partitioned computer, the MBR stores all the information about the way in which the file system is organized. • The MBR quickly hands over control to a loading function, which loads the OS.

4.5.2 Linux roles and File Permissions

In Linux, most system entities are treated as files. In order to organize the system and enforce boundaries within the computer, Linux uses file permissions. File permissions are built into the file system structure and provide a mechanism to define permissions on every file. Every file in Linux carries its file permissions, which define the actions that the owner, the group, and others can perform with the file. The possible permission rights are Read, Write and Execute. The **ls** command with the **-l** parameter lists additional information about the file.

Consider the output of the **ls -l** command in the command output.

```
[analyst@secOps ~]$ ls -l space.txt
-rwxrw-r-- 1 analyst staff 253 May 20 12:49 space.txt
(1)(2)(3)(4)(5)(6)(7)
[analyst@secOps ~]$
```

The output provides a lot of information about the file space.txt.

The first field of the output displays the permissions that are associated with **space.txt** (**-rwxrw-r--**). File permissions are always displayed in the User, Group, and Other order.

The file **space.txt** in has the following permissions:

- The dash (-) means that this is a file. For directories, the first dash would be a “d”.
- The first set of characters is for user permission (**rwx**). The user, **analyst**, who owns the file can **R**ead, **W**rite and e**X**ecute the file.
- The second set of characters is for group permissions (**rw-**). The group, **staff**, who owns the file can **R**ead and **W**rite to the file.
- The third set of characters is for any other user or group permissions (**r--**). Any other user or group on the computer can only **R**ead the file.

The second field defines the number of hard links to the file (the number **1** after the permissions). A hard link creates another file with a different name linked to the same place in the file system (called an inode). This is in contrast to a symbolic link, which is discussed on the next page.

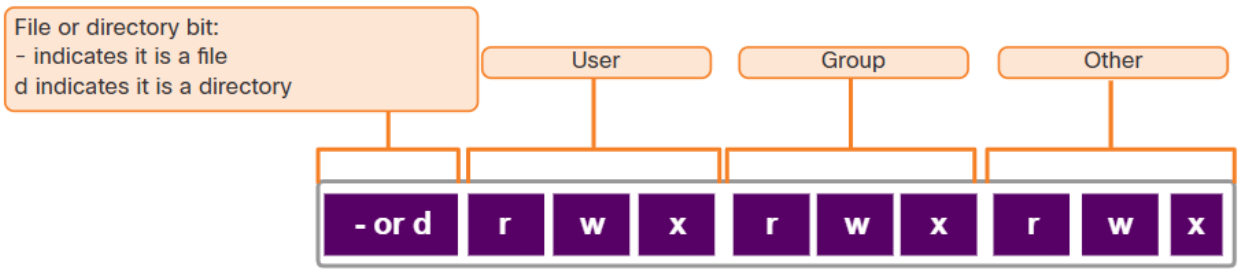
The third and fourth field display the user (**analyst**) and group (**staff**) who own the file, respectively.

The fifth field displays the file size in bytes. The **space.txt** file has 253 bytes.

The sixth field displays the date and time of the last modification.

The seventh field displays the file name.

The figure shows a breakdown of file permissions in Linux.



Use octal values to define permissions.

Binary	Octal	Permission	Description
000	0	---	No access
001	1	--x	Execute only
010	2	-w-	Write only
011	3	-wx	Write and Execute
100	4	r--	Read only
101	5	r-x	Read and Execute
110	6	rw-	Read and Write
111	7	rwx	Read, Write and Execute

4.5.3 Hard links and Symbolic links

A hard link is another file that points to the same location as the original file. Use the command **ln** to create a hard link. The first argument is the existing file and the second argument is the new file.

As shown in the command output, the file **space.txt** is linked to **space.hard.txt** and the link field now shows 2.

```
[analyst@secOps ~]$ ln space.txt space.hard.txt
[analyst@secOps ~]$
[analyst@secOps ~]$ ls -l space*
-rw-r--r-- 2 analyst analyst 239 May  7 18:18 space.hard.txt
-rw-r--r-- 2 analyst analyst 239 May  7 18:18 space.txt
[analyst@secOps ~]$
[analyst@secOps ~]$ echo "Testing hard link" >> space.txt
[analyst@secOps ~]$
[analyst@secOps ~]$ ls -l space*
-rw-r--r-- 2 analyst analyst 257 May  7 18:19 space.hard.txt
-rw-r--r-- 2 analyst analyst 257 May  7 18:19 space.txt
[analyst@secOps ~]$
[analyst@secOps ~]$ rm space.hard.txt
[analyst@secOps ~]$
[analyst@secOps ~]$ more space.txt
Space... The final frontier...
These are the voyages of the Starship Enterprise. Its continuing mission:
- To explore strange new worlds...
- To seek out new life; new civilizations...
- To boldly go where no one has gone before!
```

```
Testing hard link
[analyst@secOps ~]$
```

Both files point to the same location in the file system. If you change one file, the other is changed, as well. The **echo** command is used to add some text to **space.txt**. Notice that the file size for both **space.txt** and **space.hard.txt** increased to 257 bytes. If you delete the space.hard.txt with the **rm** command (remove), the **space.txt** file still exists, as verified with the **more space.txt** command.

A symbolic link, also called a symlink or soft link, is similar to a hard link in that applying changes to the symbolic link will also change the original file. As shown in the command output below, use the **ln** command option **-s** to create a symbolic link.

```
[analyst@secOps ~]$ echo "Hello World!" > test.txt
[analyst@secOps ~]$
[analyst@secOps ~]$ ln -s test.txt mytest.txt
[analyst@secOps ~]$
[analyst@secOps ~]$ echo "It's a lovely day!" >> mytest.txt
[analyst@secOps ~]$
[analyst@secOps ~]$ more test.txt
Hello World!
It's a lovely day!
[analyst@secOps ~]$
[analyst@secOps ~]$ rm test.txt
[analyst@secOps ~]$
[analyst@secOps ~]$ more mytest.txt
more: stat of mytest.txt failed: No such file or directory
[analyst@secOps ~]$
[analyst@secOps ~]$ ls -l mytest.txt
lrwxrwxrwx 1 analyst analyst 8 May  7 20:17 mytest.txt -> test.txt
[analyst@secOps ~]$
```

Notice that adding a line of text to **test.txt** also adds the line to **mytest.txt**. However, unlike a hard link, deleting the original **text.txt** file means that **mytext.txt** is now linked to a file that no longer exists, as shown with the **more mytest.txt** and **ls -l mytest.txt** commands.

Although symbolic links have a single point of failure (the underlying file), symbolic links have several benefits over hard links:

- Locating hard links is more difficult. Symbolic links show the location of the original file in the **ls -l** command, as shown in the last line of output in the previous command output (**mytest.txt -> test.txt**).
- Hard links are limited to the file system in which they are created. Symbolic links can link to a file in another file system.
- Hard links cannot link to a directory because the system itself uses hard links to define the hierarchy of the directory structure. However, symbolic links can link to directories.

4.6 Working with Linux GUI

4.6.1 X windows system

The graphical interface present in most Linux computers is based on the X Window System. Also known as X or X11, X Window is a windowing system designed to provide the basic framework for a GUI. X includes functions for drawing and moving windows on the display device and interacting with a mouse and keyboard.

X works as a server which allows a remote user to use the network to connect, start a graphical application, and have the graphical window open on the remote terminal. While the application itself runs on the server, the graphical aspect of it is sent by X over the network and displayed on the remote computer.

4.6.2 the Linux GUI

Although an operating system does not require a GUI to function, GUIs are considered more user-friendly than the CLI. The Linux GUI as a whole can be easily replaced by the user. As a result of the large number of Linux distributions, this chapter focuses on Ubuntu when covering Linux because it is a very popular and user-friendly distribution.

UI Component	Description
Apps Menu	• The Apps Menu shows icons for the apps that are installed on the system. • A right-click menu provides shortcuts that allow starting or configuring the apps. • The system search box is available from Activities View.
Ubuntu Dock	• This is a dock on the left side of the screen that serves as an application launcher and switcher for app favorites. • Click to launch an application and when the application is running, click again to switch between running applications. • If more than one instance of an application is running, Launcher will display all instances. • Right-click any application that is hosted on the launcher to see details about that the application.
Top Bar	• This multipurpose menu bar contains a menu for the application that currently has the focus. • It displays the current time and indicates whether there are new system messages. • It also provides access to the Activity desktop view and the system Status Menu.
Calendar and System Message Tray	• Click the day and time to see the full appointment calendar and any current system messages. • Access the appointment calendar from here to create new appointments.
Activities	• Switch to application view to switch to or close running applications. • A powerful search tool is available here that will find apps, files, and values within files. • Allows switching between workspaces.

Status Menu	• Allows configuration of the network adaptor and other running devices. • The current user can logoff or change their settings. • System configuration changes can be made here. • The workstation can be locked or shutdown from here.

4.7 Working on Linux Host

4.7.1 Installing and Running Applications of linux host

Many end-user applications are complex programs written in compiled languages. To aid in the installation process, Linux often includes programs called package managers. A package is the term used to refer to a program and all its supporting files. By using a package manager to install a package, all the necessary files are placed in the correct file system location.

Package managers vary depending on Linux distributions. For example, **pacman** is used by Arch Linux while **dpkg** (Debian package) and **apt** (Advanced Packaging Tool) are used in Debian and Ubuntu Linux distributions.

4.7.2 Keep the system up to date

Also known as patches, OS updates are released periodically by OS companies to address any known vulnerabilities in their operating systems. While companies have update schedules, the release of unscheduled OS updates can happen when a major vulnerability is found in the OS code. Modern operating systems will alert the user when updates are available for download and installation, but the user can check for updates at any time.

Task	Arch	Debian / Ubuntu
Install a package by name	**pacman -S**	**apt install**
Remove a package by name	**pacman -Rs**	**apt remove**
Update a local package	**pacman -Syy**	**apt-get update**
Upgrade all currently installed packages	**pacman -Syu**	**apt-get upgrade**

4.7.3 Process and Forks

A process is a running instance of a computer program. Multitasking operating systems can execute many processes at the same time.

Forking is a method that the kernel uses to allow a process to create a copy of itself. Processes need a way to create new processes in multitasking operating systems. The fork operation is the only way of doing so in Linux.

Forking is important for many reasons. One of them relates to process scalability. Apache, a popular web server, is a good example. By forking itself, Apache is able to serve a large number of requests with fewer system resources than a single-process-based server.

When a process calls a fork, the caller process becomes the parent process, with the newly created process referred to as its child. After the fork, the processes are, to some extent, independent processes; they have different process IDs but run the same program code.

Command	Description
ps	• Used to list the processes running on the computer at the time it is invoked. • It can be instructed to display running processes that belong to the current user or other users. • While listing processes does not require root privileges, killing or modifying other user's processes does.
top	• Used to list running processes, but unlike **ps**, **top** keeps displaying running processes dynamically. • Press **q** to exit top.
kill	• Used to modify the behavior of a specific process. • Depending on the parameters, **kill** will remove, restart, or pause a process. • In many cases, the user will run **ps** or **top** before running kill. • This is done so the user can learn the PID of a process before running kill.

4.7.4 Malware on Linux host

Linux malware includes viruses, Trojan horses, worms, and other types of malware that can affect the operating system. Due to a number of design components such as file system structure, file permissions, and user account restrictions, Linux operating systems are generally regarded as better protected against malware.

While arguably better protected, Linux is not immune to malware. Many vulnerabilities have been found and exploited in Linux. These range from server software to kernel vulnerabilities. Attackers are able to exploit these vulnerabilities and compromise the target. Because Linux is open source, fixes and patches are often made available within hours of the discovery of such problems.

If a malicious program is executed, it will cause damage, regardless of the platform. A common Linux attack vector is its services and processes. Vulnerabilities are frequently found in server and process code running on computers connected to the network

4.7.5 Rootkit Check

A rootkit is a type of malware that is designed to increase an unauthorized user's privileges or grant access to portions of the software that should not normally be allowed. Rootkits are also often used to secure a backdoor to a compromised computer.

The installation of a rootkit can be automated (done as part of an infection) or an attacker can manually install it after compromising a computer. A rootkit is destructive because it changes kernel code and its modules, changing the most fundamental operations of the OS itself. With such a deep level of compromise, rootkits can hide the intrusion, remove any installation tracks, and even tamper with troubleshooting and diagnostic tools so that their output now hides the presence of the rootkit. While a few Linux vulnerabilities through history have allowed rootkit installation via regular user accounts, the vast majority of rootkit compromises require root or administrator access.

Rootkit removal can be complicated and often impossible, especially in cases where the rootkit resides in the kernel; re-installation of the operating system is usually the only real solution to the problem. Firmware rootkits usually require hardware replacement.

chkrootkit is a popular Linux-based program designed to check the computer for known rootkits. It is a shell script that uses common Linux tools such as **strings** and **grep** to compare the signatures of core programs. It also looks for discrepancies as it traverses the /proc file system comparing the signatures found there with the output of **ps**.

4.7.6 The Piping Commands

Although command line tools are usually designed to perform a specific, well-defined task, many commands can be combined to perform more complex tasks by a technique known as piping. Named after its defining character, the pipe (**|**), piping consists of chaining commands together, feeding the output of one command into the input of another.

For example, the **ls** command is used to display all the files and directories of a given directory. The **grep** command compares searches through a file or text looking for the specified string. If found, **grep** displays the entire contents of the folder where the string was found.

The two commands, **ls** and **grep**, can be piped together to filter out the output of **ls**. This is shown in the output of the **ls -l | grep host** command and the **ls -l | grep file** command.