

People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research



Saad Dahleb University of Blida
Faculty of Sciences
Computer Science Department

Master Thesis In
Information System Security

Design and Implementation of an Automated Security Hardening and Management System for Linux Servers

Realized By

Yacine BENDOU
Mohamed Islam BENOAIE

Hosting Organization

ELIT-Groupe Sonelgaz

Supervisor:

Dr. Zakaria SAHNOUNE

Supervisor:

Mr. Abdelghani BENKOULAL

Academic Year: 2022-2021

Abstract

Cybersecurity is an emerging field in computer science with the goal of developing robust and secure systems. System hardening is a cybersecurity term that refers to the process of adding more protection for a system by mitigating vulnerabilities in the system. This process is based on an international guidelines and working on them can take a day for just one system. With the growth and widespread use of digital information, The scenarios of attacks became daily and the solution of the systems founders is to make updates for the editions so this process became a time consumer more an more. In order to automate and facilitate the work, we offered a web based application for ELIT to manage the data centers in a quicker and more efficient way by getting all the information of the process.

Résumé

La cybersécurité est un domaine émergent de l'informatique dont l'objectif est de développer des systèmes robustes et sécurisés. Le durcissement du système est un terme de cybersécurité qui fait référence au processus d'ajout de plus de protection pour un système en atténuant les vulnérabilités du système. Ce processus est basé sur des directives internationales et travailler sur celles-ci peut prendre une journée pour un seul système. Avec la croissance et la généralisation de l'information numérique, les scénarios d'attaques sont devenus quotidiens et la solution des fondateurs de systèmes est de faire des mises à jour pour les éditions donc ce processus est devenu de plus en plus consommateur de temps. Afin d'automatiser et de faciliter le travail, nous avons proposé une application Web pour ELIT afin de gérer les centres de données de manière plus rapide et plus efficace en obtenant toutes les informations du processus.

ملخص

يعد الأمن السيبراني مجالاً ناشئاً في علوم الكمبيوتر بهدف تطوير أنظمة قوية وآمنة. تصلب النظام هو مصطلح للأمن السيبراني يشير إلى عملية إضافة المزيد من الحماية لنظام ما عن طريق التخفيف من نقاط الضعف في النظام. تستند هذه العملية إلى إرشادات دولية ويمكن أن يستغرق العمل عليها يوماً لنظام واحد فقط. مع نمو المعلومات الرقمية واستخدامها على نطاق واسع ، أصبحت سيناريوهات الهجمات يومية وكان الحل لمؤسسي الأنظمة هو إجراء تحديثات للإصدارات حتى أصبحت هذه العملية مستهلكاً للوقت أكثر وأكثر. من أجل أتمتة العمل وتسهيله ، قدمنا تطبيقاً قائماً على الويب لـ ELIT لإدارة مراكز البيانات بطريقة أسرع وأكثر كفاءة من خلال الحصول على جميع المعلومات الخاصة بالعملية.

Contents

I	Theoretical Background	1
1	Linux Systems	2
1.1	Introduction	2
1.2	The POSIX Standard	5
1.2.1	What is POSIX	5
1.2.2	POSIX Defined Standards	5
1.3	LINUX Architecture	8
1.3.1	Hardware Basics	9
1.3.2	The Kernel	13
1.3.3	Commands and Utilities	14
1.3.4	Files and Directories	15
1.4	World of The Open Source	15
1.4.1	Open Source Software	15
1.5	Conclusion	18
2	Linux System Administration	19
2.1	Introduction	19
2.2	Package Manager	20
2.2.1	What is a Package	20
2.2.2	RPM Package manager	22
2.2.3	RPM & YUM	23

2.2.4	Debian Package manager	24
2.2.5	dpkg & Apt	24
2.3	Managing Disks and file systems	25
2.3.1	Linux Files Types	25
2.3.2	Linux File system structure	29
2.3.3	Linux File system Types	34
2.3.4	Ext, Ext2, Ext3 and Ext4	34
2.4	Managing System Users	36
2.4.1	The Identification The authentication	36
2.4.2	Users	36
2.4.3	Groups	37
2.4.4	Passwords	38
2.4.5	Users Authentication	40
2.5	Conclusion	46
3	Linux Server Administration	47
3.1	Introduction	47
3.2	Managing Services	49
3.2.1	Init & Systemd	49
3.2.2	Units	50
3.3	Network Management	54
3.3.1	Desktop	54
3.3.2	Entreprise	55
3.3.3	Remote Access	57
3.4	Virtual Servers	65
3.4.1	Types of Virtual Servers	66
3.4.2	Advantages disadvantages	68
3.4.3	Virtual Server vs. Virtual Machine (VM)	68

3.5	Conclusion	69
4	Linux System Security & Hardening	70
4.1	Introduction	70
4.2	Type of Attacks against the system	71
4.2.1	Reading data	71
4.2.2	Changing data	72
4.2.3	Denial of service	72
4.2.4	Access to computer	73
4.3	System Hardening	73
4.3.1	System Hardening Types	74
4.3.2	System Hardening Process	76
4.3.3	Server Auditing	77
4.3.4	Benchmarks	79
4.3.5	Logs System	82
4.4	System hardening Automation	85
4.4.1	Monitoring Architectures	86
4.4.2	Available solutions	88
4.4.3	Ansible Architecture	89
4.5	Conclusion	91
II	Management System and Automation of The Linux servers Hardening	93
5	Conception	94
5.1	Diagram of Classes	94
5.1.1	Description	97
5.2	Use Cases Diagram	103
5.2.1	Description	105

- 6 Development and Implementation 107**
- 6.1 Introduction 107
- 6.2 Tools Used: 107
 - 6.2.1 Visual Studio Code : 107
 - 6.2.2 XAMP Server : 107
 - 6.2.3 Laravel v9: 108
 - 6.2.4 Node JS 110
 - 6.2.5 YAML 111
 - 6.2.6 Bootstrap 111
 - 6.2.7 GitHub 111
 - 6.2.8 AdminLte 111
 - 6.2.9 Ansible 112
- 6.3 Architecture of the Solution 112
- 6.4 How the Application works 114
- 6.5 The Process of Creation 115

List of Figures

0.1	company organization chart	6
0.2	The Mind map of the project	8
1.1	The Principle of the operating system[1]	3
1.2	UNIX logical architecture [1]	8
1.3	UNIX architecture[9]	9
2.1	mimetype command results	28
2.2	file command results	29
2.3	Linux File Hierarchy Structure [10]	31
2.4	Linux file system structure[13]	34
2.5	PAM Diagram [14]	42
2.6	Permissions List	43
2.7	Portions of the permissions	43
3.1	Types of server virtualization [25]	66
4.1	Ansible Inventory	90
4.2	Ansible Playbook	91
4.3	Ansible Architecture [28]	91
5.1	Diagram of classes	96
5.2	Diagram of classes / permission / Role / User	97
5.3	Diagram of classes / Playbook / Regex / Expressions	98

5.4	Diagram of classes / ScanEng / Audit / Playbook / Template/ user	99
5.5	Diagram of classes / Server/ Audit / Tags	101
5.6	Diagram of classes / File / Results /Audit_server	102
5.7	Diagram of the use cases	104
6.1	Project Architecture	113

List of Tables

- 2.1 Absolute mode guide 44
- 2.2 Absolute mode references 44
- 2.3 Symbolic mode guide 45
- 2.4 Symbolic mode identity guide 45
- 2.5 Symbolic mode relationship 45

- 3.1 Telnet Advantages Disadvantages 60

Appreciations

First of all, we would like to thank God for giving us the Courage and patience to develop our work, and thus accomplish our studies.

We sincerely, want to Thank our promoters Mr.Zakaria Sahnoune and Mr. Abdelghani Benkoulal, who have shown themselves to be attentive and always available throughout the realization of this project as well for the inspiration, the help and the time that they kindly shared with us.

We express our gratitude and appreciation to the department professors of computer science that have given us so much to be what we are today.

Dedications of Bendou Yacine

It is with genuine gratitude and warm regard that I dedicate this work for my Family, Dad Abdelkader, Mom Nacera, my Lovely Khalti Ouahiba and Ami Mustapha Considered as my second parents, my siblings, my sister Sara, my brothers Mahdi and Ibrahim and my brother in law Abdelouahab, the beloved Yanis and Dania.

I'd like to dedicate my work to many brothers Younes Hamid, Merouane, Tahar, Fady, Hicham, and my friends Ghofrane, Saya and the beloved ones.

This project is dedicated to my greatful teacher Mr. Derdiche and Ms. Bacha for having a the great impact on my life.

I also dedicate this Thesis To My Doctors Dr.Gouichiche, Dr.Ghezali, Dr.Lyazidi, Dr.Meftah, Dr.Guemri.

"Yacine"

Dedications of Benaïe Islam

I dedicate my work to ; My Mother who has always supported me and pushed me to do and be better.

My brother who always makes me smile and cheerful ,My grandmother to whom i wish a long and healthy life ,My dearest Friends who are always by my side.

To my teachers Ms Bacha and Mr Zahar who helped me understand getting better at Computer Science . And to my dearest cousin Mr Belkorane Hedjala Mohamed who stood by my side to get involved in this project .

"Islam"

General introduction

The Hosting Organization

Specialized in information and communication technologies, *El Djazair Information Technology* "ELIT" is an Algerian company with more than 300 computer engineers, more than 40 customers, and highly available and secure infrastructures.

Beyond the aspects recognized in the IT field, computer networks, website development, electronic messaging, etc., **ELIT** ensures the security of information systems via a state-of-the-art security platform, with a 100% Algerian human resource. With Multiple approved running solutions for the management like Finance and Accounting Management System "**HISSAB**", Commitment Management System "**ILTIZAMATE**", Cash Management System "**MALIYA**", Human Resources and Payroll Management System "**NOVA**", Supply and Inventory Management System "**ATTAD**", Occupational Medicine Management System "**GMT**", Billing management system "**FAWTRA**", Electronic Document Management System "**GED**", Mail management system "**BARIDI**", Mandate management system "**ELIT Mandats**", Workplace risk prevention system "**AMLT**", Legal Cases Management System "**SAAJ**", Management System for cars parks "**Parc Auto**".

Even more in the It security as being a specialized community in it. ELIT offers a platform for the IS security awareness ”**ELIT Security Awareness**”, A project Compliance Audit ”**AUDELIT**” measuring the performance and ensuring the protection, monitoring, and controlling of information and network security is a primary necessity for each company. Another project was offered to ensure upstream security monitoring and the analysis and processing of incidents downstream thanks to a multidisciplinary team of experts responsible for reacting in the event of IT security incidents already named as Security Incident Alert and Response Center ”**CSIRT Sonelgaz**”. To Monitor the environment: detect threats, opportunities, and trends; Ellit offers **ELIT VEILLE**.

Context

Hardening in the computing field is usually referring to the process of securing a system focusing on reducing the surfaces of vulnerabilities that are more known and exploitable when a system is set to serve multiple functions; in general, a mono-function or a single function system is safer and more secure than a multipurpose one. Changing default passwords, removing new software, minimizing usernames or logins, and removing unwanted and none helpful services are typical ways to reduce the risk of attacks.

There are various methods of hardening Unix, Linux, and Windows systems. Applying patches and updates to the Kernel; correctly managing the network by closing and opening only the needed ports, with all the development in the field intrusion-detection systems are one of the recommended ways, firewalls and making in place an intrusion-prevention system which can alarm for any attacks on the system. There are also hardening scripts and tools like Lynis, Bastille Linux, JASS for Solaris systems, and Apache/PHP Hardener that can,

for example, deactivate unneeded features in configuration files or perform various other protective measures.

While system hardening requires a great and continuous effort, it provides substantial benefits for organizations, such as a higher level of security, the main purpose of system hardening techniques and mechanisms is to reduce the attack surface; this offers a significantly lower risk of malware attacks, unauthorized access, data leaks, or other malicious undesired activity. In addition, this procedure offers better system functionality. System hardening best practices often involve reducing the number of programs and functionality. This translates into fewer operational issues, a reduced chance of misconfiguration, which can affect user operations, fewer incompatibilities, and a reduced chance of cyber-attacks that hurt the user functionality. More than that, simplified compliance and auditing are one of the gains; those techniques can help in turning a complex environment into a simpler one with fewer programs and accounts and a stable, predictable configuration; this translates into a more straightforward and transparent environment which is simpler to monitor and audit.

Nowadays, with the huge increase of the IT domain and the connection of all the types of people with different mindsets and purposes in this life, system hardening has become an obligation before moving to the production level for any organization or company or even simple users with an information system based infrastructure or a simple computer with some sensitive data. For a simple user with one operating system or even two, the heaviness of the process can not be felt. Still, for organizations with multiple systems, physical or virtual, the process costs a loss of time for the employees and a massive break in the financial level.

This project is part of the strategy of the ELIT to satisfy the clients and Saving Money and time of the organization. The project took place within the department of the Information systems security working on a mission of being in charge of the security of the information assets, of the information systems of the Sonelgaz Group and of guaranteeing the integrity, confidentiality, availability and traceability of the data of all the information systems of the company.

Our goal in this project is to propose a solution and implementing it for the automation management of the Linux systems hardening.

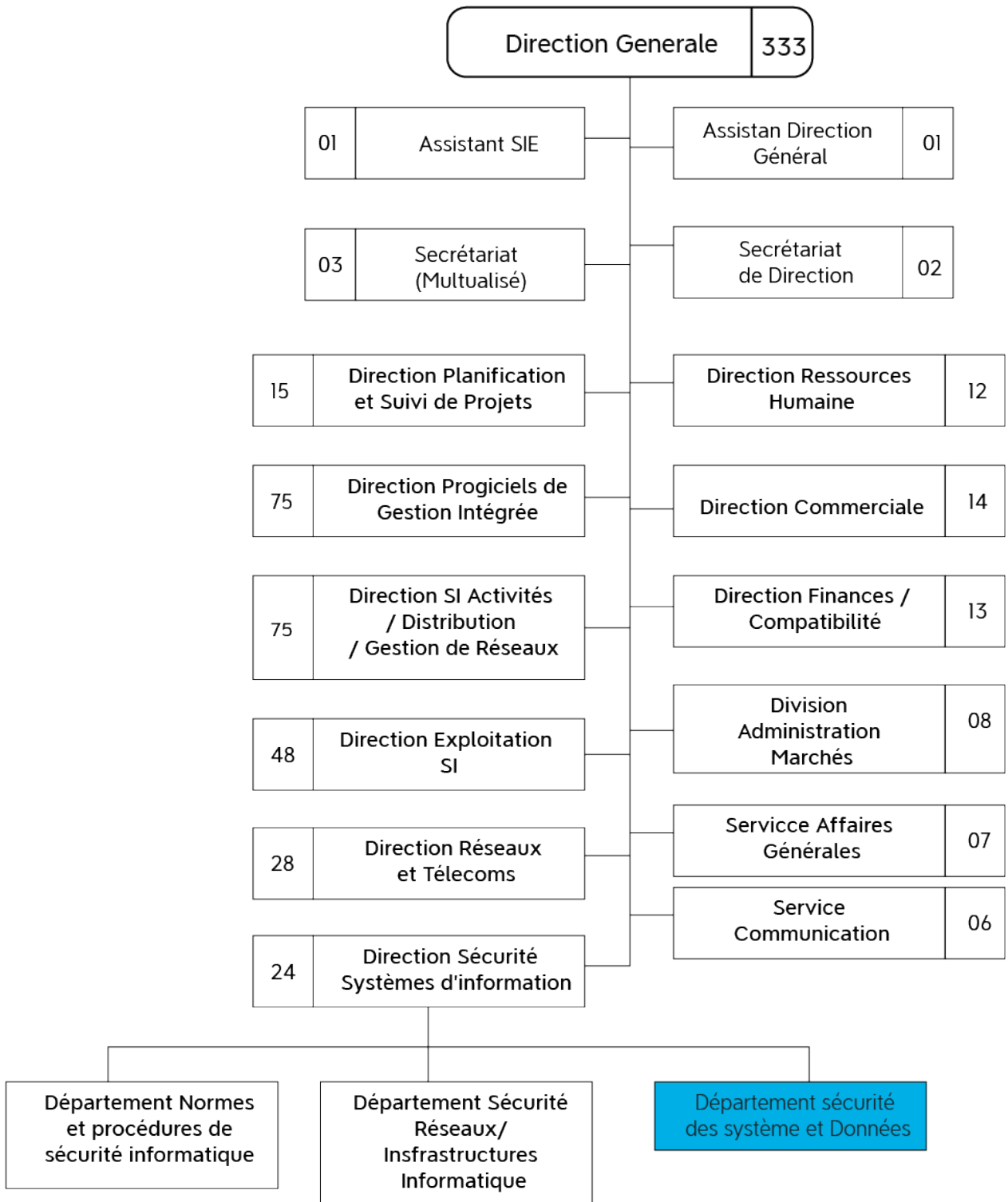


Figure 0.1: company organization chart

Problematic

ELIT is an enterprise owning two datacenters with a large number of servers that offers informatics solutions to the group Sonelgaz and multiple ministries. Given the strategic importance of the group Sonelgaz and the importance of the information systems provided, ELIT is obliged to secure its infrastructure against any risk. Therefore, one of the principal axes is the hardening and the tracking of the servers after passing to the production; this mission becomes more and more complex given the increasing number of the hosted servers in the datacenters and the absence of an automated tool.

Objectifs

Intending to alleviate the pain from the department IT responsible for the hardening and retrieve the lost time in this process, ELIT planned these objectives to be realized:

Implementing a system of automation Linux systems hardening

Implementing a system of auditing Linux systems hardening

Approach Adopted

To achieve the goals already mentioned before and develop the systems in the desired way, a well-defined mind map was created, illustrating an iterative and agile path to reach the end.

Organization of the Document

This document is divided into two parts. The first one describes state of the art of system hardening and automation: The first chapter is designed for the

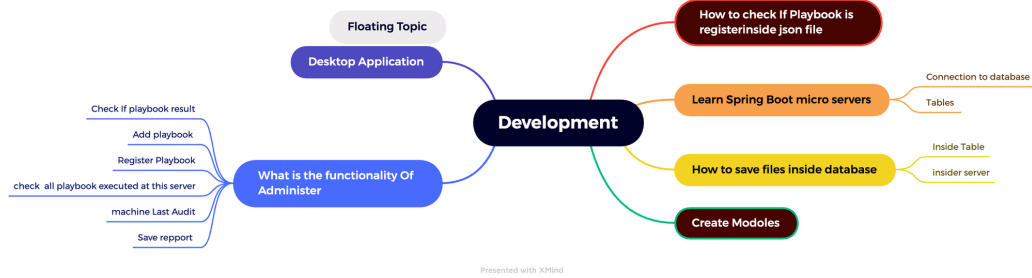


Figure 0.2: The Mind map of the project

general view of the Linux system and its architecture. The second chapter is for the Linux systems Administrator, which is essential to understand how they work. The third one is made for the Linux servers to understand the connection and multiple services offered by the Linux servers. The fourth one is for the security of the Linux systems to know how the Linux is made safe. the last one is for the hardening and the available tools of automation of the configuration

Part I

Theoretical Background

Chapter 1

Linux Systems

1.1 Introduction

Pressing on the starting button of the computer makes the electrical power flow inside the hardware, which starts the working space and makes the computer usable. For the machine to be functional, a process is completed to charge the programs in the computer's memory and then executed. The aim is to simplify the use of the machine for both the power and non-power users. The whole of these programs forms an operating system whose purpose, based on its name, is performing specific operations on the system. It is a transitional layer between the user and the material.

The Application Programming Interface, or **API** for short, is provided by the Operating system for the programmers to help them be more productive by ignoring some basic task management, some of which can be listed below

- Memory
- Devices access
- Disk access

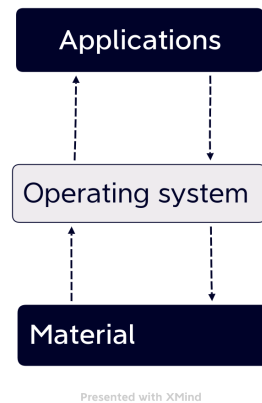


Figure 1.1: The Principle of the operating system[1]

- Programs
- Security
- Gathering of the information

When speaking about the Graphical User Interface, it is worth mentioning that historically, this tool was not packed with the operating system out of the box as it was not considered mandatory. It is a group of programs bundled together to simplify the use for all users, professionals and non-professionals alike. It was considered an unnecessary hardware resource consumption layer until the Windows system entered the field as a law breaker where Graphical User Interfaces were shipped out of the box. Of course, the Linux system offers GUIs, but on the contrary, users need to install them as optional programs [1].

Linux falls into is one of the Multi-tasking and Multi-user based operating systems. These traits are considered two of this system's primary features, in addition to being Portable, Secure and Open source, having a hierarchical file system, and supporting a powerful Shell. The roots of the Linux system can be traced back to Unix.

Multi Processing: The ability to manage multiple processes and needs physical conditions which are served by having multiple microprocessors or logically by using technologies like Hyperthreading. [1]

Multi-user: Multiple users can access systems with different levels of permissions with the state of being connected or not. [1]

Portable: The Linux OS can run on different types of hardware, and the Linux kernel supports the installation of any hardware environment. [1]

Open source: The source code of the Linux operating system is freely available, and multiple teams are working together to improve the performance of the Linux operating system. [1]

Hierarchical file system: The Linux operating system provides a typical file structure in which user files or system files are arranged. [1]

Security: The Linux operating system facilitates user security systems with various authentication features such as controlled access to specific files, password protection, or data encryption. [1]

Shell The Linux operating system allows a unique interpreter. Users can employ such programs to execute operating system commands. It can be applied to perform different types of tasks. [1]

Ken Thompson made the starting point in 1969 of the Research Group at Bell Laboratories began experimenting on a multi-user, multi-tasking operating system using an otherwise idle. After Dennis Richie joined the project and other members of the Research Group, the early versions of Unix were produced. An earlier project was the stronger base point for Richie, **MULTICS** and the **Unix** is itself based on the name **MULTICS**. The early versions were written in assembly code, but the third version was rewritten in a new pro-

programming language, C. C was designed and written by Richie expressly as a programming language for writing operating systems. Unix moved out of the laboratory and into mainstream computing, and soon most major computer manufacturers were producing their own versions.^[2]

For simple needs, the operating system Linux was offered by Linus Torvalds to serve. Unix was the reference of the author, and it is important to mention that the Linux contains no **UNIX** code; it is a rewrite based on published **POSIX** standards.

1.2 The POSIX Standard

1.2.1 What is POSIX

POSIX stands for Portable Operating System Interface. It is a series of standards specified by the **IEEE** to maintain compatibility between operating systems. Therefore, any **POSIX**-compliant software should be compatible with other **POSIX**-compliant operating systems. Because of this, most of the tools we use on Linux and Unix-like operating systems behave almost the same. For example, if we use the `ps` command, it should behave the same on OpenBSD, Debian, and macOS.

1.2.2 POSIX Defined Standards

The C API

In terms of the C languages, standards are defined by the **POSIX**. Therefore, source code level programs can be ported to other operating systems. However, it can also be implemented in any standardized language.

In addition to the **C API**, **POSIX** has also added rules for writing programs such as Pointer-type initialization and execution concurrency. It also enhances rules for the synchronization of the memory, such as minimizing memory modification when it's already in use. More than that, security mechanisms are also stated for directory protection and file access. [3]

File Formats

POSIX defines the format of strings used in files, standard output, standard error, and standard input rules as shown in the following expression. [4]

`<format>, <arg1>, ..., <argN>`

Environment Variables

The environment file that the login shell processes when the login is successful. By convention, variable names should only contain uppercase letters and underscores. The name can also include a digit, although the **POSIX** standard does not recommend putting the digit at the start of the name. [5]

Locale

An environment variable is a variable that we can define in an environment file, which the login shell will process upon successful login. The locale defines the language and cultural conventions used in the user's environment. Each locale consists of categories that define the behavior of software components, such as DateTime formats, currency formats, and number formats.

Character Set

A character set is a collection of characters consisting of the code and bit pattern for each character. As we all know, computers can only understand binary characters, so a character set represents the symbols a computer can handle... For that, a standard character set that conforms to the one defined by **POSIX** is needed.

Regular Expressions

A regular expression or **RE** is a string that defines a search pattern for finding text. The standard C library implements RE and is used as a backend by programs like **awk**, **sed**, and **grep**. A **POSIX**-compliant implementation can use either Basic Regular Expressions (**BRE**) or Extended Regular Expressions (**ERE**). **BRE** provides basic text search symbols, while **ERE** supports more symbols. Most **POSIX**-compliant utilities rely heavily on **BRE**, although advanced text editing utilities also support **ERE**^[7]. Additionally, both **BRE** and **ERE** have several requirements:

- **BRE** and **ERE** should use NULL-terminated strings.
- Literal escape sequences and newlines produce undefined results. Therefore, our program should treat them as normal characters.
- **POSIX** does not allow explicit NULL characters in **REs** or text to match.
- By default, our implementation should be able to do case-insensitive searches.
- The length of our RE should not exceed **256** bytes

Directory Structure

Most major Linux distributions conform to the File System Hierarchy (**FHS**) standard. **FHS** defines a configurable tree-like directory structure. The first directory in the Hierarchy is the root directory from which all other directories, files, and special files branch.[8].

1.3 LINUX Architecture

Unix has been around for nearly five years, shaping the modern operating system, key software technologies and development practices. As Linux being based on the **Unix** Architecture the last figure represents a simplified diagram of the internal structure of a **UNIX** operating system.

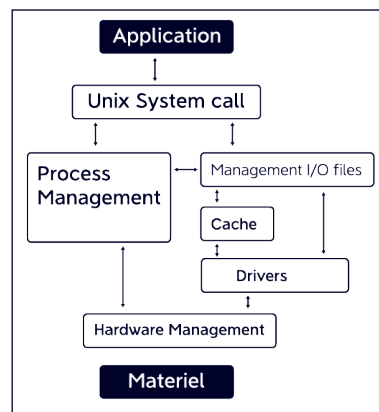


Figure 1.2: UNIX logical architecture [1]

The top layer, named application, is the user layer, and the bottom one, Materiel, is the hardware. Between the two, we can find the work offered by an operating system which can be listed below:[2]

- System Calls
- Management of the Processes

- management of the Input/output of files
- Cache management
- Drivers

Here is a basic block diagram of a Unix system:

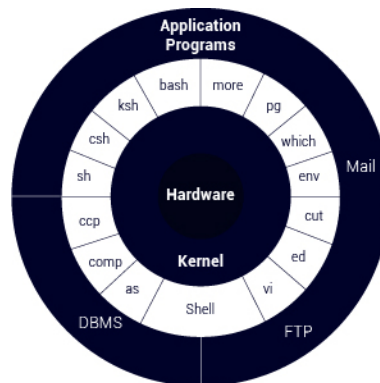


Figure 1.3: UNIX architecture[9]

1.3.1 Hardware Basics

An operating system must work closely with the hardware system on which it is based. The operating system requires certain services that can only be provided by hardware. To finish, To understand the Linux operating system, you need to understand the basics of the underlying hardware.

When looking at the PC from the outside, the most visible components are the system box, keyboard, mouse, and video monitors. There are several buttons on the front of the system box and a small display showing some Digital and floppy disk drives. Most systems these days come with a CD-ROM if you feel you must protect your data, and then there's also a tape drive for backups. These devices are collectively referred to as Like the periphery.

While the CPU can take full control of the system, it's not the only smart device. All Peripheral controllers, such as IDE controllers, have a certain level of intelligence. Inside the computer, You will see one that contains a CPU or microprocessor, memory, and a number of slots for ISA or PCI peripheral controllers. Some controllers like IDEs, The hard disk controller, can be installed directly on the system board.

The CPU

The CPU, or rather the microprocessor, is the heart of every computer system. Microprocessor computing Perform logical operations and manage data flow by reading instructions from memory, then executing them. In the early days of computing, the functional component was the microprocessor, A separate (and physically large) unit. At that time, the term central processing unit was coined. Modern microprocessors combine these components into an integrated circuit etched in very small dimensions, a piece of silicone. Registers of the microprocessor are Internal memory used to store data and perform operations on it. Actions performed can push the processor to stop and jump to somewhere else in memory. These tiny building blocks give modern microprocessors nearly limitless capabilities, Millions or even billions of instructions per second. Instructions must be fetched from memory when executed. Instructions can help themselves Reference data in memory, and that data must be fetched from and stored in memory suitable. The size, number, and type of registers in a microprocessor depend entirely on their type.

The Memory

All systems have a memory hierarchy with different speeds and sizes of memory in different places Hierarchy. The fastest memory is called cache memory, and

that's what it sounds like - memory Used to save or cache the contents of main memory temporarily. However, this type of memory is very fast and Expensive, so most processors have a small amount of on-chip cache and more systems Based on the (integrated) cache. Some processors have caches that contain instructions and data. But others have two, one for instructions and one for data.

Cache and main memory must be coherent (coherent). In other words, if a noun memory is cached in one or more locations, then the system must ensure that Caching and storage are the same. The task of cache coherence is partially done by hardware, Partly determined by the operating system. This also applies to many important system tasks, where hardware and software must work closely together to achieve their goals.

Buses

The various components of the system board are connected to each other through multiple connection systems called the bus. The system bus is divided into three logical functions; the address bus, the data bus, and the control bus. The address bus specifies the storage location (address) of the data transfer that the data bus holds the transmitted data. The data bus is bidirectional; it allows data to be read into the CPU and written by the CPU. The control bus contains various lines for timing and control signals of the entire system. There are many kinds of buses, such as ISA and PCI bus are popular A method of connecting peripherals to the system.

Controllers and Peripherals

Peripherals are real devices controlled by a controller chip in the system, such as a graphics card or hard drive on a circuit board or a card inserted into it.

IDE hard disk is controlled by IDE controller chip and SCSI. The hard disk passes through the SCSI hard disk controller chip and so on. These controllers are connected to the CPU and to each other through various buses. Most systems built today use PCI and ISA buses to connect them together, The most important system component. Controllers are processors like the CPU itself. They can be thought of as Smart assistants for CPUs. The CPU has overall control over the system.

All controllers are different, but they usually have registers that control them. Software running on The CPU must be able to read and write these control registers. Registers can contain status. Describe an error. Another one can be used for control purposes; changing the mode of the controller. Each controller on the bus can be individually addressed by the CPU, making it a software device. The driver can control it by writing to its registers. The IDE Ribbon is a good example because it gives you. Each driver on the bus can be accessed individually. Another good example is the PCI bus that, which makes it possible for Each device, such as a graphics card, can be accessed independently.

Address Spaces

The system bus connects the CPU to the main memory and is separate from the connection bus, The CPU, and the system's hardware peripherals. In summary, the storage space owned by the hardware I/O space is called I/O space. The I/O space itself can be further subdivided, but we won't be too worried right now. The CPU can access system memory and I/O memory space, while the controller itself can only access system memory indirectly. Then only with the help of the CPU. Sometimes the controller needs to read or write large amounts of data directly from the system memory. For example, when user data is written to disk. In this case, Direct Memory Access (**DMA**) controllers

are used to allow hardware peripherals to directly access system memory, but This access is strictly controlled and monitored by the CPU.

Timers

All operating systems need to know the time, which is why modern PCs include a special peripheral called Real-Time Clock (RTC). This provides two things: a reliable time of day and an accurate time interval. The RTC has its own battery, so it can continue to run even when the PC is not turned on. The PC always gets the correct date and time. Interval timer enables operation A system for precisely planning important work.

1.3.2 The Kernel

The kernel is the base of the operating system. It interacts with the hardware and most tasks such as memory management, task scheduling, and file management [9]. Linux kernel development is closely monitored by the Linux kernel development team [11].

On a purely technical level, the kernel is the middle layer between hardware and software. Its purpose is to pass application requests to the hardware and act as the low-level driver to be addressed equipment and components of the system. However, there is another interesting point core. The kernel can be regarded as an enhanced machine that, in the view of the application, abstracts the computer on a high level. For example, when the kernel addresses a hard disk, it must decide which path to use to copy data from disk to memory, where the data reside, which commands must be sent to the disk via which path, and so on. Applications, on the other hand, need only issue the command that data are to be transferred. How this is done is irrelevant to the application — the

details are abstracted by the kernel. Application programs have no contact with the hardware itself, only with the kernel, which for them, represents the lowest level in the Hierarchy they know — and is, therefore, an enhanced machine. The kernel and shell are the heart and soul of the operating system.

User vs Kernel mode

Kernel component code runs in a unique privileged mode called kernel mode and has full access to all computer resources. This code demonstrates a single process, runs in a single address space, and does not require context switching. Therefore, it is very fast and efficient. The kernel runs all processes and enables these processes to perform various services in the system. It also facilitates secure access to processes on the hardware. Support code that is not required to run in kernel mode is in the system library. User programs and other types of system programs are implemented in user mode. It does not include access to kernel mode and system hardware. User utilities/programs use system libraries to access kernel functions for low-level system tasks.[\[13\]](#)

1.3.3 Commands and Utilities

When using an operating system, the user must learn how to use the system command. `ls`, `mv`, `cat`, and `mkdir`, etc., are a few examples of commands and utilities. There are over 250 standard commands, plus many others available through third-party software. All the commands come along with various options.[\[9\]](#)

1.3.4 Files and Directories

Files are the only way to organize the data in the Unix. All files are then organized into directories. These directories are further organized into a tree-like structure called the filesystem with different types of files. Directory files, Non-directory files, REGULAR files, SPECIAL files, Symbolic LINK files.[11]

1.4 World of The Open Source

When evoking the term "Open Source", the mainstream is of this concept is that it refers to a category of software that offers an abundance of freedom in the usage and reimplementing of both the effort and its source code, but in reality, this interpretation deprives "Open Source" from an important quantity of what it really stands by delimiting it to one of its various fields, "Open Source Software", the concept itself originated in the field, but is nowadays broader and includes every project, be it in the field of computer science or not, following what is called "the open source way".

1.4.1 Open Source Software

It is agreed on that Open Source Software is software with source code that anyone can inspect, modify, and enhance, but in addition to this criteria, it is software whose distribution terms must comply with The Debian Free Software Guidelines (**DFSG**) present on the official Debian website

Free Redistribution

The license of a Debian component may not restrict any party from selling or giving away the software as a component of an aggregate software distribution

containing programs from several different sources. The license may not require a royalty or other fee for such a sale.

Source Code

The program must include source code and must allow distribution in source code as well as compiled form.

Derived Works

The license must allow modifications and derived works and must allow them to be distributed under the same terms as the license of the original software.

Integrity of The Author's Source Code

The license may restrict source code from being distributed in modified form only if the license allows the distribution of patch files with the source code for the purpose of modifying the program at build time. The license must explicitly permit the distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software. (This is a compromise. The Debian group encourages all authors not to restrict any files, source or binary, from being modified.)

No Discrimination Against Persons or Groups

The license must not discriminate against any person or group of persons. No Discrimination Against Fields of Endeavor The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business or from being used for genetic research.

Distribution of License

The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

License Must Not Be Specific to Debian

The rights attached to the program must not depend on the program's being part of a Debian system. If the program is extracted from Debian and used or distributed without Debian, but otherwise within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the Debian system.

License Must Not Contaminate Other Software

The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be free software.

Example Licenses

The **GPL**, **BSD**, and Artistic licenses are examples of licenses that we consider free.

As mentioned in the guidelines, Open Source manifests itself in the form of different licenses adhering to the previous criteria. For instance, the Gnu Public License, or **GPL** for short, follows the Open Source requirements in addition to new criteria, which are Copyleft licenses. Non-permissive or Copyleft software is basically software that forces the one benefiting from the Open Source program to make the source code available to the user so that they are also able to benefit

from it and so on. Adhering to Permissive or Non-copyleft licenses makes it so that the project can be made Proprietary.

1.5 Conclusion

Linux is a free operating system, which allows you to regain control over your computer. For this reason it is widely used on web servers, but it is of course perfectly suited for personal use. To discover it, several methods exist, depending on the use you want to make of it, for its usage general way will be discussed in the up coming section

Chapter 2

Linux System Administration

2.1 Introduction

Starting from the linguistic side with the word administration and exploring it, it refers to the arrangements and tasks needed to control the operation of a plan or organization [15], in another way, it is the management of any office, business, or organization; direction [16]. As an abstract of those definitions, the word administering is all about managing the given physical and moral resources in the right way to accomplish the tasks in order to achieve the set goals

Defining that word from a technical side and specifically in the domain of informatics administration or the role of administrator refers to the notion of managing some specific processes (installation, maintenance, improving, supervising, security). The administrator gets some personal roles and permissions to the data and functions of the system that other simple users can not have, and these last give the opportunities to manage the system in the desired way.

After getting the access, the Linux systems have on them a tool that helps the worker to facilitate the tasks, and it is known as the Package Manager.

2.2 Package Manager

In the starting days of Linux, adding software was not an easy process. Doing it was based on surfing the web and getting the source code from the group that made it, compiling it to a runnable version, and saving it on your computer. With some luck finding it already compiled is not a miracle, and it would run on your computer.

2.2.1 What is a Package

For different operating systems, none UNIX/Linux based, there is no given interactive installation program (no install.exe). Some editors try to facilitate this process by providing scripts for the installation, and also, these last just based on decompressing and unarchiving files.

In Linux, it is traditional to get the tools and the products in the form of packages. A package is a file that contains the product to install and the rules defined by the creator. Rules can be different :

Dependencies Management

The installation of a program depends on other programs, so the presence of these lasts is mandatory for the begging of the setting up. In the case of absence, the process will be directed to fail.

Pre-Installation

Before starting, predefined tasks must be done to prepare the infrastructure, like loading the file's permissions, creating the needed files and directories, and

other tasks that need to be accomplished before.

Post-Installation

After finishing the installation with success, state other actions and processes must be done, such as setting the settings of the configuration file. The different Linux distributions have a diversity of package managers. Each one and its derivatives has its specific manager. Highlighting the point on the form of the packages, they could be defined as a tarball containing executable files which are considered as commands in Linux, documentation, configuration files, and libraries. The construction of a tarball is made by gathering multiple files in one file for convenient storage or distribution. Installing a software from a tarball each file will be spread in the appropriate directory (`/usr/share/man`, `/etc`, `/bin`, and `/lib`, to name a few).

Even if creating a tarball and dropping a set of programs onto the system is considered doable in an easy way, the following tasks are considered mountain climbing: software dependencies, documentation of the software, default files of the program, and Updates of the software.^[1]

Finding solutions to These difficulties was one of the major interests of the editors. In this evolution, the packages developed from simple tarballs to more complex packaging formats. **RPM** And **DEB** are the main references of the Linux distributions.^[1]

DEB (.deb) packaging: the focus of The Debian GNU/Linux project was on creating **.deb** packaging which is used by Debian and other distributions based on **Debian** (Ubuntu, Linux Mint, KNOPPIX, and so on). Linux distributions could install, manage, upgrade, and remove softwares by using tools

such as **apt-get** and **dpkg**[1]

RPM (.rpm) packaging: the other manager was originally named Red Hat Package Manager renamed after to **RPM** Package Manager, **RPM** is the preferred package format for Red Hat distributions (**RHEL** and **Fedora**), **SUSE** and others based on **Red Hat** distributions (**CentOS**, Oracle Linux, and so on). At first **rpm** command was the command to manage the RPM manager and was updated and replaced after by the **yum** command in order to improve the RPM facility.

2.2.2 RPM Package manager

An **RPM** package is a collection of files needed to provide functionality, such as word processors, Photo viewers, or file servers. In **RPM**, commands, configuration files, and documentation constitute the functionality of the software. However, **RPM** files also contain metadata. It stores information about the contents of that packet, where the packet came from, what It must be running, along with other information.

Before diving into **RPMs**, you can learn a lot from **RPM**'s name the package itself. Find the name of the **RPM** package currently installed on the system (like the Firefox web browser) You can type the following into Fedora's shell, or **Red Hat** Enterprise Linux:

The command: `# rpm -q firefox`

The result: `firefox24.7.01.el7_x0.86_64`

The last result can show that the basename of the package is Firefox. The release number is 24.7, the version number is 1. The firefox package was built for Red Hat Enterprise Linux 7.0 (el7_0) and is compiled for the x86 64-bit architecture (x86_64).

2.2.3 RPM & YUM

Yellowdog Updater Modified (YUM) project aims to solve management problems RPM package dependencies. His main contribution is to stop thinking about **RPM** Packages as separate components and treat them as part of a larger software repository.^[17]

With repositories, the problem of dealing with dependencies doesn't fall on the runner's software installed but on a Linux distribution or third-party software provider provided by the software. For example, the Fedora Project is important. Make sure every package in their Linux distribution requires every component to be Resolved by another package in the repository.

Repositories can also be built on top of each other. For example, the repository *rpmfusion.org* It is assumed that the user already has access to the Fedora main repository. So if a package rpmfusion.org installation requires libraries or commands from the main Fedora Repository, Fedora packages can be downloaded and installed at the same time as you Install the rpmfusion.org package.

This is the basic syntax of the yum command:

```
# yum [options] command
```

The result of the yum install package command is to copy the requested package From yum repository to the local system. The files in the package are converted to Necessary filesystems (**/etc**, **/bin**, **/usr/share/man**, **etc.**). Related Information Packages are stored in the local RPM database, where they can be queried.

2.2.4 Debian Package manager

The manager of this Linux distribution is the **dpkg** which refers to **DEBIAN PACKAGE**, it is the **rpm** responsible for the Debian distributions and all the derivatives of it. The same or similar role listed before of rpm is done by the **dpkg**. The Debian packages are defined by the **.deb** extension, they dispose of the same information and ways like the rpm package. The **dpkg** command is responsible for the installation, creation, removal, and management of the Debian packages.

There are various tools for managing Debian packages, ranging from graphical or text-based interfaces to low-level tools for installing packages. All available tools depend on lower-level tools to function properly and are presented here in order of decreasing complexity. It's important to understand that higher-level package management tools like aptitude or synaptic depend on apt, which in turn relies on **dpkg** to manage packages on the system.[\[17\]](#)

2.2.5 dpkg & Apt

Whether it be for the **rpm** and the **dpkg**, the problem is the same: the two manage the dependencies of the packages, authorize or not their use, but do not resolve them here, the use of apt occurs it resolves the problems. **APT** refers to *Advanced Packaging Tool*. Without specifying a local or distant package the apt takes in consideration the packages in the CD,DVD, in a local repository, in internet (**FTP,HTTP**) ,etc. [\[1\]](#)

2.3 Managing Disks and file systems

A Linux file system is a structure that stores all information on a computer. In fact, this is one of the defining characteristics of the **UNIX** system on which Linux is based. Almost everything you need to identify on your system (data, commands, symlinks, devices, and directories) are represented by elements in the file system. Knowing where things are and knowing how to bypass file systems from the shell is a key skill in Linux.

In other terms, a Linux file system is a structured collection of files on a drive or partition. A partition is a segment of storage that contains some specific data. We can have different memory partitions in our machine. Typically, each partition contains a file system. General computer systems need to store data in a systematic way so that we can easily access files in less time. It stores data on a hard disk drive (**HDD**) or equivalent storage type. File system maintenance can be due to the following reasons:

Computers store data primarily in **RAM** memory; data may be lost when you shut down. However, non-volatile **RAM** (flash **RAM** and **SSD**) can be used to save data after a power outage.

Storing data on a hard drive is preferable to standard **RAM** because **RAM** costs more than disk space. Compared with memory, the cost of hard disks is gradually decreasing.

2.3.1 Linux Files Types

The file type helps us identify the type of content stored in the file. Linux supports seven different file types. These file types are regular files, directory

files, link files, special character files, block special files, socket files, and named pipe files.

Regular or ordinary files

Regular or normal files store data of various content types such as text, audio, video, images, scripts, and programs. There are hundreds of content types. On Linux, plain files can be created with or without extensions. An extension is a set of characters used with a filename to give it a special identity or to group it with files of a similar content type. Files of various content types often use well-known file extensions for easy identification and handling.

Although file extensions are not required for Linux filesystems, you should still use them. They help us identify the type of content stored in the file. For example, if a file has a .mp4 extension, you probably know it's a video file.

Directory files

The hierarchy of the directories, which is presented in the section after, file systems are based on the directories which organize files in a suited way. Directories are also files, but instead of storing data, they are where other files are stored. The directory uses directory entries to store the location of files placed in the directory. Each directory entry stores a unique name and location for a specific file.

Special files

For the hardware devices such as hard drivers, monitors, terminal emulators, printers, and the CD/DVD drives are treated by the Linux as special files. So this makes the access to files and devices the same way for the applications.

This feature makes developing programs on Linux easier and more flexible. This type presents two subtypes, the first one a *special character file* which is specific for devices that transfer data in bytes, such as a monitor or a printer. The second one concerns the devices that transfer data in blocks, and this type is called *special block file*.

Link files

This type of file is the key to using files with a different filename and from different locations. A link file is considered a pointer to another file, and here we can find two types of links. The first one is the Hard Link, whose purpose is to create a mirror copy of the file. This type can not be created for a directory or a file on another filesystem. The second type, named the symbolic link or soft link, can manage the lack of the previous type. It creates a pointer to the original file.

Socket files

Sockets are communication endpoints that applications use to exchange data. For example, when an application wants to communicate with another application, it connects to that application's socket. Any application that provides services to other applications or remote clients uses sockets to accept connections. Each socket has an associated IP address and port number, allowing it to accept connections from clients.

Sockets are very complicated. To simplify the communication process between local applications, Linux uses socket files. Socket files allow local system applications to exchange data without having to go through the complicated process of networking and sockets.

Socket files are the special files that use a file name as their address instead of an IP address and port number. `sendmsg()` and `recvmsg()` are the methods used for system calls to enable communication between local applications.

Named pipe files

Linux allows us to send the output of any process or command as input to another process or command. This function is called Pipe. Pipes only work if both processes are started by the same user and in the same parent process space.

In the case of not executing processes under the same permissions and same user names, standard pipes do not respond. For these circumstances, named pipes are the solution. They are similar to the standard pipes, except that they can be accessed as part of the filesystem. Named pipe files are also named as the **FIFO** (*First In First Out*) files. They are empty pipe files. The kernel processes named pipe files without writing them to the file system and can be found anywhere in the file system.


How to identify the type of a file

For a complete list of content types and file extensions supported by Linux systems, see the `/etc/mime.types` file. **MIME** (*Multipurpose Internet Mail Extensions*) provides uniform names and classifications for file content types.

```
(alpha@alpha)-[~]
└─$ mimetype /etc/passwd
/etc/passwd: text/plain
```

Figure 2.1: mimetype command results

There are many ways to identify the type of file in Linux. The easiest way is to use the **"file"** command. To find the type of file, provide the name of the file as a parameter. For example, to determine the file type of an **"abc"** file, use the following command. The output of this command displays not only the type of the specified file but also the type of content stored in the specified file.



```
(alpha@alpha)-[~]
└─$ file /etc/passwd
/etc/passwd: ASCII text
```

Figure 2.2: file command results

You can also identify the file type by looking at the output of the **"ls -l"** command. The **"ls -l"** command lists the contents of the specified file. The following command lists the contents of the current directory.

2.3.2 Linux File system structure

The Linux file system has a hierarchical file structure because it contains the root directory and its subdirectories. All other directories are accessible from the root directory. A partition usually has only one file system but can have multiple file systems.

File systems are designed to manage and provide storage space for persistent data storage. All filesystems require namespaces, which are a way of naming and organizing. A namespace defines the naming process, the length of a filename, or the subset of characters that can be used in a filename. It also defines the logical structure of files in a memory segment, e.g., Using directories to organize specific files. Once the namespace is described, a metadata description must be defined for that particular file.

The data structure must support a hierarchical directory structure; this struc-

ture is used to describe the free and used space for a given block. It also contains other details about the file, such as file size, creation date and time, update, and last modified time. It also holds extended information about parts of the hard disk, such as partitions and volumes. Extended data and the structures represent information about the file system stored on the drive; it is unique and independent of file system metadata.

The Linux File Hierarchy or File System Hierarchy Standard (**FHS**) defines the directory structure and directory contents in **Unix**-like operating systems. It is maintained by the Linux Foundation. In **FHS**, all files and directories appear under **root** / even if they are stored on different physical or virtual devices. Most of these directories exist on all **UNIX** operating systems and are generally used in the same way. Under these, we can find the representation of the directories in the system.

Root Directory – / –

Everything on your Linux system is located under the `/` directory. In another way, a Linux file system starts with a directory called root `-/-`. All files and directory files are created under this directory. Except for the root directory, each directory has a parent directory.

Bin Directory – /bin/ –

Contains basic user binaries (programs) that must exist when the system is mounted in single-user mode. Applications such as Firefox are stored in `/usr/bin`, while important system programs and utilities such as the bash shell are stored in `/bin`

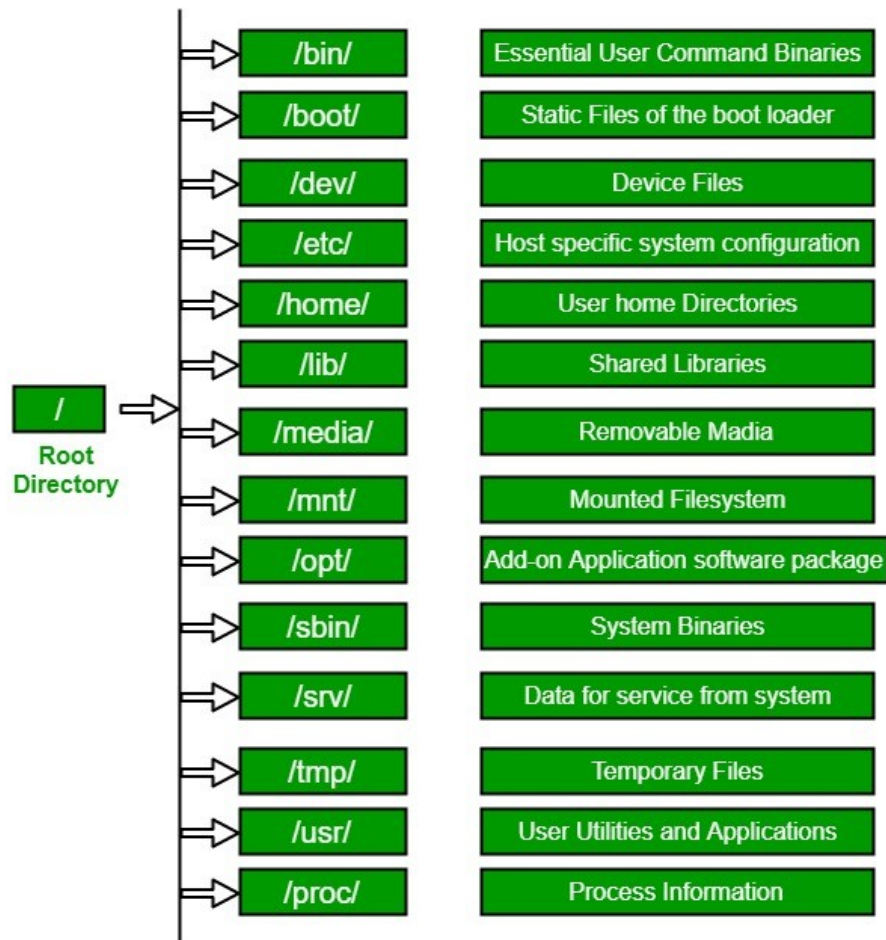


Figure 2.3: Linux File Hierarchy Structure [10]

Boot Directory – /boot/ –

Contains the files needed to boot the system. For example, this is where the **GRUB** boot loader files and the Linux kernel are stored. However, the boot-loader configuration files are not there; they are in `/etc` along with other configuration files.

Dev Directory – /dev/ –

Linux exposes devices as files, and the `/dev` directory contains a number of special files that represent devices. These aren't actually files as we know them, but

they appear as files. For example, `/dev/sda` represents the first **SATA** drive in the system. If you want to partition it, you can start the partition editor and tell it to edit `/dev/sda`. This directory also contains pseudo-devices, virtual devices that don't actually correspond to any hardware. In this directory, all the special files of the system are stored.

Etc Directory – `/etc/` –

Contains configuration files that can usually be edited manually in a text editor.

Home Directory – `/home/` –

Contains each user's home folder. For example, if your username is bob, your home folder is at `/home/bob`. This home folder contains the user's data files and user-specific configuration files. Each user only has to write access to their own home folder and must be elevated (became root) to modify other files on the system.

Lib Directory – `/lib/` –

Contains the libraries needed by the base binaries in the `/bin` and `/sbin` folders. Libraries required by binaries in the `/usr/bin` folder are in `/usr/lib`.

Media Directory – `/media/` –

Contains subdirectories for mounting removable media devices plugged into the computer.

Opt Directory – `/opt/`

Contains subdirectories for optional packages.

Proc Directory – `/proc/` –

It is similar to the `/dev` directory in that it does not contain any standard files. It contains special files that represent system and process information.

root Directory – `/root/` –

The `/root` directory is the home directory of the root user. It's not under `/home/root`, but under `/root`. This is not the same as the system root `/`.

Sbin Directory – `/sbin/` –

It is similar to the `/bin` directory. It contains important binaries that are usually designed to be run by the root user for system administration.

Usr Directory – `/usr/` –

The `/usr` directory contains applications and files used by the user, not the system. For example, non-essential applications are located in the `/usr/bin` directory instead of the `/bin` directory, and non-essential system management binaries are located in the `/usr/sbin` directory instead of the `/sbin` directory.

Var Directory – `/var/` –

The `/var` directory is a writable copy of the `/usr` directory and must be write-protected during normal operation. Log files and everything else that would normally be written to `/usr` during normal operation are written to the `/var` directory. For example, you can find log files in `/var/log`.

Those are the most important directories that build the structure of the file system. Some of these directories exist only on certain systems when certain

subsystems (such as the X Window System) are installed.

2.3.3 Linux File system Types

By default, after a fresh installation of the system, Linux proposes the choice of multiple file systems **Ext**, **Ext2**, **Ext3**, **Ext4**, **JFS**, **ReiserFS**, **XFS**, **btrfs**, and **swap**.

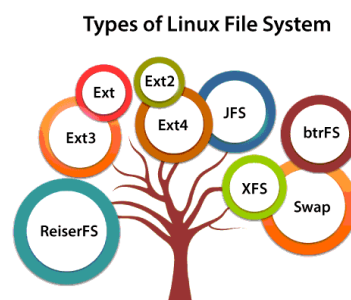


Figure 2.4: Linux file system structure[13]

2.3.4 Ext, Ext2, Ext3 and Ext4

The file system **Ext** stands for *Extended File System*, **Ext2** was the first Linux filesystem that allowed the management of 2 TB of data. **Ext3** was developed by Ext2; it is an updated version of Ext2, including backward compatibility. The main disadvantage of **Ext3** is that it does not support servers, as the file system does not support file recovery and disk snapshots. The **Ext4** filesystem is the fastest of all Ext filesystems. It is a very compatible option for **SSD** (*Solid State Drive*) hard drives and the default file system in Linux distributions.

JFS File System

JFS stands for *Journalled File System* and was developed by **IBM** for **AIX Unix**. It is a replacement for the **Ext** file system. It can also be used in place

of **Ext4** when stability with a small amount of resources is required. It's a handy filesystem when **CPU** power is limited.

ReiserFS File System

ReiserFS is a replacement for the **Ext3** file system. It has improved performance and advanced features. In the early days, ReiserFS was used as the default file system in **SUSE** Linux, but then some policies changed, so **SUSE** reverted to **Ext3**. The filesystem supports file extensions dynamically, but there are some performance disadvantages.

XFS File System

The **XFS** file system is considered a high-speed **JFS** designed for parallel **I/O** processing. **NASA** still uses this filesystem with their high storage servers (300+ TB servers).

Btrfs Filesystem

Btrfs stands for **B-tree** file system. It is used for fault tolerance, repairing systems, fun management, extensive memory configuration, etc. It is not suitable for production systems.

Swap File System

The paging file system is used by the Linux operating system to perform memory paging when the system sleeps. A system that never sleeps must have swap space equal to the size of its **RAM**.

2.4 Managing System Users

Adding and managing users is a common task for Linux system administrators. User account Maintain boundaries between the people who use your systems and the processes that use them run on your system. Groups are a way to assign assignable permissions to your system Simultaneously for multiple users.

2.4.1 The Identification The authentication

That process depends on two procedures: the first one is **identification** which is the operation of knowing the who in order to retrieve the permissions of the person. A user is defined by login, and the second one is **authentication** which consists of grabbing the proof of who we are. As an easy example, a user is authenticated by a password in a system.[?]

Everyone who uses your Linux system should have a separate user account. Having a user account Gives you an area to securely store your files and ways to customize your users. The user interface (GUI, paths, environment variables, etc.) adapts to the way you use your computer.

2.4.2 Users

A user is an entity in the Linux operating system that can manipulate files and perform other operations. Each user is assigned an **ID** that is unique to each user in the operating system. In this article, we will learn about users and the commands used to get user information. After installing the operating system, the root user has an ID of 0, and system users have IDs from 1 to 999 (both inclusive), so local user **IDs** start at 1000 and above (max $65535 = 2^{16} - 1$ [?]).

The user must be associated with at least one **GID** in addition to the **UID**. In the system, the user entity is defined by a set of attributes :

- a Name for the connexion
- a password
- an **UID**
- a **GID** referring to the main group of the user
- a description
- repository of connexion
- command of connexion

All those parameters can be updated on the `/etc/login.defs` file. Adding users to the system is always based on some rules concerning the format of the login listed on the **POSIX** standards.[?]

2.4.3 Groups

Linux systems can divide multiple users into many groups. These groups are collections of users who have the same permissions (such as read, write, or execute permissions) to a specific file or resource shared by users in the group. Linux allows you to add new or existing users to an existing group to take advantage of the group-specific permissions it grants. We will learn about different Linux groups and how to list all the members of that group. Linux is developed with two types of groups :

Primary or Login Group: It is a group associated with files created by a specific user. The name of this primary group is the same as the name

of the user who created this particular file. Each user must belong to exactly one group.

Secondary or Supplementary Group: You can use this type of group to grant permissions to a group of users who belong to the group. Users cannot be assigned to one or more subgroups.[\[19\]](#)

The `/etc/group` file contains the description of the groups in the system. Each line is defined by four fields:

Group: password: GID: user1: user2: ...

- 1 Name of the group
- 2 Password
- 3 GID
- 4 List of users

2.4.4 Passwords

Password management has become a hot topic over the past decade. A quick Google search revealed several options for a tool to choose a string that protects your personal information being shared. Some of these applications simply run on your computer and store your passwords offline in an encrypted format(**SHA-256**, **SHA-512**, **Blowfish**).

Additional features are richer, offering online synchronization with multiple devices, password sharing, two-factor authentication (2FA), and more. For some of these services, the simplicity of password management has been lost in the sea of features offered. Also, due to the convenience of online vaults that many of these services offer, you lose control of your data as your credentials

are synced to servers that you have no control over. [20]

The passwords are saved in the system. Two files are responsible for achieving the process starting by the `/etc/passwd`. A file that contains public information about the users of the local system, each line of the file represents a user, and it is composed of seven fields:

```
Login : password : UID : GID: comment: homedir: shell
```

- 1 User Name
- 2 Password: for the old version the crypted password. “x” the password is in the `/etc/shadow` file. “?” The account is locked.
- 3 UID
- 4 GID
- 5 Description for more informations
- 6 The Home folder to land in after a login
- 7 the default shell

For the second file `/etc/shadow`, it is the file responsible for stoking the encrypted passwords of the users. in this file, passwords are well defined; each line contains nine fields

```
alpha:$y$j9T$VzMkF3ud3QQnlQixq1/Dl0$IhMxSch0gu6Rv.FoVTfE/Vgmi4pOUZM2BfaiazWVIb1:19052:0:99999:7:::
```

- 1 Login
- 2 The encrypted password “\$y\$” indicates the encryption type password. For this field of encryption the most used values are the listed below:

\$1\$: MD5

\$2a\$: Blowfish

\$5 : SHA-256

\$ SHA-512

Other DES

- 3** The number says after the last modification from the 1st January 1970
- 4** Number of days of no changing "0" changing is permitted anytime
- 5** Number of days for the new change
- 6** Number of days before the expiration of the password
- 7** Number of days to deactivate the account after the expiration of the password
- 8** Number of days of the account deactivation starting from the 1st January 1970
- 9** Reserved

In some Linux distributions, not all of them, we can find an additional file `/etc/gshadow`. In this file the encrypted passwords of the groups are found.[?]

2.4.5 Users Authentication

Standard Authentication

Authentication is the official sysadmin term for logging into a system. This is the process by which users prove to the system that they are whom they say they are. This is usually done with a password but can also be done with

other methods such as fingerprints, PINs, etc. The process starts with the login screen. The user must have a valid account and know the correct password. When he enters his password, the system performs two checks:

Ensures the availability of the account (is not locked)

Confirms the entered password is correct by checking the `/etc/passwd` file.

If these two checks are completed successfully, he will be logged into the system. Another element that comes into play when logging in is the `etc/login.defs` file. The system refers to this file to determine password restriction parameters, in particular, how long passwords can be retained before needing to be reset.

PAM Authentication

The sensitive role of the passwords makes the game more attractive for no authorized users to break the encryption. For that, a module was made to make the game harder. **PAM**(*Pluggable Authentication Module*) is a set of modules that provides severe constraints for the creation and the modification of the passwords for a more robust authentication environment. [] PAM is mostly involved when you log into the system from the console or from the network using **SSH** or **Cockpit**. **PAM** separates the standard and specialized tasks of authentication from applications.

PAM is considered as a guard between standard, specialized tasks of authentication and the applications. Programs like login, gdm, ssh, ftp, etc., all want to know who the users are and whom they say they are, but there are many ways to do this. Users can provide username and password as credentials, which can be stored locally or remotely using **LDAP** or **Kerberos**. Users can also provide fingerprints or certificates as credentials. It would be tedious to require every application developer to override authentication checks for every

new method. Calling the PAM library leaves checking to the authentication expert. **PAM** is pluggable in that we can have different applications run different tests and modular in that we can add new methods with new libraries.

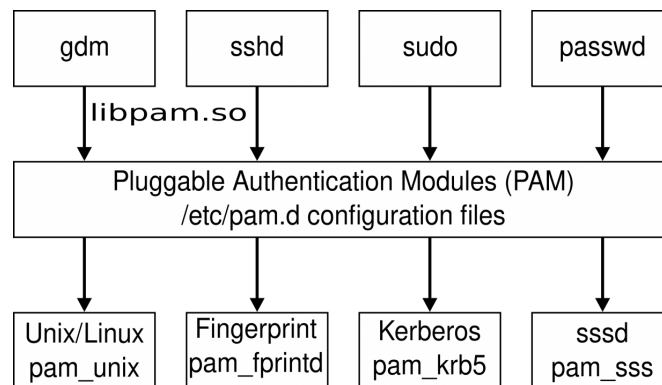


Figure 2.5: PAM Diagram [14]

From an application developer’s perspective, the information contained in the local configuration of the **PAM** library should not matter. In effect, applications are designed to treat the functionality documented here as a “*black box*” that handles all aspects of user authentication. “*All aspects*” include user authentication, account management, session initialization/termination, and password (authentication token) reset

Permissions management

16

When managing Linux permissions, administrators should be able to see at a glance object types and permissions assigned to user owners, groups, and others.

After the first character in the first column, we have nine characters that define all required permissions for a given object. The first three characters of

```
[root@cerberus ap6]#
[root@cerberus ap6]# pwd
/ap6
[root@cerberus ap6]#
[root@cerberus ap6]# ls -la
total 0
drwxr-xr-x. 5 root root 82 Feb 9 22:42 .
dr-xr-xr-x. 18 root root 235 Feb 9 22:33 ..
drwxr-xr-x. 2 root root 6 Feb 9 22:41 devops
-rw-r--r--. 1 root root 0 Feb 9 22:42 index.html
drwxr-xr-x. 2 root root 6 Feb 9 22:41 it
drwxr-xr-x. 2 root root 6 Feb 9 22:41 marketing
-rw-r--r--. 1 root root 0 Feb 9 22:42 README.md
[root@cerberus ap6]#
```

Figure 2.6: Permissions List

the group are associated with the user owner, the other three are associated with the group owner, and the remaining three are associated with others.

Permissions are presented in two ways. We can use letters, which are called symbols, and they are represented by r (read), w (write), and execute (x). They can also be represented numerically, where they are read as four, written as two, and executed as one, a representation called octal, both work.

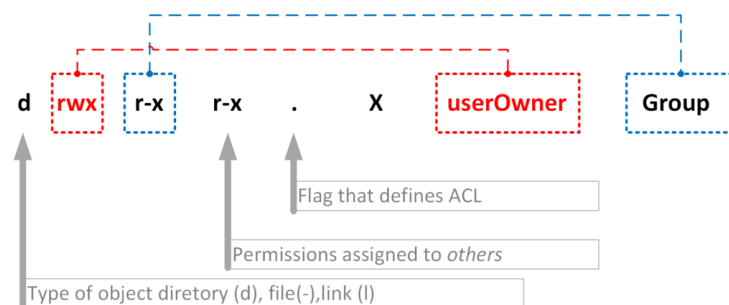


Figure 2.7: Portions of the permissions

The permissions management is based on the ”**chmod**” which means ”*change mode*” command using either symbolic or octal methods. The syntax is the following:

```
chmod permissions resource-name
```

Access level	Octal value
Read	4
Write	2
Execute	1

Table 2.1: Absolute mode guide

Identity	Position
User	First or left-most
Group	Middle
Others	Last or right-most

Table 2.2: Absolute mode references

To get the idea in a better way, here are two examples of manipulating permissions for a file named *"txt_file"*:

```
chmod 740 txt_file
```

```
chmod u=rwx,g=r,o-rwx txt_file
```

In the last two examples, some numbers and characters appeared, and the two commands appear different, but in reality, they end with the same goal, and to get the difference, we need to present the two modes the *"absolute mode"* and the *"symbolic mode"*

Starting with the first mode listed, the *"absolute mode"* is one of the two methods of defining permissions, and it is referred to it as the octal mode of the numeric mode. in this type, each access level (read, write, execute) has a specific octal value, as it is listed in the following table:

Each identity (user, group, others) has a position:

The three permissions values are associated with identities in the following way:

UGO
740

Moving to the second mode, the *Symbolic mode* is known more as the friendly mode or the kiddy mode, which uses more symbols, and the symbols are easier to understand. In the three following tables, the guides to use this method is listed.

For the Access level, each one has a referring symbol:

Access level	Symbol
Read	r
Write	w
Execute	x

Table 2.3: Symbolic mode guide

For the identity is set in a clear way as bellow:

Identity	Symbol
User	u
Group	g
Others	o

Table 2.4: Symbolic mode identity guide

Finishing by the relationship between the operations and the tasks:

Task	Operator
Grant a level of access	+
Remove a level of access	-
Set a level of access	=

Table 2.5: Symbolic mode relationship

Since Linux is a multi-user system, permissions and ownership are used for security reasons. There are three types of users on a Linux system Users, groups, and others. Linux divides file permissions to read, write, and execute, denoted by r, w, and x, respectively. A file's permissions can be changed using the "*chmod*" command, which can be further broken down into absolute and symbolic modes, and the "*chmod*" command can change the ownership of a file/directory. Use the following commands: *chown* user file or *chown* user:group file The *chgrp* command can change What does group ownership *chgrp* group filename x mean - run directory? Permission to "go into" the directory and possible access to subdirectories.

2.5 Conclusion

The job of a Linux systems administrator is to manage the operations of a computer system like maintain, enhance, create user account/report, taking backups using Linux tools and command-line interface tools. Most computing devices are powered by Linux because of its high stability, high security, and open-source environment for this work Linux is a major strength in computing technology. Most of the mobile phones, personal computers, supercomputers, and cloud-servers, webserver are powered by Linux. The general work of Linux in servers can be found in the following section.

Chapter 3

Linux Server Administration

3.1 Introduction

Managing a Linux server is different from managing a Linux workstation, and managing a Linux server is very different from running a desktop operating system like Windows or Mac OS X. To start in Linux server management, the first point to understand is that the available Linux distributions to serve the workstation or desktop uses are more different than the Linux servers. The Linux servers' editions are more powerful; they are fabricated to handle high demands.

Linux Server includes additional features to simplify network management. These management tools include advanced system administration functions and the ability to manage databases. Your Linux server version can also run advanced web applications and other services. A high level of security and solid stability while maintaining a high degree of flexibility are the benefits of choosing a Linux server. As the name suggests, servers exist to serve. The data they provide can include web pages, files, database information, emails, and many other types of content. As a server administrator, some additional challenges to your system administration skills are as follows:

Remote access To use a desktop system, you usually sit at its console. The server system, By contrast, tends to be locked in air-conditioned environments. In most cases, most of the administrative work is done once the physical computer is in place. Some of these machines are done using remote access tools. Usually, no GUI is available, so you have to rely on command-line tools for things like remote logins, Remote Copy and Remote Execution. The most common of these tools are based on *Secure Shell (SSH)* settings.

Diligent security To be useful, the server must be able to accept requests content from remote users and systems. Not unlike desktop systems, Closing all network ports that allow incoming access requests to the server Must make itself vulnerable by allowing some access to its ports.

Continuous monitoring You usually turn off your laptop or desktop. When you're not using it, the system typically keeps the server running 24 hours a day, 365 days a year. Because you don't want to sit next to each server and monitor them yourself all the time, you can configure tools to monitor each server, collect log messages, etc. you can Enable System Activity Reporter to collect 24/7 CPU usage data, Storage usage, network activity, and disk access.

The subdomains and tools helping to do the work will be discussed clearly in the next sections to get how things are done clearly in the administration.

3.2 Managing Services

Services are programs or processes that run continuously on your server, usually from when the server starts. They continuously support requests and monitoring from other processes or external clients. In another way, services are basic processes that typically run in the background, not under the direct control of an interactive user, waiting for requests from other software programs, or performing basic tasks at the right time. Many services are implemented as daemons.

On Linux, a daemon is a program that runs as a background process. Traditionally, the process name of a daemon process ends with the letter "d" to make it clear that the process is a daemon process and to distinguish it from a normal computer program. For example, "syslogd" is a daemon that implements "syslog" functionality, and "sshd" is a daemon that serves incoming "SSH" connections.

3.2.1 Init & Systemd

All services work with multiple scripts, which are stored in the "/etc/init.d" directory, this *init.d* is a daemon process which is the first process of a Linux system. Then other processes, services, daemons, and threads are started by init. So *init.d* is the configuration database for the init process. Now let's check some daemon scripts by printing some processes. Daemon scripts contain start, stop, status, and restart functions. Let's take ssh as an example. **Systemd** is generally considered more powerful than traditional init systems.[\[21\]](#)

In recent years with all the new development, Linux distributions have increasingly switched from other init systems to "*systemd*". The **systemd** tool suite provides a fast and flexible initialization model for managing an entire

machine from boot. However, before getting on it, the term "Units" needs to be clarified in the case of this newly adopted solution.

3.2.2 Units

A Unit is any resource the system can process and manage and is the main object the systemd tools can handle. These resources are defined using configuration files called unit files. It is similar to a service or job in other init systems. However, a unit has a broader definition as it can be used to abstract services, network resources, devices, filesystem mounts, and isolated resource pools. Ideas that can be handled in other init systems with unified service definitions can be broken down into component units according to their concerns which are organized by function, allowing you to easily enable, disable or add functionality without changing the core behavior of the unit. The basic objects that systemd manages and responds to are "*Units*". Units can be of many types, but the most common type is a "service" (represented by a unit file ending in `.service`). Some of the functions that the unit can easily implement are **socket-based activation, bus-based activation, path-based activation, device-based activation, implicit dependency mapping, instances and templates, easy security hardening, drop-ins, and snippets**[\[22\]](#)

Systemd classifies Units according to the type of resource they describe. The easiest way to determine the unit type is to look at its type suffix, which is appended to the end of the resource name. The following list describes the unit types available to systemd

- .service:** A service unit describes how a service or application is managed on the server. It includes how the service is started or stopped when

you would like it to start automatically and dependencies and ordering information for related software.

- .socket:** A socket unit file describes the network, IPC sockets, or FIFO buffers that systemd uses for socket-based activation. These always have an associated `.service` file that is started when activity is seen on the socket defined by that unit.
- .device:** An Unit describing a device designated by a `udev` or `sysfs` filesystem as managed by systemd. Not all devices have `.device` files. Some scenarios that might require a `.device` unit are ordering, assembling, and accessing devices.
- .mount:** This unit defines the mount points on the system to be managed by systemd. They are named after the mount path and change slashes to hyphens. Units can be automatically created for entries in **`/etc/fstab`**.
- .automount:** The `.automount` unit configures the mount point for automounting. These must be named after the mount point they refer to and must have the appropriate `.mount` unit to define the details of the mount point.
- .swap:** This unit describes the swap space on the system. The names of these units must reflect the scoped device or file path.
- .target:** Target devices provide synchronization points for other devices when they power up or change state. They can also be used to bring the system into a new state. Other units specify their relationship to the target to bind to its action.
- .path:** This unit defines the paths available for path-based activation. By

default, when the path reaches the specified state, a `.service` unit with the same base name is started, and it uses `inotify` to monitor paths for changes.

- .timer:** The `.timer` unit defines a timer managed by `systemd`, similar to a delayed or scheduled activation of a cron job. The matchmaking session will start when the timer is reached.
- .snapshot:** snapshot units are automatically created by the `systemctl` `snapshot` command. It allows you to rebuild the current state of the system after changes. Snapshots do not exist in the session and are used to reset temporary state.
- .slice:** A `.slice` unit is associated with a Linux control group node, so resources can be limited or allocated to any process associated with a slice. The name reflects its hierarchical position in the cgroup tree. By default, cells are placed in specific segments based on their type.
- .scope:** Scope units are automatically created by `systemd` based on information received from its bus interface. These are used to manage the set of externally created system processes.

Units Management

Files that define how `systemd` handles units can be found in many places, each with different priorities and meanings. System copies of unit files are typically stored in the `"/lib/systemd/system"` directory. When installing software unit files on a system, this is where they are placed by default. Unit files stored here can be started and stopped as needed during a session. Files in this current directory should not be modified. You should not edit files in this directory.

Instead, consider overwriting the file with a different unit file Location that replaces the file at that place.

If you want to change a unit's work, the best place to go is the `"/etc/systemd/systemd"` directory. Unit files found in this directory take precedence over any other in the file system. Placing a replacement file in this directory is the safest and most flexible method if you need to change the system copy of a unit file. If you only want to override specific directives in system unit files, you can provide unit file snippets in subdirectories. These will append or modify system replication instructions, so you can only specify the options you want to change.

To manage services on a systemd-enabled server, our primary tool is the `"systemctl"` command. Using this command, a service can be started, stopped, restarted, reloaded, enabled, and disabled. By default, most systemd unit files are not started automatically at boot. This tool must be executed with a `"root"` privilege, and the syntax of the command is the following:

```
sudo systemctl option name-of the service
```

This command is also usable to check the system's status, view basic Log information, and stop and reboot the server.

```
systemctl list-units
systemctl list-units --all
systemctl list-unit-files
journalctl
```

By default, this will show you entries from the current and previous boots if the journal is configured to save previous boot records. Some distributions

enable this by default, while others do not (to enable this, either edit the `/etc/systemd/journald.conf` file and set the `Storage=` option to "persistent", or create the persistent directory by typing `sudo mkdir -p /var/log/journal`).

3.3 Network Management

3.3.1 Desktop

NetworkManager runs on desktop systems. Network interfaces are managed by default. With NetworkManager, you can automate. Usually accepts address and server information needed to connect to the Internet. However, you can also set the address information manually. You can configure a proxy server or a virtual private network connection that can enable your Desktops to be behind an organization's firewall or through a firewall. Whether you connect to the Internet from Linux, Windows, smartphone, or any other device, The type of device that supports the network, this connection must have certain conditions to work. The computer must have a network interface (wired or wireless), IP address, Assigned DNS servers, and routes to the Internet (identified by the gateway device).

In the desktop case, the graphical and command-line tools are available for checking about the information of the connection. In this document, the interest will be on the command-line tool; It provides more detailed information about the network interfaces. The main commands used in this context are

```
ip addr show ip a
```

Another common command that displays network interface information is the **ifconfig** command. By default, **ifconfig** displays similar information to **ip addr**, but also **ifconfig** Displays the number of received (RX) and transmitted (TX) packets and the number of packets data and any erroneous or dropped packets

if config wlan0 results

Whether you use Network Manager or the command line to change network configuration, most Update the same configuration file to set up routes in files in the `/etc/sysconfig/network-scripts` directory. Other network settings are stored in other files in the `/etc` directory of the host file. If you edit these files directly, consider disabling the network manager service and enabling the network service due to the Network Manager sometimes overwriting files you manually configured.

3.3.2 Enterprise

So for the last one, it describes setting up a single system to connect to a network. Linux offers more than that. A Linux configuration can handle and manage all the hosting systems connected. A general overview of the infrastructure will be presented in this section.

Starting by configuring Linux as a router, If your computer has multiple network interfaces (usually two or more NICs), You can configure Linux as a router. All it takes to get there is to change the Kernel parameter to enable packet forwarding. To turn on the `ip_forward` parameter, Immediately and temporarily type the following as root:

```
cat /proc/sys/net/ipv4/ip_forward
0
echo 1 /proc/sys/net/ipv4/ip_forward
cat /proc/sys/net/ipv4/ip_forward
```

Packet forwarding (routing) is disabled by default, and the value of `ip_forward` is set to 0. Setting it to 1 enables packet forwarding immediately. Make this change permanent. Finally, you need to add this value to the `/etc/sysctl.conf` file so that it looks like this:

```
net.ipv4.ip_forward = 1
```

When a Linux system is used as a router, it is often used as a firewall between private systems Networks and public networks, such as the Internet. If so, using the same system as the firewall that performs network address translation (NAT) is also desirable. and provides DHCP services so that systems on the private network can use Linux systems with private IP addresses.

The other infrastructure configuration consists of configuring the DHCP system. A Linux system cannot just use a DHCP server to obtain its IP address and other information. It can also be configured to act as a DHCP server itself. In its simplest form, a DHCP server can distribute IP addresses from the address pool to any system that requests them. Usually, however, the DHCP server also distributes the DNS servers and default gateways. Configuring a DHCP server should not be done lightly. Don't add a DHCP server on a network you don't control but already own A working DHCP server. Many clients are set

up to receive address information from any DHCP The server that distributes it.

The primary configuration file is `/etc/dhcp/dhcpd.conf` for IPv4 networks (there is a `dhcpd6.conf` file in the same directory to provide DHCP service for IPv6 networks). By default, the `dhcp` daemon listens on UDP port 67, so remember to keep that port open on your firewall. To configure a DHCP server, you could copy the `dhcpd.conf.sample` file from the `/usr/share/doc/dhcp-4*` directory and replace the `/etc/dhcp/dhcpd.conf` file. Then modify it as you like

Another configuration for this type is Configuring the **DNS** server. Most professional **DNS** servers (Domain Name System) are implemented under Linux Berkeley Internet Domain Name (BIND) Service. Hostname to IP Address mapping is done in zone files located in the `"/var/named"` directory. When you install, Bind the `chroot` package and move the binding configuration file to `"/var/named/"` `chroot` directory, trying to copy files from `/etc`, and `/var` Configure bind is required to configure the `named` daemon (provide services). Stored in the directory structure `"/etc/named/chroot"`.

3.3.3 Remote Access

A remote desktop, according to Wikipedia, is "a software or operating system feature that allows a personal computer's desktop environment to be run remotely on one system (usually a PC, but the concept applies equally to a server) while being displayed on a separate client device." Remote access can also be interpreted as the remote control of a computer from other devices connected via the Internet or other networks. This is commonly used by many

computer manufacturers and the help desks of large corporations for technical troubleshooting of customer issues. To achieve this process, services are available known as the telnet and the **SSH**.

Telnet

Telnet is a client-server protocol for character-based data exchange over a TCP connection. Telnet allows remote control of computers through text-based input and output. For this, a client-server connection is established by default using the TCP protocol and TCP port 23, and the remotely controlled device acts as a server and waits for commands. The telnet client, the controlling entity in this process, also known as *remote access* or *remote login* can be installed on either special devices or ordinary computers. However, according to different terminal devices, the presentation of the transmitted data information will be different. Protocols of the TCP/IP protocol suite can also be used to control applications without a graphical user interface.

Telnet is required whenever you need to connect to another computer or network component. Everything is done via a text-based command line. This has been especially useful in the past for shared mainframe services. But even today, telnet is still used, albeit less and less, to manage networks, run applications, and share databases. Nowadays, with multiple databases, telnet has also played a vital role in organizations using large databases over the years. For example, library protocols were a fundamental part of online catalogs published in the 1980s, better known as **OPAC** (Online Public Access Catalog). These digital publication databases will initially be accessible through the library's local network terminals. With the increasing success of the Internet, it can also be accessed through a locally available web interface, whose communication is

usually supported by the Telnet protocol.

For Interaction with programs on application servers, a typical use case for a Telnet client is to access text-based programs on an application server. For example, free Internet chess servers are still available today through a Telnet connection. You can choose from available opponents or move your pieces around the board by entering text. At the same time, graphical interfaces such as Jin Applet (Jin is an open-source, cross-platform, graphical client for chess servers, written in Java.) or Javaboard, which can move game pieces with the mouse, have replaced text-based input.

For the main use we need in our case, which consists of administering networks and servers, The telnet got the lion's share before the SSH protocol. Telnet has long been a convenient protocol for network and server administrators. The ability to remotely manage devices in a network is great for administrative tasks, especially since nearly all devices support the protocol. It can also be used to check the availability of a specific port or to detect errors on the email server (SMTP, port 25) by sending an email directly from the server [24]. Telnet solutions are an efficient way to configure servers, such as web servers. Change directory structures, file access permissions, or passwords quickly and easily.

For Telnet connections being standard TCP connections, clients can be used for testing or other services that use TCP as the transport protocol. For example, with a simple request, you can check the functionality of an HTTP server or (as mentioned) the status of an email server. This versatility is complemented by the fact that the connection protocol can be used across platforms. Only a few devices do not support official IETF standards. It also doesn't matter whether the client and server machines are based on the same operating system.

Another benefit of telnet is that, if authorized, it allows unrestricted access to the resources of the controlled system. However, telnet presents a high-security risk: when using the Telnet protocol, neither connection establishment nor data transmission is encrypted. Therefore, any information you send may be intercepted in plain text by third parties, including credentials required for remote access. This means that hackers won't have much trouble taking over the system. A secure alternative to Telnet is *Secure Shell (SSH)*.

Advantages	Disadvantages
Telnet client is versatile	Unencrypted data exchange
It can be used cross-platform	Full access makes it easier for hackers
Unlimited access to target resources	Only few servers can be reached via Telnet

Table 3.1: Telnet Advantages Disadvantages

SSH (Secure Shell)

This protocol was released after the telnet to complete the lacks of the previous protocol. The ssh command provides a secure encrypted connection between two hosts over an insecure network. This connection can also be used for terminal access, file transfer, and tunneling other applications. Graphical X11 applications (The X Window System (X11, or simply X) is a windowing system for bitmap displays, common on Unix-like operating systems.[23]) can also be run securely from remote Locations via **SSH**. Almost every Unix and Linux system includes the ssh command. This command is used to start an **SSH** client program that allows a secure connection to an SSH server on a remote computer. The ssh command is used for logging into a remote computer, transferring files between two computers, and running commands on the remote computer. In our case of research, the **SSH** wins the use, and all the focus will be on it.

Linux typically uses the **OpenSSH** client, which is an open-source imple-

mentation of the SSH protocol. It is based on the free version by Tatu Ylonen and further developed by the OpenBSD team and the user community. The `ssh` command to log into a remote machine is very simple [26]. To log in to a remote computer named `sample.ssh.com`, enter the following command at a shell prompt:

For the use cases of the SSH, it replaced the telnet, so it is the same already discussed in the previous section of telnet, and in order to know how to use this command, we will first get to know the building blocks of it. and to start, we need to know about the client-side.

The OpenSSH client program is called `ssh`. SSH clients typically use information from the `.ssh` directory in the user's home directory. It also reads `/etc/ssh/ssh_config`, which contains its system-wide configuration. Moving to the OpenSSH server program, it is called `sshd`. As already discussed in the services section, The server is usually started during boot, and it is considered as a daemon service running in the background of the system and reads its configuration from the `/etc/ssh` directory. Its main configuration file is usually `/etc/ssh/sshd_config`.

The SSH provides an encryption key-based authentication mechanism called public-key authentication. One or more public keys can be configured as authorization keys; the private key corresponding to the authorization key is used for authentication to the server. Typically, both the authorization key and the private key are stored in the `.ssh` directory of the user's home directory. Basically, such keys are like fancy passwords, except that the password cannot be stolen from the network, and the private key can be encrypted locally. The SSH is

based on the SSL protocol, which uses asymmetric cryptography to exchange the key(ensures the data that is transferred between a client and a server remains private). SSH keys provide the same level of access as a username and password and are typically used for privileged accounts that have access to the operating system. The number of SSH keys is usually ten times the number of passwords. To get it clear the explanation of the encryption, SSH uses three data encryption types during the communication between the machines:

symmetric encryption starting with symmetric encryption, it generates a single key exchanged by the two machines. The machines then use the key for encryption and decryption. This method is fast, takes no resources, and SSH uses it for every session. Whenever a client and server negotiate an algorithm to use for an SSH session, they always choose the first algorithm in the list of clients supported by the server. Symmetric encryption is often referred to as **shared key** or **shared secret encryption**. Usually, only one key is used, or sometimes a pair of keys, one of which can be easily computed with the other. Symmetric keys are used to encrypt all communications during an SSH session. Both client and server use an agreed method to derive the key, and the generated key is never shared with anyone.

asymmetric encryption Discussing Asymmetric encryption, data is asymmetrically encrypted when the machine uses two different but mathematically related keys (public and private keys) to perform encryption. Client computers involved in setting up encryption can use the private key to decrypt the information. Contrary to popular belief, asymmetric encryption is not used to encrypt the entire SSH session. Instead, it is used during a symmetric encryption key exchange algorithm. Before initiating a secure connection, both parties generate an ephemeral pub-

lic/private key pair and share their respective private keys to create a shared secret. Once a secure symmetric communication is established, the server generates it using the client's public key, queries it, and transmits it to the client for authentication. If the client can successfully decrypt the message, it has the private key needed to connect - the SSH session begins.

hashing One-way hashing is another form of encryption used in Secure Shell connections. One-way hash functions differ from the above two forms of encryption in that they are never decrypted. They generate a unique, fixed-length value for each input with no clear trend that can be exploited. This makes them almost irreversible. Generating a cryptographic hash from a given input is easy, but generating input from a hash is impossible. This means that if the client has the correct input, it can generate a cryptographic hash and compare its values to verify that it has the correct input. SSH uses hashes to verify the authenticity of messages. This is done using HMAC or hash-based message authentication codes. This ensures that received commands cannot be tampered with in any way.[\[26\]](#)

How It works The way SSH works are that it uses a client-server model to allow authentication of two remote systems and encryption of data transmitted between them. SSH works by default on TCP port 22, and the host (server) listens on port 22 for incoming connections. It organizes secure connections by authenticating clients and opens the correct shell environment upon successful authentication. The client must initiate the SSH connection by initiating a TCP handshake with the server, ensuring a secure symmetric connection, verifying that the identity provided by the server matches a previous record (usually recorded in an

RSA Keystore file), and verifying the required user credentials. There are two steps to establishing a connection - first, the two systems must agree on an encryption standard to protect future communications, and second, the user must authenticate themselves. If the credentials match, the user is granted access.

Now after knowing how the system works in a general view, discussing session encryption negotiation is a must. When a client tries to connect to a server over TCP, the server provides the encryption protocol and the specific version it supports. If the client has a similar matching protocol and version pair, then an agreement is reached, and the connection is started using the accepted protocol. The server also uses an asymmetric public key, which clients can use to verify the authenticity of the host. Once set up, both parties use the so-called Diffie-Hellman key exchange algorithm to create a symmetric key. This algorithm allows both client and server to obtain a common encryption key that will be used to encrypt the entire communication session from now on. The algorithm works in the way listed below:

- 1 Both client and server agree on a very large prime number, which of course, has no common factor. This prime value is also called the starting value.
- 2 Next, the two parties agree on a common encryption mechanism that manipulates the seed value in a specific algorithmic way to generate another set of values. These mechanisms, also known as crypto generators, perform a wide range of operations on the seed. An example of such a generator is **AES**

- 3 Both parties independently generate another prime number. This is used as the secret private key for the Interaction.
- 4 This newly generated private key with a shared number and encryption algorithm (such as AES) is used to calculate the public key that is distributed to another computer.
- 5 The parties then use their personal private key, the other machine's shared public key, and the original prime number to create the final shared key. This key is computed independently by the two computers, but the same encryption key is generated on both sides.
- 6 Since both parties now have a shared key, they can symmetrically encrypt the entire SSH session. The same key can be used to encrypt and decrypt messages.

Nonetheless, SSH keys are ignored in most identity and access management projects. They require the same type of configuration and termination procedures and audit attention as passwords or other authentication methods.

3.4 Virtual Servers

A virtual server has the same functionality as a physical server but lacks the underlying physical mechanisms. A physical server can use virtualization's hypervisor or container engine to create multiple separate virtual servers, and the instances share physical server resources such as CPU and memory. Virtualization requires an abstraction layer between server hardware and software to create multiple virtual instances on a single physical server. On modern computers and servers with hypervisors or container engines, this can be achieved with just a few clicks. For medium to large organizations, administrators can

implement virtualization strategically to optimize space and performance requirements. Broadly speaking, with collocation and private data centers, virtual servers are available to everyone in remote Locations. Network administrators can remotely control the functionality of virtual servers without physical access to the host server.

3.4.1 Types of Virtual Servers

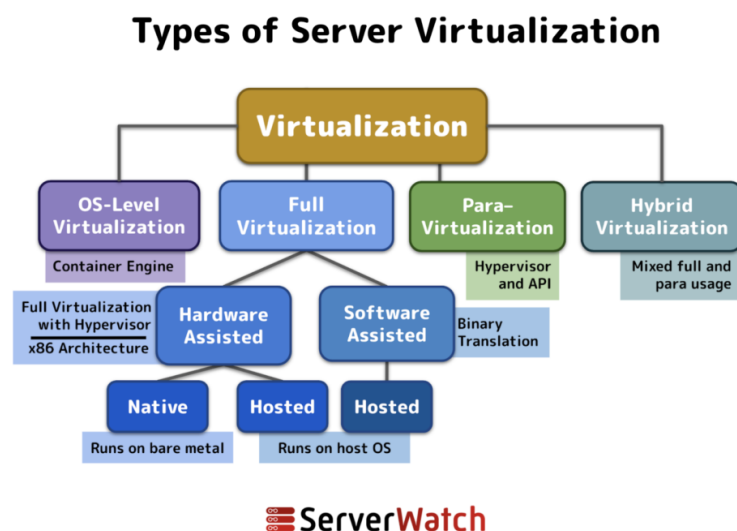


Figure 3.1: Types of server virtualization [25]

All virtualization methods can help companies optimize the availability and agility of physical servers. These approaches differ in the resources and goals of the network to be virtualized.

Full Virtualization

Full virtualization uses a hypervisor to intercept and simulate virtual servers. The *software-assisted* method implements the hypervisor using a directly executed binary translation (BT). Hardware-assisted virtualization can be imple-

mented with current x86 processors, known as bare metal (hypervisor type 1) or a hosted method on the operating system (hypervisor type 2).

OS-Level Virtualization

Operating system-level virtualization is the latest approach in this field due to the virtualization capabilities embedded in modern operating systems. Like paravirtualization, virtual servers do not emulate the host's hardware at the operating system level. With appropriate software, the operating system core creates separate lightweight instances, so-called containers. Virtualization and containerization are slightly different processes.

Para-Virtualization

Paravirtualization also uses a hypervisor, but virtual servers cannot fully emulate the hardware of a physical host. Instead, APIs (often built into modern servers) directly exchange calls to the host and virtual server operating systems. The resulting virtual server sees its environment as an extension of the host and adjacent virtual server resources.

Para-virtualization vs. Full Virtualization

In general, paravirtualization is safer and faster than full virtualization. By communicating directly with the hypervisor through APIs and drivers, paravirtualization is known for better performance. When considering virtual server migration, full virtualization has the advantages of portability and compatibility.

3.4.2 Advantages disadvantages

Starting with the positive side of this solution. It Reduces costs by reducing power, cooling, and overhead costs. Space optimization by compressing traditional physical servers into virtual servers. Increased scalability as administrators can create new virtual servers as needed. Backup and restore capabilities enable fast and reliable recovery. Setup, configuration, and maintenance technical support from the virtualization vendor. Improve workload efficiency and load balancing for network demands. Easily deploy new updates and software to a set of virtual servers.

In having the positives, the negatives also exist, and they can be listed in this way: Technical management for creating, configuring, monitoring, and securing virtual instances. Performance lags when the host's virtual server is at a high activity level. The upfront cost of purchasing a physical host and licensing virtualization software. Compared to cloud platforms, scalability is poor. Older applications may not be compatible with virtualization. Limited disk space is usually limited to a single virtual machine or multiple containers. Less control than managing an internal server fleet; tied to vendor SLAs.

3.4.3 Virtual Server vs. Virtual Machine (VM)

Virtual servers and virtual machines (VMs) are often combined, but there are differences. While a virtual server can be accessed in several ways, a virtual machine is a virtual server that uses full virtualization. Likewise, a container is a virtual server that uses OS-level virtualization.

3.5 Conclusion

The open source movement has grown to become a significant force in today's computing environment. In some sectors of the software industry, open source programs have become popular enough to provide real competition to proprietary alternatives; in others, they have emerged as the dominant standard. Linux is becoming increasingly popular as the operating system for servers, gradually eating away at the market share of Windows NT. Apache has long been, and will remain in the foreseeable future, the most popular web server program. And for those developing UNIX-compatible software, the GNU development tools have become almost universally standard. In addition of all this the security The Hardening point in this field is the major thing to be discussed in the following section.

Chapter 4

Linux System Security & Hardening

4.1 Introduction

As Linux users, we have some inherent advantages over our Windows counterparts regarding security (or lack thereof). Like gamblers, hackers use the laws of probability and averages to find vulnerable computer systems to hack into. They typically target the types of systems with the most security holes. They'll also focus primarily on areas where unprotected systems have the greatest opportunity, such as the most prevalent types of systems on the Internet. In both cases, Windows is the primary operating system. The security community says that Windows has a larger attack surface in terms of vulnerabilities and the number of systems.

Linux is more secure and rarer than Windows-based systems, which means that attacks on Linux systems are less common than Windows systems. However, it would be foolish to be complacent about protecting every system, whether running Windows, Linux, or any other operating system.

Security should be one of the most important considerations in all stages

of setting up a Linux computer. Implementing a good security policy on a computer requires a good understanding of the basics of Linux and some of the applications and protocols used. Linux security is a huge topic, and many complete books are on the subject. Although Linux users are less susceptible to viruses than some other major operating systems, Linux users and administrators still face many security concerns. One of the most important steps in any task is figuring out why you're doing it. Not only are you saying that we need to make the system secure, but you also need to consider the implications of security, the risks associated with available data, and the impact your security measures will have on your users. How do you know if you've achieved your goal of making the system secure without first considering any of these factors?

4.2 Type of Attacks against the system

There are many different types of attacks that take place. These may vary depending on the services you offer or the type of attackers targeting you. These are areas that will be considered later for establishing protection methods. This list is not exhaustive but can give you an idea of which areas to focus on. New methods are still emerging, and security administrators need to ensure they don't become obsolete.

4.2.1 Reading data

Computer systems commonly associated with espionage or theft often contain information that must be kept secret or secure. This can range from emails about project quotes to personal information or bank details. Disclosure of this information could cause serious damage to the company or result in legal action. Therefore, the legal principles state that personal data "must have appropriate

security measures”.

4.2.2 Changing data

Possibly more serious, the attack could gain enough access to update the data. This could be an act of vandalism that destroys the organization or leaves a business card. One of the biggest risks is that data could be altered and ignored. The field is particularly well-known cases of attackers replacing websites with modified versions.

4.2.3 Denial of service

In a denial of service (DoS) attack, an attacker disables or makes a service provided by the system unavailable. The previous DoS attack was ”Ping of Death”. Creating an ICMP echo command larger than the maximum allowed size can cause the computer to fail. The vulnerability was discovered on Windows Vista and was originally fixed in earlier versions of Windows ten years later (<http://www.v3.co.uk/v3/news/2249151/ancient-flaw-hits-vista>). Many past DoS attacks have been addressed with bug fixes. However, a more problematic threat is also known as distributed denial of service. The first known examples were attacks on Altavista and Yahoo in early 2000. However, similar attacks were launched in 2009 against various websites, including Twitter, and more recently against websites responsible for filtering Pirate Bay and other file-sharing sites. A distributed denial of service attack works by having an attacker or would-be attacker install a Trojan horse on many different computers. When these Trojans are triggered simultaneously, they directly attack a single system. The combined effect of thousands of simultaneous attacks prevents the system from functioning. This attack is becoming more sophisticated, and

security administrators are devoting more resources to solving these types of problems.

4.2.4 Access to computer

While some systems allow other users to access your system, these user accounts can sometimes be compromised. Computers may not contain confidential material, and users may be unable to write data, but they may use your system to cause harm. For example, if someone manages to attack a computer between a secure network and an unsecured network, they can use your computer as a way to switch between the two networks. Another technique for using your computer to attack another computer is a distributed denial-of-service attack. When triggered, an attacker can inject a Trojan horse on your computer to attack another computer. It can be embarrassing if someone finds out that your organization's systems are being used to commit any of these crimes. It may even appear that someone in your organization committed the crime.

4.3 System Hardening

System hardening is a process to secure a computer system or server by eliminating the risks of cyberattacks. The process involves removing or disabling system applications, user accounts, and other features that cyber attackers can infiltrate to gain access to your network. These features, sometimes known as the attack surface, often serve as the entry points for malicious cyber activities or hackers.[\[27\]](#)

System hardening is important because the attack surface of a corporate or personal network is one of the most vulnerable Locations for cyberattacks. At-

tack surface entry points can enable hackers, malware, and other cyber threats to access an organization's sensitive information. Through system hardening, organizations can reduce their vulnerability to cyber threats and the likelihood that cyber threats will gain access to their networks.

4.3.1 System Hardening Types

System hardening protects not only a computer's software applications (including the operating system) but also its firmware, databases, networks, and other critical elements of a given computer system that attackers can exploit.

- Server hardening
- Software application hardening
- Operating system hardening
- Database hardening
- Network hardening

Server hardening

Server hardening refers to protecting servers' ports, features, data, and permissions. A server is a powerful computer that provides resources, services, or data storage to other devices on an authorized network. Server hardening techniques include disabling USB ports when the system is turned on, regularly updating or patching server software, and creating stronger passwords for all users who have access to the server.

Software application hardening

With software application hardening, users or organizations can add or update security measures for all programs and applications on their network. These applications may include web browsers, word processors, or spreadsheet programs. Users implementing software application hardening update their application code or add more software-based network security policies.

Operating system hardening

Operating system (OS) hardening is securing the operating system of an endpoint device such as a computer or mobile phone on a network. In computing, an operating system is a specific type of software that manages the basic functions of a device, such as starting and running programs. Strategies for OS hardening include installing or updating patches and reducing the number of people authorized for an organization's OS. Although related, operating system hardening and software application hardening are different processes. Software application hardening emphasizes protecting third-party programs, ie. H. Software created by companies other than the company that manufactures your device. At the same time, OS hardening focuses on improving the security of the underlying software that allows these third-party applications to run.

Database hardening

With database hardening, users can protect their digital databases and database management systems (DBMS). A database is a storage space for your company's valuable information that can be accessed digitally through devices or systems on a network. On the other hand, DBMS is the software users use when they want to access, store, modify or evaluate the information stored

in the database. Database hardening strategies include disabling unnecessary functions, encrypting database resources, and reducing user privileges.

Network hardening

Network hardening refers to securing communication channels and systems between servers, end devices, and other technologies running on a shared network. With all these systems and devices regularly interacting, one potential vulnerability could lead to a breach across the network. Businesses or individuals can improve network hardening by installing intrusion detection systems that detect suspicious activity, firewalls, and encrypting network traffic.

4.3.2 System Hardening Process

While IT infrastructure varies based on organizational needs and use cases, the technologies used to build these systems are common across industries. Whether it's marketing excellence or the quality of these products/services, this standardization makes it possible to create security configuration guidelines for each technology. Developed by cybersecurity experts, these guides provide a checklist of system hardening that organizations can apply to every technology that makes up their infrastructure.

These system hardening standards are freely available from entities such as the National Institute of Standards and Technology and the Center for Internet Security. Each repository contains a list of vendors, their technology offerings, and the benchmarks used to secure each listed technology. Additionally, the benchmark document contains recommendations and detailed instructions for implementing security measures based on use cases. In this format, security professionals can download documents applicable to their infrastructure and

create a checklist of hardening steps specific to their technology stack. If we look at an example of an organization that uses a Linux Server to host business applications and manage system access. Typically, a hardening checklist for securing such servers will include the following steps:

- Disable guest accounts and enforce strict password policies for all users.
- Regularly install updates, patches, and patches for the operating system.
- Restrict access to administrator accounts and set account lockout policies.
- Apply regular updates to third-party applications and antivirus software definition updates.
- Configure Active Directory Group Policy restrictions and enforce role-based access control.
- Disable unnecessary Windows services and close unused network ports.
- Configuring system of alerts.

The steps are more like a general checklist. A comprehensive system hardening guide will include additional steps and detailed information on the level of security each step provides and how to implement it.

4.3.3 Server Auditing

A system audit involves reviewing and evaluating control and computer systems and their use, efficiency, and security within the organization that processes information. By examining systems as an alternative to control, tracking and verification, computer processes and technologies can be used more efficiently and safely to ensure appropriate decisions are made.[\[29\]](#) In another way, Server auditing is not like a tax or compliance audit; instead, it's a way of tracking

and reviewing activities on your server. The process starts with creating an audit policy. These policies define the events you want to monitor and record, which you can then examine for potential security threats

Importance of Server Auditing

Server auditing is important for security, but it also helps keep operations running at your company. Servers are typically used for heavy workloads and large volumes of network traffic, and any impact to them can cause downtime, corrupt information, or a security breach, all of which can negatively impact your business.

With a defined audit policy, administrators can track changes or attempts to access critical information through Windows server auditing, Windows file server auditing, and SQL Server auditing. The results give administrators insight into the impact of the change performance degradation, for example and the ability to identify the level of the threat.

Auditing is also important from a compliance perspective. Many organizations use SQL Server to store sensitive information, and this data may be subject to HIPAA and SOX requirements, among others. If you have a SQL Server audit policy to monitor changes and modifications, you can create reports based on this information and demonstrate regulatory compliance.

Common Types of Audits

Although each business has its own audit needs, certain types of audits are commonly conducted. These include the following:

Account logins: Auditing account logins is essential to identify unauthorized login attempts. In addition, it's important to track successful logins to ensure only valid users are logged in.

Account management: Account management audits identify changes to roles and user accounts, keeping you updated on which users are authorized.

Object changes: Auditing object changes is necessary to see when a database object has been created, copied, changed, or dropped.

Policy changes: Auditing policy changes allows you to identify changes to the audit policy itself and confirm whether those modifications are expected, which is critical for accurate audits.

4.3.4 Benchmarks

They are configuration baselines and best practices for securely configuring a system. Each of the guidance recommendations references one or more Controls that were developed to help organizations improve their cyberdefense capabilities . They provide a clear set of standards for configuring common digital assets everything from operating systems to cloud infrastructure. This removes the need for each organization to 'reinvent the wheel' and provides organizations with a clear path to minimizing their attack surface. The best benchmarks for auditing a server and highly recommended are **CIS AND NIST**

NIST

:The National Institute of Standards and Technology is a non-regulatory government agency that develops technology, metrics, and standards to drive innovation and economic competitiveness at U.S.-based organizations in the science

and technology industry. As part of this effort, **NIST** produces standards and guidelines to help federal agencies meet the requirements of the Federal Information Security Management Act (FISMA). **NIST** also assists those agencies in protecting their information and information systems through cost-effective programs.

CIS

The Center for Internet Security is a nonprofit entity whose mission is to 'identify, develop, validate, promote, and sustain best practice solutions for cyberdefense.' It draws on the expertise of cybersecurity and IT professionals from government, business, and academia from around the world. To develop standards and best practices, including **CIS** benchmarks, controls, and hardened images, they follow a consensus decision-making mode.

Example of **CIS** benchmarks (CIS Debian Linux 10 Benchmark)

Profile Applicability:

Level 1 - Server

Level 1 - Workstation

Description:

The `/etc/ssh/sshd_config` file contains configuration specifications for `sshd`. The command below sets the owner and group of the file to root.

Rationale:

The `/etc/ssh/sshd_config` file needs to be protected from unauthorized changes by non-privileged users.

Audit:

Run the following command and verify Uid and Gid are both 0/root and Access does not grant permissions to group or other:

```
stat /etc/ssh/sshd_config
```

```
Access: (0600/-rw-----) Uid: ( 0/ root) Gid: ( 0/ root)
```

Remediation:

Run the following commands to set ownership and permissions on `/etc/ssh/sshd_config`:

```
chown root:root /etc/ssh/sshd_config
```

```
chmod og-rwx /etc/ssh/sshd_config
```

CIS Controls:

Version 7

Protect Information through Access Control Lists

Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

4.3.5 Logs System

A log file is a file that records events that occur during the execution of the operating system or other software. This is where Linux servers have log files that contain messages about the server, including the kernel, services, and applications running on it. Log files are located in the `/var/log` directory. There are four main types of log files generated in the Linux environment:

- Application Logs.
- Event Logs.
- Service Logs.
- System Logs.

You can observe and find detailed information about server performance, security, error messages, and potential problems from the log files. Therefore, any problems encountered by the server can be notified through the detailed view of the log file. Therefore, by looking at the log files, you can solve and prepare for future problems. The different files existing in the Linux are all with a specific task, as it is shown below :

`/var/log/messages`

This file contains the information of generic system activity. This is the log file which stores informational and non-critical system messages. This file stores mainly the non-kernel boot errors, application-related service errors, and the messages that are logged during system startup. If something goes wrong, then one should have to check this file first! Like you are facing some issues with the sound card. To check if something went wrong during the system startup process, you can look at the messages stored in this log file.

`/var/log/secure.log`

This file contains the information about all user authentication details. This log file is mainly used to get the usage of the authorization system. This file stores all security-related messages, including the authentication failure. This file saves the **sudo** logins, SSH logins, and other errors logged by the system security service daemon. This file is very useful for detecting hacking attempts. Also, this file stores the information about successful logins and can be used to verify the activities of valid users.

`/var/log/boot.log`

This file contains the information of all bootup message details. This file saves the messages of issues related to the improper shutdown, unplanned reboots, or booting failures. Log entries from this file are useful to detect the duration of system downtime caused by an unexpected shutdown.

`/var/log/kern.log`

This file contains the information logged by the kernel. Entries of this file are useful for solving kernel-related errors and warnings. Log entries of this file are helpful to detect the issues with the custom-built kernel and are also used for debugging hardware and connectivity issues.

`/var/log/fail.log`

This file contains the information of all failed login attempts. Entries of this log file are used to find any attempted security breaches involving username/password hacking and brute-force attacks.

`/var/log/cron.log`

This file contains the information of all cron jobs. If any of your cron has issues, you can find related entries from this file. When a cron job runs, this log file records all relevant information, including successful execution and error messages, in case of failures.

`/var/log/mail.log`

This file contains the information of all mail server-related details. This file saves the entries or information about postfix, smtp, MailScanner, SpamAssassin, or any other email-related services running on the mail server. One can track all the emails sent or received during a particular period. One can check failed mail delivery issues from the entries of this file. Details regarding any possible spamming attempts blocked by the mail server can be obtained from this file. One can detect the origin of an incoming email by detailed checking the entries of this file.

`/var/log/httpd/`

This directory contains the information about the logs recorded by the Apache server. This directory has two files – **error_log** and **access_log** which saves the information from Apache server. The **error_log** contains messages related to **httpd** errors such as memory issues and other system related errors. If something goes wrong with the Apache web server, check this log for diagnostic information.

`/var/log/mysqld.log`

This file contains the information of all mail debug, failure, and success messages related to the **mysqld** and **mysqld_safe** daemon. Entries of this file are used to identify problems while starting, running, or stopping **mysqld**. One can get the details about client connections to the MySQL data directory from this file.

4.4 System hardening Automation

Automated system hardening solution reduces the manual effort for golden image hardening by using Automation solutions that meet industry standards, such as CIS, NIST, and Azure Baseline Security. There has always been a constant battle between **agent-based** and **agentless** monitoring and management technologies. Some vendors claim to be agentless and do not require special agent deployment. At the same time, others require the use of proprietary proxies. Both methods have advantages and disadvantages. However, most ITOps teams combine agentless and agentless monitoring and need to use both most efficiently and cost-effectively.

4.4.1 Monitoring Architectures

Agent-less Architecture

For agentless monitoring, implementing an integrated SNMP agent extends to remote shell access, such as SSH. "Agentless" is a bit misleading. All management requires an agent, whether embedded in a management platform, managed device, or separately installed software.

The industry has embraced the de facto definition of agentless, that is, a management agent embedded in the device software or functions as a manager that does not require a separate installation or license. Agentless monitoring really means leveraging existing embedded capabilities. This type has some pros and cons, starting with the pros:[\[30\]](#)

- Does not require the installation of agents.
- OpManager supports a wide range of monitor types for agentless monitoring.
- Supports virtual hosts.
- Supports all kinds of servers.

And about the cons:

- Credentials must be shared.
- Dependent protocols like WMI/ SNMP must be enabled in the end system.
- Dependent ports in the end machine and OpManager server must be enabled.
- When OpManager is down, monitor data is not collected.

Agent-based Architecture

Agent-based technology enables in-depth monitoring and management. Every software vendor will tell you that their proxy is best for their platform. While this may be true, it could also be because their management platform was only built to work with their proprietary agent. The result is vendor lock-in, and switching vendors can lead to costly, large-scale, and long-term deployment of alternative technologies. As a result, when IT needs change, it can be very costly to meet them. Generally speaking, open standards and flexibility are better in the long run. Many providers will tell you that you have to choose between the two. Agentless monitoring vendors see this as their value proposition but ignore the fact that they have already invested heavily in agent-based technology. A proxy-based provider wants to sell you a proxy. The pros are as the following[30]

- Dependent protocols like WMI/ SNMP don't have to be enabled in the end system.
- Single way HTTPS outbound connection from end machine to OpManager server doesn't call for inbound connection requests to the end machine.
- Enabling of necessary ports will suffice rather than enabling all the ports.
- Monitor data is collected even when OpManager is down.
- Public exposure activities such as opening ports and allowing outside network requests are not needed.
- This monitoring technique is much more secure.

the cons of this type are as below:

- Agent must be installed on all monitoring servers.
- Only limited monitor types are supported for the time being in OpManager.

4.4.2 Available solutions

There is a list of popular DevOps "configuration management tools" on the Internet. With these tools, you can easily deploy, configure, and manage servers. They are easy to use and powerful enough to automate complex multi-tier IT application environments. The top four tools include Chef, Puppet, Ansible, and SaltStack. Choosing the right DevOps tool for your business needs and environment can be a bit of a hassle.

Chef

Chef is an automation platform that provides an efficient way to configure and manage infrastructure. The boss is using Ruby and a DSL language to write the configuration. Its architecture is similar to Puppet's master-agent model. It also uses a pull-based approach with additional logical boss workstations to drive configuration from host to agent. It provides a configuration in a Ruby DSL with a client-server architecture.

Puppet

Puppet is full-featured configuration automation and deployment orchestration solution. It is an open-source tool based on Ruby. In order to work, it relies on a custom domain scripting language (DSL) that is closer to JSON. It operates as a master client setup and uses a model-driven approach. Large enterprises often use it to automate system administrators who spend long hours configuring, deploying, troubleshooting, and maintaining server operations.

SaltStack

The SaltStack configuration tool is based on a master client setup model or a decentralized model. SaltStack provides the Python programming language to run commands over the SSH protocol using a push model. The platform also allows the grouping of clients and configuration templates for easy control of the environment. It provides low-latency, high-speed communication for remote execution and data collection in a system administrator environment.

Ansible

Ansible is an extremely simple IT automation engine that automates cloud provisioning, configuration management, application delivery, in-service orchestration, and many other IT needs. It uses no proxies and no additional custom security infrastructure, so it's easy to deploy - best of all, it uses a very simple language (YAML in the form of Ansible playbooks) that allows you to describe your automated tasks in English.

4.4.3 Ansible Architecture

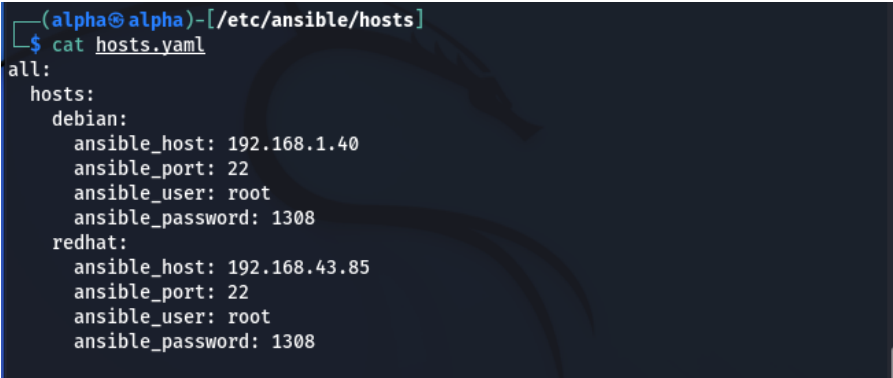
Ansible simplifies complex orchestration and configuration management tasks. It uses the Python language and allows users to write commands using YAML as a necessary programming paradigm. Ansible provides several push models for sending command modules to nodes via SSH running sequentially. In the table below, a comparison between the four solutions is made based on many factors. and our choice goes for this solution and the architecture will be discussed in the next section.

Modules

Ansible works by connecting to your nodes and emitting scripts called "Ansible modules" to them. Most modules accept parameters that describe the desired state of the system. Ansible then runs these modules (via SSH by default) and removes them when done. Your module library can reside on any computer and does not require a server, daemon or database.

Inventory

By default, Ansible represents the machines it manages in a file (INI, YAML, and so on) that puts all of your managed machines in groups of your own choosing. A basic YAML `/etc/ansible/hosts/hosts.yaml` might look like this:

A terminal window showing the command `cat /etc/ansible/hosts/hosts.yaml` and its output. The output is a YAML file defining an inventory of hosts. It starts with `all:` and `hosts:`. Under `hosts:`, there are two groups: `debian` and `redhat`. Each group has four parameters: `ansible_host`, `ansible_port`, `ansible_user`, and `ansible_password`.

```
(alpha@alpha)-[~/etc/ansible/hosts]
└─$ cat /etc/ansible/hosts/hosts.yaml
all:
  hosts:
    debian:
      ansible_host: 192.168.1.40
      ansible_port: 22
      ansible_user: root
      ansible_password: 1308
    redhat:
      ansible_host: 192.168.43.85
      ansible_port: 22
      ansible_user: root
      ansible_password: 1308
```

Figure 4.1: Ansible Inventory

Playbooks

An Ansible playbook is a model of automation tasks, which are complex computing operations that occur with little or no human intervention. Ansible playbooks run on a set, group, or classification of hosts, which together form an inventory. A basic YAML `/etc/ansible/playbook.yaml` might look like this

```
hosts: redhat
remote_user: root
become: yes
tasks:
  - name: Install Apache
    yum:
      name: httpd
      state: present
      register: "output"
      ignore_errors: true
  - name: Restart Apache
    service:
      name: httpd
      state: restarted
  - name: Remove Apache
    yum:
      name: httpd
      state: absent
```

Figure 4.2: Ansible Playbook

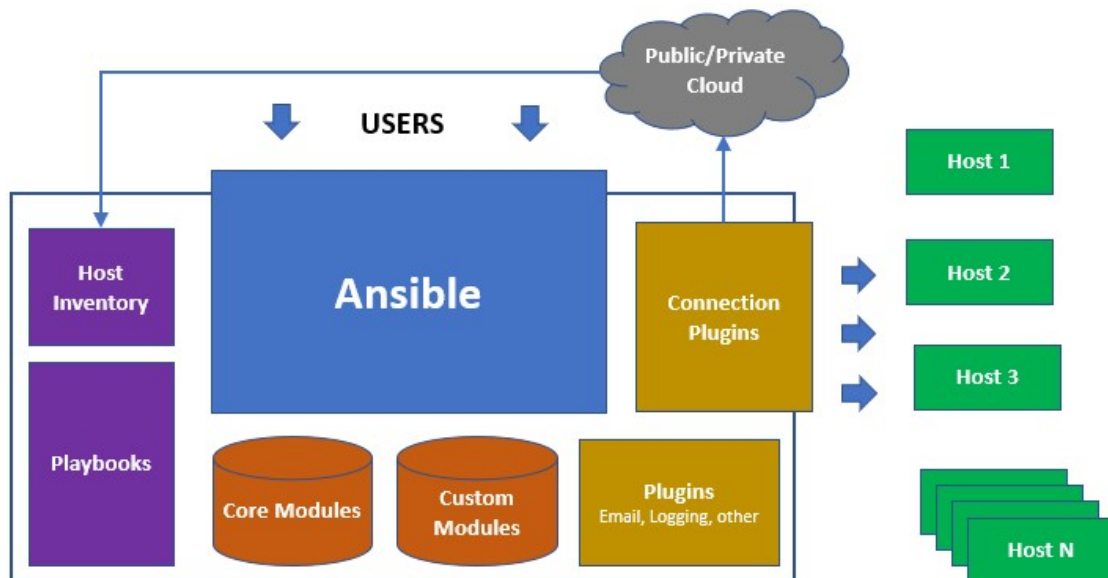


Figure 4.3: Ansible Architecture [28]

4.5 Conclusion

A reasonable level of computer security is not difficult to maintain on a home machine. More effort is required on business machines, but Linux can indeed be a secure platform. Due to the nature of Linux development, security fixes often come out much faster than they do on commercial operating systems, making

Linux an ideal platform when security is a requirement. For a more secure Linux, system Hardening the solution and to facilitate the process the practical side of the project will be discussed in the following part.

Part II

Management System and Automation of The Linux servers Hardening

Chapter 5

Conception

5.1 Diagram of Classes

Class diagram is the backbone of object-oriented modeling - it shows how different entities (people, things, and data) relate to each other. In other words, it shows the static structures of the system. A class diagram describes the attributes and operations of a class and also the constraints imposed on the system. Class diagrams are widely used in the modeling of object-oriented systems because they are the only UML diagrams that can be mapped directly to object-oriented languages.

A class is depicted in the class diagram as a rectangle with three horizontal sections, as shown in the figure below. The upper section shows the class's name (Flight), the middle section contains the properties of the class, and the lower section contains the class's operations (or "methods"). The purpose of the classes diagram can be summarized as:

- Analysis and design of the static view of an application;
- describing the responsibilities of a system;
- providing a base for component and deployment diagrams; and,

- Forward and reverse engineering.

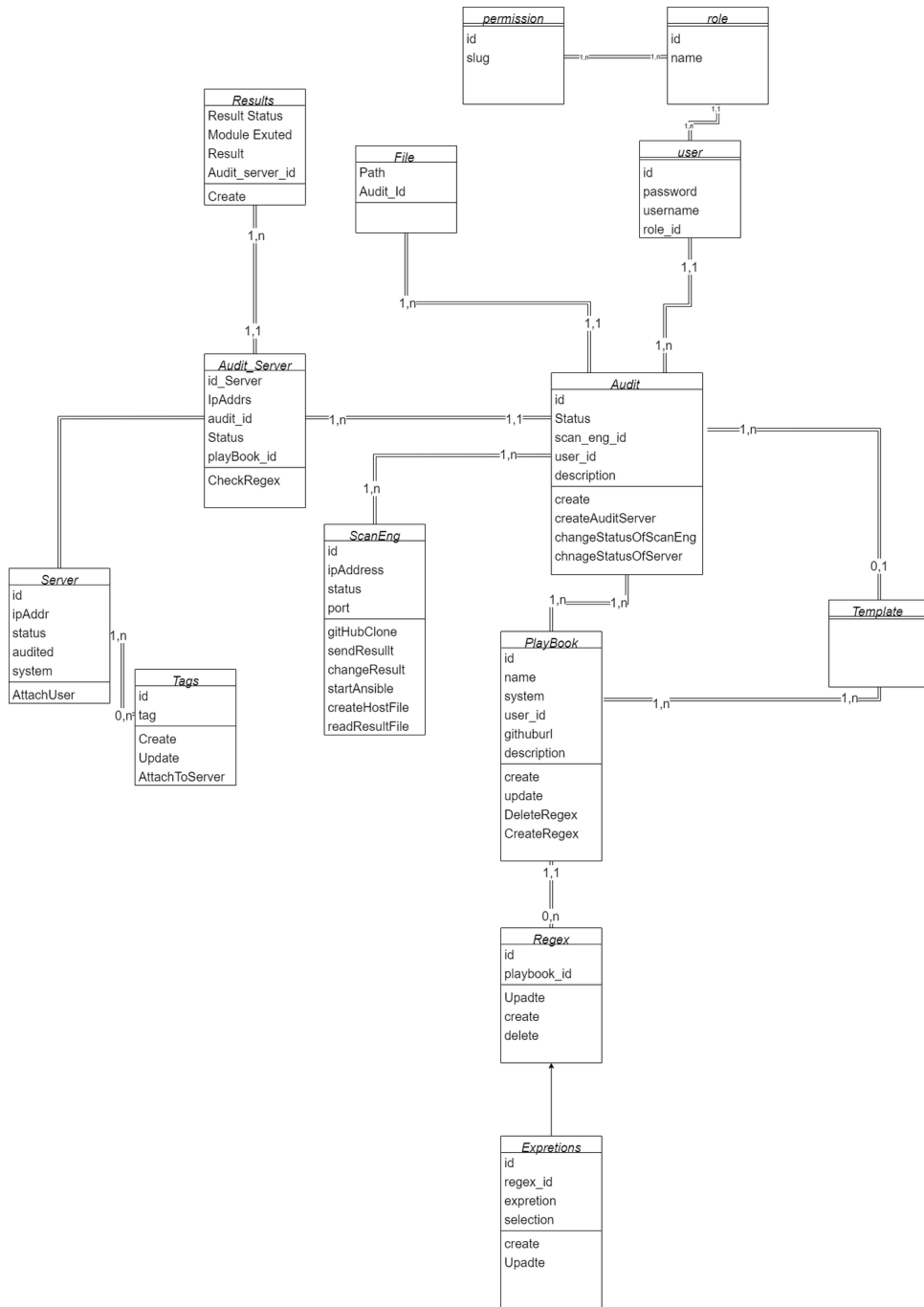


Figure 5.1: Diagram of classes

5.1.1 Description

Entities

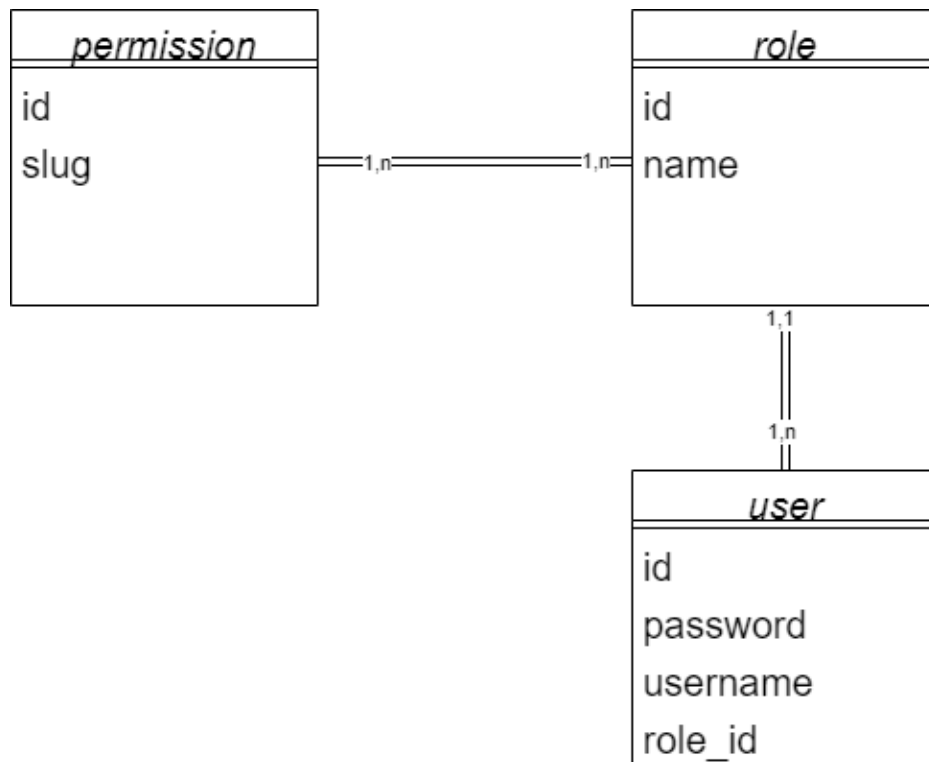


Figure 5.2: Diagram of classes / permission / Role / User

permission : entity that have (id,name,slug) the slug for permission and name is for description of permission

role : Entity with (id,name) the benefits of this entity is to not declare for every user a role but we can assign for users roles and they can have the same role (get the same permissions).

user : entity have (id,password,username) in this entity it is normal user with *role_id* to define the role of this user (the user can't has more than one role).

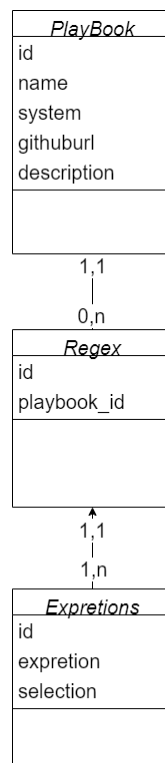


Figure 5.3: Diagram of classes / Playbook / Regex / Expreions

Playbook : this entity has (id,name,system,user_id,githuburl,description,tag_id)

. The name is used to search for this playbook , system for every playbook there is a specific system that we use . user_id to know the user who created this playbook ,githubUrl is the url of the playbook to use . description is for describing the playbook and the function inside it . a playbook can have 0 or n Regex .

Regex : Regex Entity has only (id,playbook_id,user_id) . regex can have only one playbook , regex can has 1,n expressions .

Expreions : this Entity has (id,regex_id,expression,selection) the selection is used for steps and where exactly we use this regex .

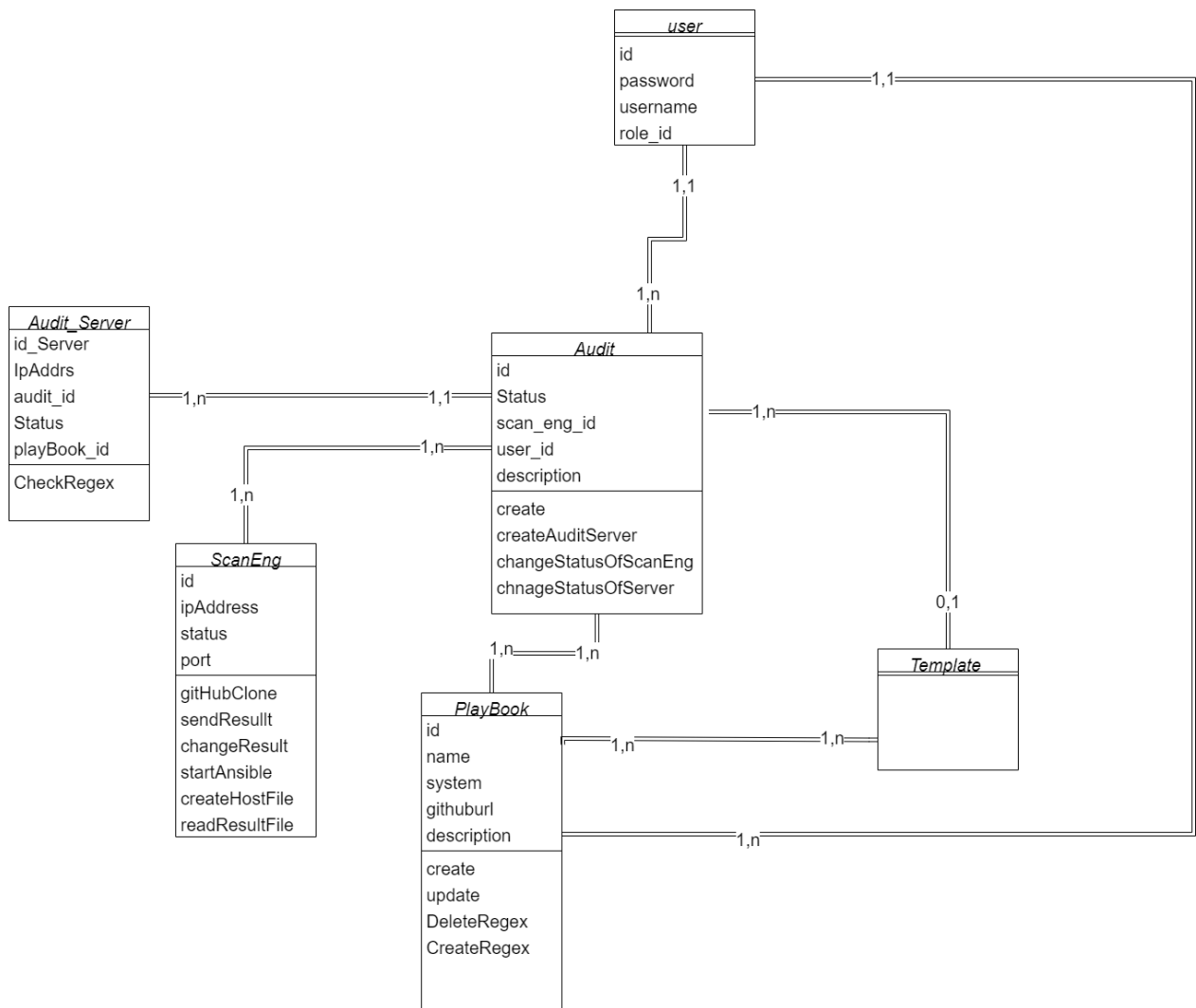


Figure 5.4: Diagram of classes / ScanEng / Audit / Playbook / Template/ user

ScanEng: this Entity needs (id,ipAddress,status,port) the status is used to see if this ScanEng is working or resting . the port is used when we send the data

Playbook : (id,name,system,user_id,githuburl,description,tag_id). The name is used to search for this playbook , system for every playbook there is a specific system that we use . user_id to know the user who created this playbook ,githubUrl is the url of the playbook to use . description is for describing the playbook and the function inside it . a playbook

can only one tag.

Template : it is not 100 % completed but the function of template is groupe of playbooks

user : entity has (id,password,username) in this entity it is normal user with role_id to define the role of this user (the user can't has more than one role). a user can have 1,n audited

Audit : this Entity has (id,status,scan_eng_id,description,user_id) the status is when audit is finished or still doing audit.

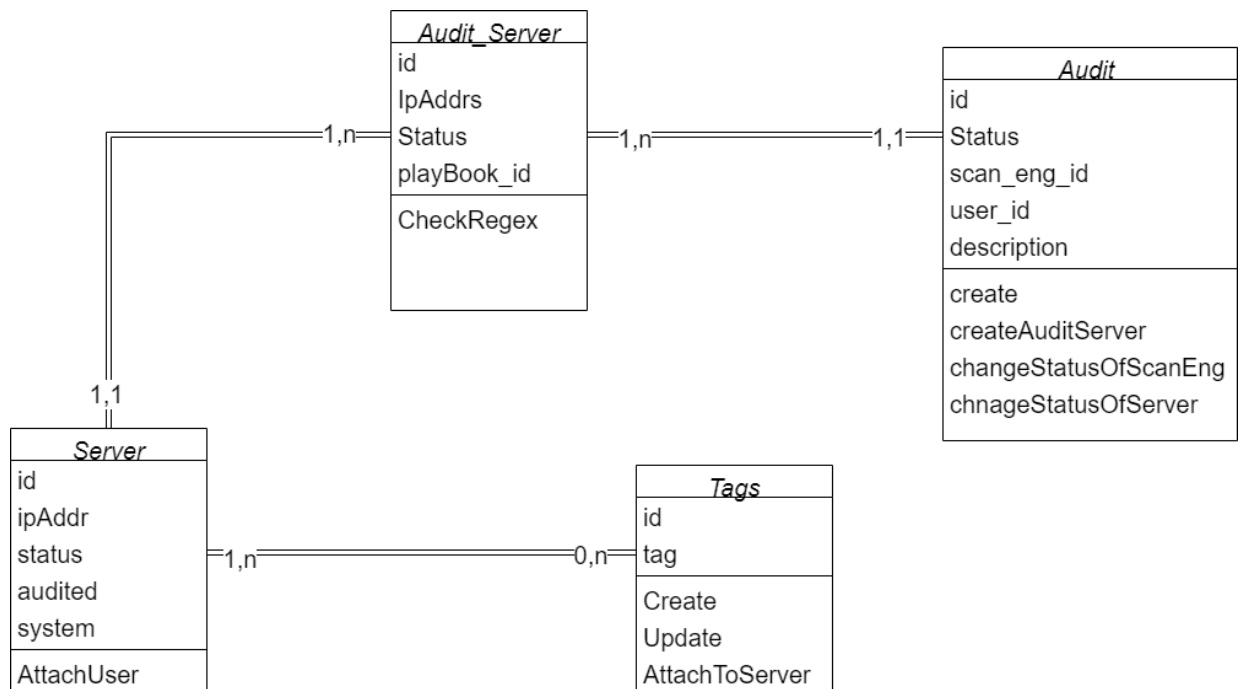


Figure 5.5: Diagram of classes / Server/ Audit / Tags

Audit_server: in this Entity we need (id,ipAddress,audit_id,Status, playbook_id,server_id)

Server: (id, ip_Addr, status, audited, system)

Tags: (id, tag)

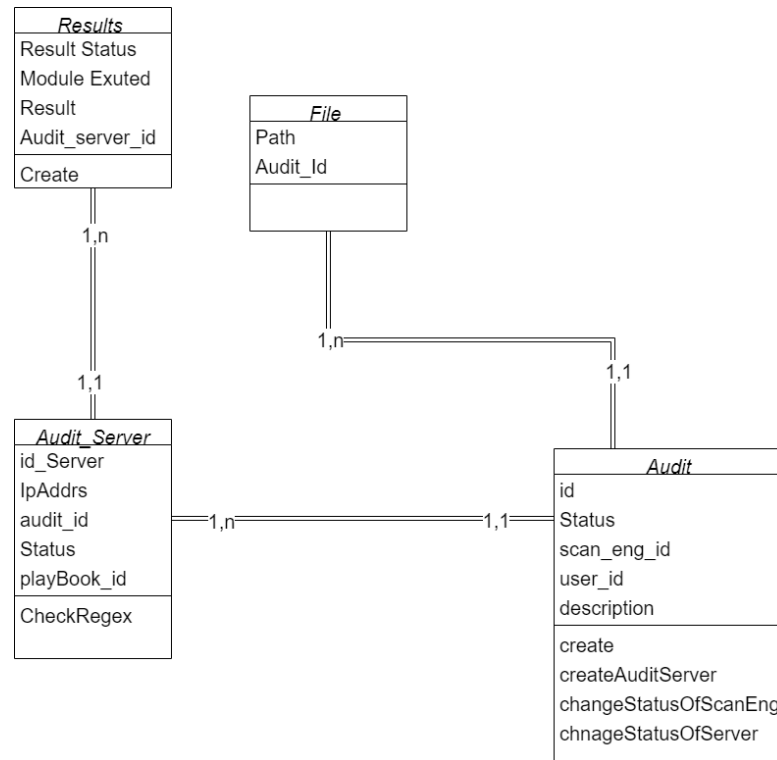


Figure 5.6: Diagram of classes / File / Results /Audit_server

File: in this entity we find the id,path,Audit_id

Results: Results Status ,Module Executed ,Result ,Audit_server_id

5.2 Use Cases Diagram

A use case diagram is a way to summarize details of a system and the users within that system. It is generally shown as a graphic depiction of interactions among different elements in a system. Use case diagrams will specify the events in a system and how those events flow, however, use case diagram does not describe how those events are implemented. The reasons why an organization would want to use case diagrams include:

- Represent the goals of systems and users.
- Specify the context a system should be viewed in.
- Specify system requirements.
- Provide a model for the flow of events when it comes to user interactions.
- Provide an outside view of a system.
- Show's external and internal influences on a system.

The general diagram of the use cases of our project is below:

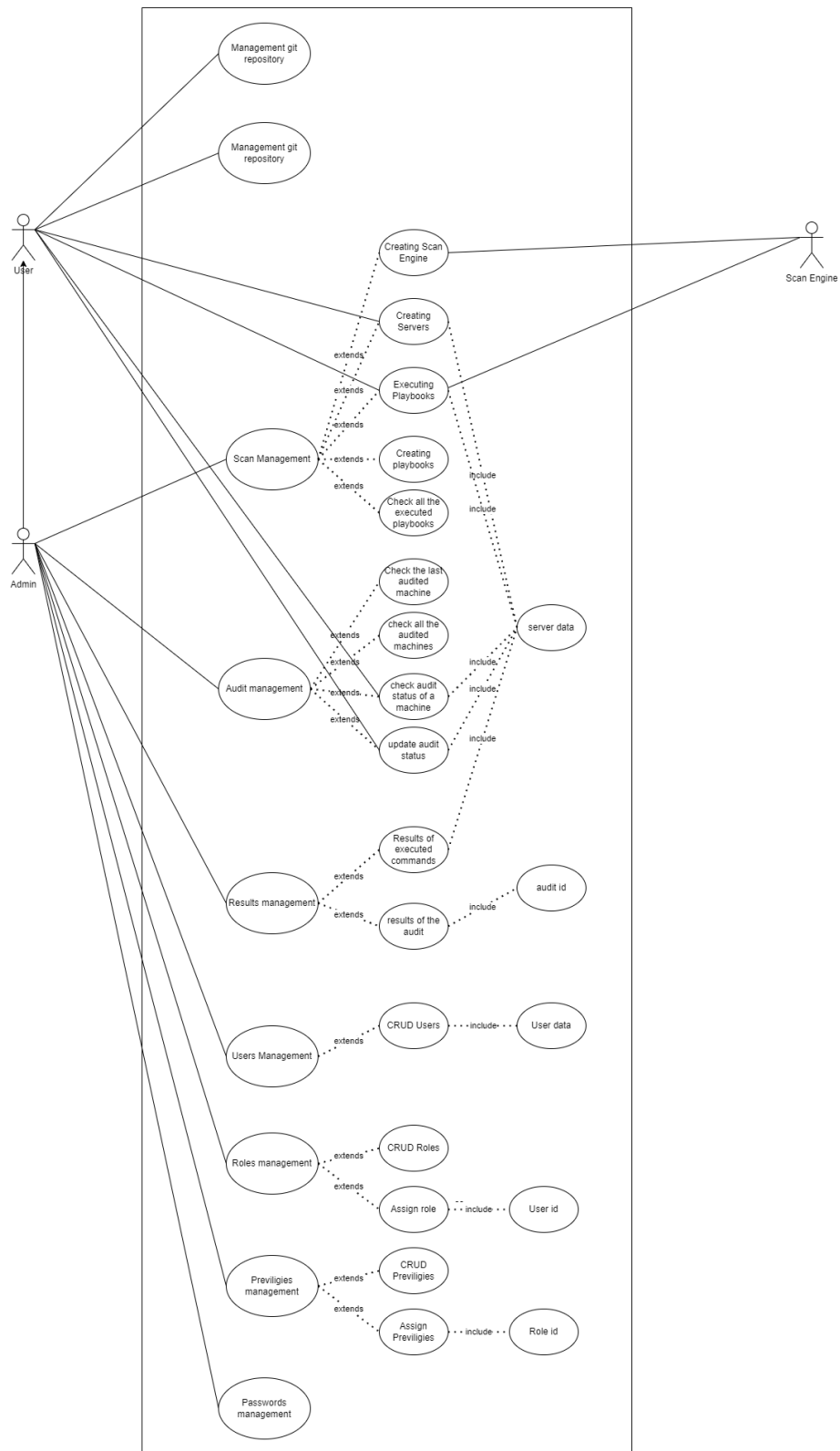


Figure 5.7: Diagram of the use cases

5.2.1 Description

User : like we see in the Figure the user has the possibility to do the following functions:

Management Of GitRepositories: this Function consists on creating and seeing what are the modification on this repository.

Creating Server : this Function is for the creation of servers and showing the details of a server.

Creating PlayBook : This Function is for creating Playbooks with showing the details of these playbooks

Executing Playbook: this function has another name it is creating an audit.

Creating ScanEng : this function is for the creation,management and showing details on the ScanEngine.

check Audit status : like the name suggests it is for the management of the audit and seeing which Server is not Audited.

Update audit status: Update Status Of a Server From Not Audited to audited or the opposite.

Admin : As We see that the admin is doing everything that user can do and more

Audit Management : it is creation and update and delete and modification of audit

Scan Management : it is creation and update and delete and modification of scanEng Playbook or Server or tags

Result Management : it is update and delete and modification for status of audit and checking the results.

Roles Management : it is creation and update and delete and modification role or attaching it to a user .

privileges Management : it is creation and update and delete and modification of the permissions or add permission to role.

Password Management : it is everything creation and update and delete and modification of the users.

Chapter 6

Development and Implementation

6.1 Introduction

In this part, we will detail the stages of the development of the solution, the choice of tools initially, and the demonstrations of the different useful features.

6.2 Tools Used:

6.2.1 Visual Studio Code :

is a source code editor developed by Microsoft for Windows, Linux, and macOS. That Features include support for debugging, syntax highlighting, smart code completion, code snippets, code refactoring, and Built-in Git, which can be used with various programming languages such as Java, JavaScript, Go, Node.js, Python, and C++.

6.2.2 XAMP Server :

XAMPP is an easy-to-install Apache distribution. It contains MySQL, PHP, and Perl. Just download and run the installer. It's that simple.[\[31\]](#)

6.2.3 Laravel v9:

Laravel is a web application framework written using expressive and elegant syntax in **PHP**. Laravel tries to simplify development by simplifying common tasks. In most web projects, such as authentication, routing, sessions, and settings hidden. Laravel is designed to allow developers to At the expense of app functionality. It provides the powerful tools needed for Big and powerful applications. It's hard to talk about frameworks without mentioning the following:

MVC (Model-View-Controller)

Models are responsible for managing data. Views are responsible for formatting for the user. The controller is responsible for managing the whole. In general, let's summarize, the model manages the database, the view creates the page HTML, and controllers do everything else.

ORM Eloquent

The PHP Laravel framework comes with the Eloquent Object Relational Mapper (ORM), which refers to a high-level implementation of the PHP Active Record pattern to make it easier Interaction with the application database. Meet different business needs through faster development, And well-organized, reusable, maintainable, and extensible code. It works. Customize the web application as it can manage and run multiple databases General database operations. Developers can efficiently use multiple databases in Eloquent Data with ActiveRecord implementation. It is an architectural model in which Models are created in the Model-View-Controller (MVC) structure corresponding to the database. The benefit is that the model performs common database operations without Writing long SQL query code. Models allow you to query your

data table and insert new data records into the table.

JSON Files:

is an open standard file format and data interchange format that uses Human-readable text objects for storing and transmitting data consisting of pairs of Property values and arrays. This is very Mainstream; there are various applications. An example is a web application Communicating with the Server. JSON is a language-independent data format. It is derived from JavaScript but is derived from Many modern programming languages that contain code for the generation and parsing of data in JSON format.

Composer

A composer is a tool that includes all the dependencies and libraries. It allows a user to create a project with respect to the mentioned framework (for example, those used in Laravel installation). Third-party libraries can be installed easily with the help of a composer.

All the dependencies are noted in the composer.json file, which is placed in the source folder.

Artisan

Command-line interface used in Laravel is called Artisan. It includes a set of commands which assists in building a web application. These commands are incorporated from the Symphony framework, resulting in add-on features in Laravel 5.1 (the latest version of Laravel).

Modularity: Laravel provides 20 built-in libraries and modules, which

helps in the enhancement of the application. Every module is integrated with the Composer dependency manager, which eases updates.

Testability: Laravel includes features and helpers which help in testing through various test cases. This feature helps in maintaining the code as per the requirements.

Routing: Laravel provides a flexible approach for the user to define routes in the web application. Routing helps to scale the application in a better way and increases its performance.

Configuration Management: LA web application designed in Laravel will be running on different environments, which means that there will be a constant change in its configuration. Laravel provides a consistent approach to handling the configuration in an efficient way.

Features of Laravel

Laravel offers the following key features, which make it an ideal choice for designing web applications.

6.2.4 Node JS

NodeJS is a platform built on top of Google Chrome's JavaScript executor for developing scalable and modular web applications. This development can be done easily, quickly, and efficiently. NodeJS uses event-driven programming to encourage exchanges between client and Server. Also, input and output are done in a non-blocking manner, making NodeJS a lightweight and efficient tool. Applications that perform intensive data exchange in real-time will benefit from this. Because NodeJS is based on JavaScript, all systems that support JavaScript browsers are client-side compatible. On the server side, all you have

to do is install the node program, and then you can run it on the command line.

6.2.5 YAML

YAML is an easy-to-digest data serialization language commonly used to create configuration files in any programming language.

6.2.6 Bootstrap

It is a collection of useful tools for design creation (graphics, animation, and Interaction) with pages in the browser, etc.) for websites and applications. It's a sentence that Contains HTML and CSS code, forms, buttons, navigation tools, more Interactive elements, and optional JavaScript extensions. This is the most Popular on the GitHub development management platform.

6.2.7 GitHub

is an open standard file format and data interchange format that uses text Human-readable object for storing and transmitting data consisting of pairs Property values and arrays (or other serializable values). This is a very Main-stream, there are various applications, an example is a web application Communicate with the server. JSON is a language-independent data format. It is derived from JavaScript, but is derived from Many modern programming languages contain code for generation and parsing Data in JSON format.

6.2.8 AdminLte

is a popular open-source WebApp template for admin dashboards and control panels. It is a responsive HTML template that is based on the CSS framework

Bootstrap 3. It utilizes all of the Bootstrap components in its design and restyles many commonly used plugins to create a consistent design that can be used as a user interface for backend applications. AdminLTE is based on a modular design, which allows it to be easily customized and built upon. This documentation will guide you through installing the template and exploring the various components that are bundled with the template.[\[32\]](#)

6.2.9 Ansible

Ansible is an open-source software deployment, configuration management, and application deployment tool that supports infrastructure as code. It runs on many Unix-like systems and can be configured for Unix-like systems and Microsoft Windows.

6.3 Architecture of the Solution

Let's say that we have an IT engineer who knows how to audit a server. Firstly he needs to know the Server (system and historic, ipAddress, user, root user ...). The employee needs to know a lot of things about this Server to start an audit. Not only is it very risky, repetitive, and Tiring. And that's not all. After that, he needs to know about the benchmark and what he needs to execute for this Server. After that, he needs to start auditing the Server, and sometimes auditing a server can take days or more. And As we see in big companies, Like Elit has thousands or more than 10 thousand servers. Not only do they need more IT engineers, but they need to know how to work with benchmarks. It is very risky to give access to an employee that can do anything with that Server and change and modify and delete. Like sometimes, employees need root access, but not only that, the employee can forget the history of his audits. There are

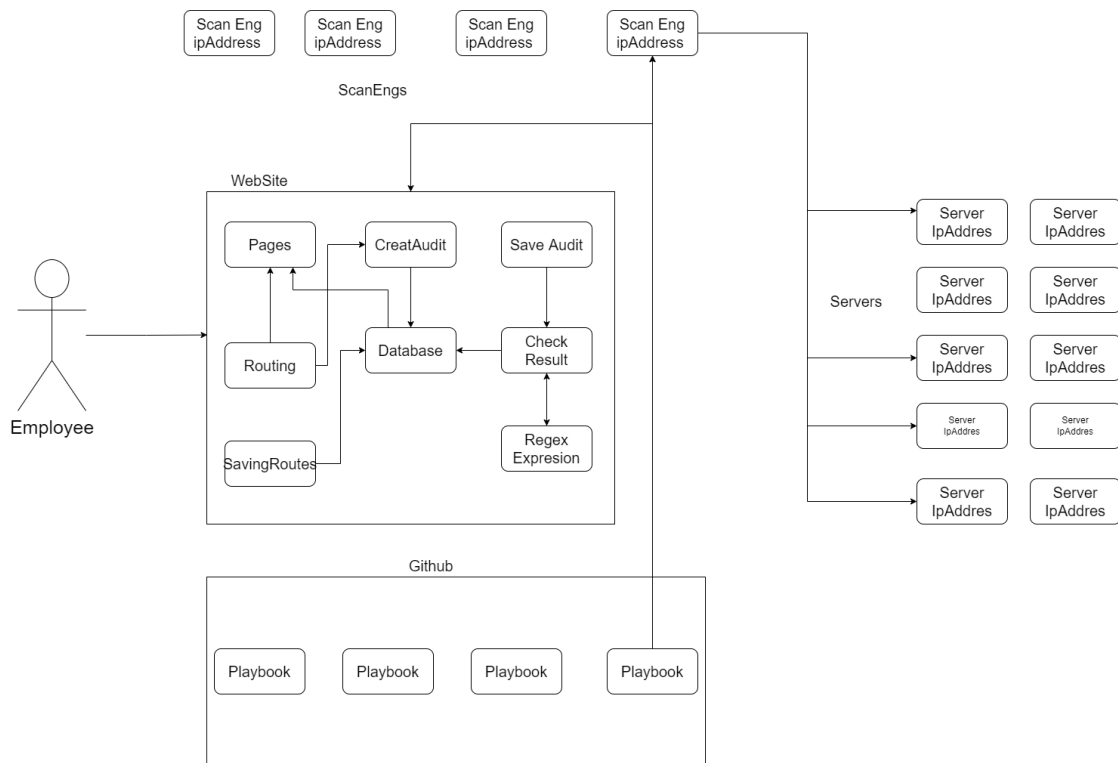


Figure 6.1: Project Architecture

a lot of problems in auditing a server. That's why we made this solution With Laravel nodejs and Ansible.

Like We see in the figure the three main objects **Scan Eng** , **Server**, **Web-Site**.

WebSite: is for the management and services and communication between a user and the other objects (ScanEng and Server) even between users,

ScanEng: it is Server, but with Ansible, git, and nodejs installed, all of these installation needs to have successful ScanEng .

Server: As the name suggests, it is a server, and it is the most important object in the project. The Server can be a Mysql server, Apache server, WebSite

...

6.4 How the Application works

The process will be explained with an example of an employee who needs to do an **audit** on the Mysql server.

1. Firstly, the employee needs to select the Server, and the selection is based on a **root** user or **user with privileges**. He can select one or more.
2. Here, it is the time to select a free **ScanEngine** to scan the servers
3. lastly the selection of playbooks. The selection can be composed of one or more than one playbook, but it needs to be arranged in a specific classification because the order of execution of playbooks is important
4. the playbooks are made as a form and sent to the **scanEngine**.

Now the process of the **ScanEngine** must be explained below:

1. Inside the ScanEngine, a GitHub repository that contains a playbook is cloned
2. then the mapping or in other way the classification of the playbooks is made.
3. now it is the time of the hosts' files that contain the IP addresses of the servers and users that are used
4. after the execution of the first playbook, the results are being sent to the user, and this process repeats for all the stack of the playbooks.
5. At the end of the execution, the status of the **ScanEngine** and the servers are changed to resting.

6.5 The Process of Creation

creation of playbook :

The playbook is a playbook of ansible which is stored inside github repository . you ask why we did it inside github and not inside application because github give us the management of playbook and to not duplicate anything we used it for management and seeing what is inside , Example of Playbook. Like we see in after every module there module and that module is used save the result of the module before it inside list . Lastly we create a file for the result.

Creation of the scan Engine:

To create a scan Eng we just need an ip Address and port the ipAddress to let us know which scanEng we are using because it is the server we need it to know where to send the form of audit to if we are using it locally .

Creation of the scan Engine:

To create a scan Eng we just need an ip Address and port the ipAddress to let us know which scanEng we are using because it is the server we need it to know where to send the form of audit to if we are using it locally .

Creation of the Server

To create a server we just need an ip Address

creation of audit :

To create Audit we need to select a server playbook and scan which is resting and not working . after the selection of these three and doing the mapping

for each server with a playbook . The application created for us a model audit_server just to know where which playbook is executed now with what server . and it attaches a regex expression with the playbook . After finishing the audit it will make the status of audit_server finished and see if it is audited or not .

Creation of a Regex Expression :

the regex expression is for checking the results and seeing if there is a positive result and it can predict how the result will be but with expression .

Creation result file :

after finishing a playbook on the scanning we just save the playbook results in file and get it in the database to check the results later if we needed .

creation of report monthly of server :

for this function we added it to see where exactly a server is audited and sometimes a company needs to make sure that clients servers are audit properly .

Conclusion

System hardening aims to eliminate as many security risks as possible. This is usually done by removing all non-essential software programs and utilities from the computer. This process should take a lot of time for an engineer, and the solution proposed aims to reduce this lost time. This last was at the organization's expectations.

This project gave us the opportunity to develop more our skills on the web development and to know more about cybersecurity and the system administration and becoming more familiar with the linux system and getting to know the domain of the DevOps and the manipulation of servers.

We tried to do our best in this Short time we faced various types of problems and we managed to solve them. We are planning to add more features and develop our solution more and the detection of vulnerabilities system is one of our major points to be added in the next versions. We Hope this project should give a hint or a help for the upcoming projects and giving the courage for Students to do such projects in the DevOps Domain

Bibliography

- [1] Sebastien ROHAUT. *LINUX Maitrisez l'administration du systeme*. Editions ENI. (2017).
- [2] David A Rusling. *The LINUX KERNEL -Version 0.8.3 .* (2001)
- [3] The Open Group Base Specifications Issue 7 2018 edition. *Headers*. <https://tinyurl.com/2p82s7sm>
- [4] The Open Group Base Specifications Issue 7 2018 edition. *File Format Notation*. <https://tinyurl.com/566dn2za>
- [5] The Open Group Base Specifications Issue 7 2018 edition. *Environment Variables*. <https://tinyurl.com/5ynfus9k>
- [6] The Open Group Base Specifications Issue 7 2018 edition. *Locale*. <https://tinyurl.com/3pr77pfr>
- [7] The Open Group Base Specifications Issue 7 2018 edition. *Regular Expressions*. <https://tinyurl.com/yckzhxf2>
- [8] The Open Group Base Specifications Issue 7 2018 edition. *Directory Structure and Devices*. <https://tinyurl.com/2xsen9bp>
- [9] tutorialspoint. (May 2022). *Unix / Linux - Getting Started*. <https://www.tutorialspoint.com/unix/unix-getting-started.htm>
- [10] *Linux File Hierarchy Structure*. (2022) <https://cochiselinuxusergroup.org/Resource>

- [11] K. C. Wang. School of Electrical Engineering Washington State University Pullman, WA, USA. *Systems Programming in Unix/Linux*. Springer International Publishing (2018)
- [12] Oreilly Edition. (1st Edition January 1999). *Open Sources: Voices from the Open Source Revolution*
<https://www.oreilly.com/openbook/opensources/book/appb.html>
- [13] javatpoint (April 2022). *Architecture of Linux*.
<https://www.javatpoint.com/architecture-of-linux>
- [14] Oracle Documentation *Introduction to the PAM Framework*
<https://docs.oracle.com/cd/E19120-01/open.solaris/819-2145/pam-01/index.html>
- [15] Cambridge Dictionary.(2022). *Définition de administration en anglais*.
<https://dictionary.cambridge.org/fr/dictionnaire/anglais/administration>
- [16] DICTIONARY.COM.(2022). *Definition of administration*.
<https://www.dictionary.com/browse/administration>
- [17] Christopher Negus. *Linux Bible*. John Wiley & Sons, Inc.9th edition (2015)
- [18] Debian Documentation.(2022) *Chapter 8. The Debian package management tools*
<https://www.debian.org/doc/manuals/debian-faq/pkgtools.en.html>
- [19] Simran Kaur.(september,2021). *How do I List All Groups in Linux*
<https://linuxhint.com/list-all-groups-linux/>
- [20] Thomas Tuffin (Red Hat, Sudoer),(March 31, 2021). *How to manage Linux passwords with the pass command*
<https://www.redhat.com/sysadmin/management-password-store>

- [21] Rimuhosting. *Managing services in Linux*
<https://rimuhosting.com/knowledgebase/linux/managing-services>
- [22] Justin Ellingwood.(January 24, 2022).*Understanding Systemd Units and Unit Files*
<https://www.digitalocean.com/community/tutorials/understanding-systemd-units-and-unit-files>
- [23] Diigital Guide IONOS.(April 22, 2022) *What Telnet is and how do you enable it?* <https://www.ionos.com/digitalguide/server/tools/telnet-the-system-wide-remote-protocol/>
- [24] SSH Academy. *OpenSSH: SSH key management needs attention*
<https://www.ssh.com/academy/ssh/openssh>
- [25] Sam Ingalls.(July 23, 2021) *What Is A Virtual Server?*
<https://www.serverwatch.com/guides/virtual-server/>
- [26] Domantas G. (jun 01, 2022) *How Does SSH Work*
<https://tinyurl.com/4nbvxref>
- [27] Indeed Editorial Team. (March 30, 2022) *What Is System Hardening? (Definition and How It Works)*
<https://www.indeed.com/career-advice/career-development/what-is-system-hardening>
- [28] Shaik Rabbani. (May 2021). *The Ansible Architecture*
<https://www.ecanarys.com/Blogs/ArticleID/401/The-Ansible-Architecture>
- [29] Richard Daniels. (August 8, 2020) .*What is Systems Audit and What are the Objectives of System Audit?*
<https://www.businessstudynotes.com/finance/auditing/systems-audit/>

-
- [30] ManageEngine OpManager.(2022) *Agent-based vs Agentless monitoring: A Comparison*. <https://www.manageengine.com/network-monitoring/help/agent-based-vs-agentless-monitoring.html>
- [31] *About Xampp*. <https://www.apachefriends.org/fr/download.html>
- [32] *AdminLTE Documentation (Version 2.3)*
<https://adminlte.io/themes/AdminLTE/documentation/>