UNIVERSIDAD YACHAY TECH

ESCUELA DE CIENCIAS MATEMÁTICAS Y COMPUTACIONALES

Final Project

Design and Implementation of Core Computer Services in a LAN for a Small Organization

Student Name: Daniel Troya A.

Instructor Name: PhD. Rolando Armas

Course : Management of Computing Services

Date: May 26, 2025

School of Mathematical and Computational Sciences

Contents

1 Executive Summary

1.1 Case Study

A small company is structured into three departments: **Administration**, **Sales**, and **TICs**. Each department is equipped with a maximum of 10 terminals (PCs). The Administration and Sales departments each utilize a dedicated printer. The IT department is responsible for maintaining all servers. Both printers and servers require static IP configurations.

You have been appointed to design and implement the organization's network infrastructure and essential services, ensuring that the following technical requirements are met:

Requirements

- Design an optimal network topology tailored to the departmental layout and PC distribution.
- Utilize a Class C private IP range (192.168.0.0 to 192.168.255.255). Each student must select a unique subnet.
- Implement a DHCP server to dynamically assign IP addresses to all terminals.
- Configure Network Address Translation (NAT) to provide internet access to all internal hosts.
- Set up secure SSH access to all servers using key-based (public/private) authentication.
- Deploy and configure an FTP server allowing anonymous access to a specific directory for uploading and downloading files.
- Install and configure a database server accessible from any terminal in the network. Create and test a small sample database.
- Set up a web server accessible from all terminals within the network.
- Develop a user-friendly shell menu for novice users to perform basic Linux operations.
- Configure appropriate users and groups for each service (FTP, SSH, Web, Database) on the servers.
- Conduct a basic vulnerability assessment of the network using `nmap`.
- Ensure at least one terminal is configured with a database client (e.g., `mysql-cli`) to connect to the database server.
- Set up at least one terminal with a web browser to verify connectivity to the internal web server and the internet.

- Deploy a network monitoring service to oversee the health and performance of the system.

1.2 Services to implement

1.2.1 DHCP Service

The DHCP (Dynamic Host Configuration Protocol) service allows IP addresses to be assigned automatically to devices within a network, eliminating the need to manually configure each terminal. This significantly reduces configuration errors and improves network administration efficiency.

One key advantage of DHCP is that it automatically detects when an IP address is no longer in use and can reassign it to another client. Additionally, it allows you to define a specific range of IP addresses that the server can assign.

For a client to receive an IP address via DHCP, it must have the DHCP client service enabled, which is included by default in most modern operating systems.

Features

- Automates IP address assignment, reducing human error.
- Requires configuration only on the server side.
- Efficiently reuses IP addresses that are no longer in use.

Vulnerabilities

While DHCP servers are extremely useful for automatically assigning IP addresses, they also present certain vulnerabilities that, if exploited, can seriously compromise the integrity of a network. Some of the most common threats include:

- **DHCP Spoofing:** An attacker can impersonate the legitimate DHCP server and redirect traffic between clients and other network services, potentially leading to man-in-the-middle attacks.
- **DHCP Starvation:** By sending a large number of fake DHCP requests, an attacker can exhaust the pool of available IP addresses, preventing legitimate devices from receiving valid network configurations.
- **Injection of Fake Parameters:** A hacker can provide incorrect settings, such as false DNS or gateway addresses, redirecting traffic to untrusted networks for malicious purposes.

1.2.2 FTP Service

The FTP (File Transfer Protocol) is used, as its name suggests, to transfer files between computers over a network using the Internet protocol. It is commonly employed to connect to servers for the purpose of downloading or uploading files. In order to transfer files, users must have access permissions to the FTP server.

There are two primary types of FTP access: one that requires user credentials (username and password), and another called **anonymous FTP**, which allows users to access a specific directory without authentication. This is useful for public file sharing, but must be configured carefully to avoid security issues.

To properly use the FTP service, configuration is required on both the client and the server sides. Wrong configurations can often lead to access issues or permission errors.

FTP operates efficiently by relying on a request-response communication model. It waits for confirmation from the receiving side before proceeding with the data transfer, which helps ensure synchronization and minimizes potential transmission delays or interruptions.

Features

- Enables file upload and download between devices across a network.
- Supports both authenticated and anonymous access.
- Requires proper setup on both client and server to avoid permission or accessibility issues.
- Uses a reliable communication model to ensure stable file transfers.

Vulnerabilities

FTP can transmit sensitive data such as usernames, passwords, and email addresses in plain text, making it highly vulnerable to interception. If an attacker is monitoring the network (sniffing), they can easily capture this unencrypted information. Additionally, FTP servers are susceptible to brute-force attacks, where attackers repeatedly attempt different username and password combinations to gain unauthorized access. Once inside, they may upload malicious files or scripts that can compromise the integrity and security of the server.

1.2.3 Web Service

Web server is software that uses the HTTPS (Hypertext Transfer Protocol Secure) protocol to send and receive requests, allowing the exchange of data between clients and the server. Through this communication, users can access and interact with websites.

A typical website consists of several files written in different programming languages and is divided into two main components:

- **Frontend:** Visual and creative part of the website — the layout, design, and user interface. It is what users directly interact with in their browsers(clients).
- **Backend:** This part manages the logic, data processing, web server configuration, APIs, and database interactions — essentially the "engine" behind the website.

All necessary files and resources for a website to function properly are stored on the web server. The server handles incoming user requests, which may include loading pages, sending forms, or accessing databases. If too many requests are made at once, the server can experience high load, leading to **latency** and slower response times.

Vulnerabilities

One common type of attack targeting web servers is the **DDoS** (Distributed Denial of Service) attack. In this scenario, a large number of compromised devices (often controlled by hackers as part of a botnet) send continuous requests to the web server. This overloads the server with traffic, consuming its resources and potentially causing it to crash or become inaccessible to users.

1.2.4 Database Service

The database service is hosted on a dedicated server with the primary objective of storing and managing structured data. This allows other computers (clients) within the network to connect and interact with the database. Thanks to the database engine and supporting components, the server can receive and respond to **queries**, which are essential for retrieving, analyzing, and modifying stored information.

A critical aspect of database management is the assignment of access permissions. Granting privileged access must be done with caution, as intentional misuse to severe consequences, such as the loss of important data through the execution of harmful commands.

Features

- Database servers can implement authentication mechanisms to enhance security and ensure only authorized users can access or modify the data.
- Server performance can be optimized by allocating system resources (such as RAM) specifically for database operations, allowing better handling of multiple simultaneous requests and faster query processing.

Vulnerabilities

One of the most knowing vulnerability in database systems is the **SQL Injection** attack. In this case, an attacker sends specially crafted queries to exploit poorly validated inputs, potentially gaining access to sensitive data such as administrator usernames and passwords. In severe cases, attackers would **dump the database**, extracting large volumes of confidential information.

Another important risk is the use of outdated database software. Running unpatched or legacy versions can expose known security flaws. Regular updates and patches are essential to mitigate these risks and protect the server from cyberattacks.

1.2.5 SSH Service

The SSH (Secure Shell) service is a very useful tool, as it allows for secure connections between computers within the same network. This protocol uses encryption to ensure that data transfers and communications are protected.

SSH is most commonly used to remotely manage Linux servers from another machine. To establish a secure connection, it is necessary to generate a key pair: a public key and a private key. The public key must be shared with the server you wish to access, while the private key remains on the client. In some cases, a password may also be required to complete authentication; however, if key-based access is properly configured, it is possible to log into the server without a password.

Vulnerabilities

A significant vulnerability was identified in versions prior to OpenSSH 9.6, known as **CVE-2023-48795**, or "Terrapin." This vulnerability allows an attacker to weaken the security of an SSH connection. The issue occurs during the connection negotiation phase, where *sequence numbers*—values that indicate the order of encrypted packets to ensure their integrity and authenticity—are used. Terrapin exploits a flaw in how those numbers are validated, allowing certain key messages to be bypassed and the security of the encrypted channel to be downgraded.

1.3 Tools

1.3.1 Ubuntu Server

Ubuntu Server is an operating system specialized in server and data center management, designed to be highly scalable and efficient. It is widely used to run critical services in enterprise environments, from small local networks to complex cloud infrastructures.

One of its advantages is its compatibility with tools like Kubernetes, which simplifies container orchestration and automates tasks such as service execution and backups. Unlike the desktop version, Ubuntu Server does not include a graphical interface by default, as it is intended to be operated via the terminal, reducing resource usage.

In this project, we will use virtual machines running Ubuntu Server as the base system. Through the terminal, we will configure various essential services such as DHCP, SSH, web servers, FTP, and databases.

1.3.2 Nmap

Nmap is a free and open-source software tool originally developed as a port scanner for Linux systems. Over time, it has been continuously improved by the community, gaining many advanced features that allow users to gather detailed information about a network, such as open ports, active services, and connected hosts.

One of its most powerful features is the ability to detect potential vulnerabilities on devices within the network. Nmap also has extensive documentation, including an official book, which provides detailed explanations of its commands and usage techniques.

It is important to note that Nmap must be used responsibly. Unauthorized scanning of private networks can be considered illegal or a violation of privacy laws, depending on the context and jurisdiction.

1.3.3 Docker

Docker is software that allows us to package and run applications within an isolated environment called a *container*. These containers are separated from the host operating system, making it easier to run multiple services or applications without interference between them.

One of its main advantages is that it enables the creation of customized environments quickly and efficiently. In this project, Docker will be used to simulate the necessary Linux servers and machines, allowing for a more realistic and modular implementation within the virtual environment defined in section (1.1).

Thanks to Docker, we can distribute and configure services such as DHCP, FTP, SSH, databases, and more, without needing to install each one directly on the host system. This improves both portability and control of the environment.

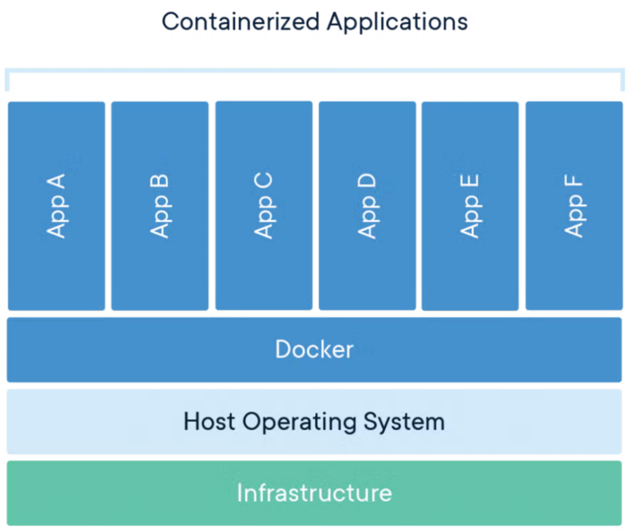


Figure 1: Docker: Graphical Example

1.3.4 GNS3

GNS3 is a software tool designed to create controlled virtual network environments, allowing users to simulate network connections and practice configurations safely and effectively. It is widely used in both educational and professional settings to learn about networking without needing physical hardware.

Thanks to integration with technologies such as Docker, VMware, and VirtualBox, GNS3 enables the emulation of both clients and servers within a virtual network environment. In this project, we will use this capability to recreate the complete network infrastructure, including terminals and essential services such as DHCP, NAT, FTP, SSH, among others.

One of GNS3's greatest advantages is its versatility. It allows direct editing of configuration files or command execution inside virtual devices, making tasks faster and more efficient. This reduces the need for lengthy setup processes and simplifies hands-on practice during network implementation.

2 General Objective

This project simulates a real-life scenario in which a functional network must be implemented for a company. Each component will play a critical role in ensuring the network operates correctly and efficiently.

- The use of a DHCP server greatly enhances the network by allowing automatic IP address assignment. This is especially useful for companies with a large number of hosts, reducing manual configuration and user interaction.
- An FTP server will enable the transfer of large volumes of data between servers. Typically, each department may have its own server, so this structure will be replicated in the project. Additionally, the use of permissions helps restrict file access to authorized users only.
- SSH service will allow secure and encrypted communication between clients and servers. It enables remote tasks such as executing scripts, checking system logs, and performing backups while protecting against potential vulnerabilities.
- Adding a web server will simulate client-server interaction, helping verify correct network configuration. A simple website will be deployed, similar to what a company might use in a real-world scenario.
- The database server will complement the rest of the services by allowing structured data storage. It may be used by the web server or to store sensitive company information that requires protection.

3 Network Topology Diagram

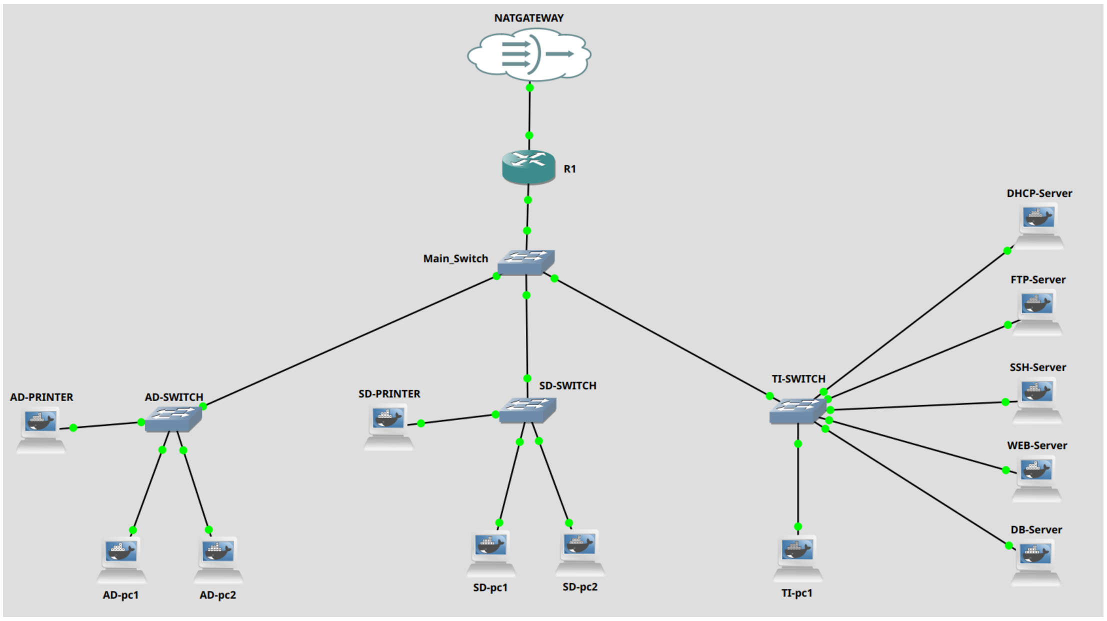


Figure 2: Network Topology

3.1 Devices used in the Topology

- Switches
- Servers
- Client

3.2 Network Topology Description

The network topology designed for this project follows an "extended star topology", which improves scalability, organization, and network performance. In this layout, each department is connected to its own local switch, and all departmental switches are then connected to a central switch, forming a hierarchical structure. The central switch is connected to the NAT gateway, which provides internet access to the entire network.

Administration Department

- Department printer (AD-PRINTER)
- Administration PC 1 (AD-pc1)

- Administration PC 2 (AD-pc2)
- Local switch (AD-SWITCH)

Sales Department

- Department printer (SD-PRINTER)
- Sales PC 1 (SD-pc1)
- Sales PC 2 (SD-pc2)
- Local switch (SD-SWITCH)

TICs Department (IT Department)

- DHCP Server (DHCP-Server)
- FTP Server (FTP-Server)
- SSH Server (SSH-Server)
- Web Server (WEB-Server)
- Database Server (DB-Server)
- TICs PC 1 (TI-pc1)
- Local switch (TI-SWITCH)

All departmental switches (AD-SWITCH, SD-SWITCH, TI-SWITCH) are connected to the **main switch**, which manages the overall network traffic and connects to the **NAT gateway** that provides internet access using a router **R1**.

3.3 IP Addressing

We used a Class C network: `192.168.40.0/24`. The IP addressing is divided into the following ranges:

- **Static Range:** `192.168.40.1 - 192.168.40.149` — manually assigned to infrastructure devices such as servers and printers.
- **DHCP Range:** `192.168.40.150 - 192.168.40.199` — dynamically assigned to client devices by the DHCP server.

Note: Static IP addresses follow this convention:

- `.1` — NAT Gateway
- `.2 - .74` — Servers
- `.75 - .149` — Printers and other static hosts

Hostname	Static IP Address
NAT	192.168.40.1
DHCP-SERVER	192.168.40.2
FTP-SERVER	192.168.40.3
SSH-SERVER	192.168.40.4
WEB-SERVER	192.168.40.5
DB-SERVER	192.168.40.6
AD-PRINTER	192.168.40.30
SD-PRINTER	192.168.40.31

Table 1: Hostnames and assigned static IP addresses

All hosts use the same default gateway for external access, which is handled via the NAT device.

Gateway: `192.168.40.1`

4 Service Implementation

Before implementing any of the services, it is necessary to configure a `Dockerfile` to automate the creation of the machine, which will then be imported into GNS3.

4.1 DHCP Server Configuration

1. First, we must configure the DHCP server interface:

   ```
   $ echo 'INTERFACESv4="eth0"' > /etc/default/isc-dhcp-server
   ```

2. Then, we will create or edit the main configuration file **dhcpd.conf**:

   ```
   $ sudo vim /etc/dhcp/dhcpd.conf
   ```

3. Now we need to expose the port used to receive DHCP requests and start the service:

   ```
   $ EXPOSE 67/udp
   ```

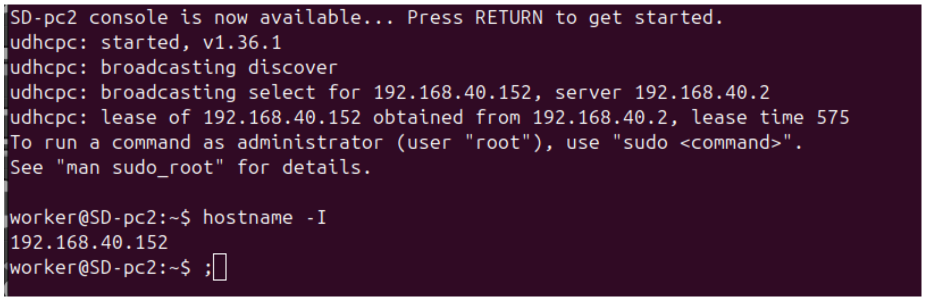
   ```
   $ service isc-dhcp-server start
   ```

The configuration file **dhcpd.conf** should contain the following:

```
# DHCP configuration for the network 192.168.40.0/24
# Static IPs reserved from 192.168.40.1 to 192.168.40.149
# 600 = 1 hour
# The ip of this Server is: 192.168.40.2

default-lease-time 600; #Time to keep this ip by default

max-lease-time 7200; #Maximum time for use ip

```

```
authoritative; #DEFINE the main DHCP for the network (this
    machine)

subnet 192.168.40.0 netmask 255.255.255.0 {
  range 192.168.40.150 192.168.40.199;  # Dynamic IP range
      (50 IPs)
  option routers 192.168.40.1;          # Default gateway (
      NAT server)
  option domain-name-servers 8.8.8.8, 8.8.4.4;
}
```

After applying the configuration, we only need to power on a machine in the network and verify that it has automatically received an IP address.



```
SD-pc2 console is now available... Press RETURN to get started.
udhcpc: started, v1.36.1
udhcpc: broadcasting discover
udhcpc: broadcasting select for 192.168.40.152, server 192.168.40.2
udhcpc: lease of 192.168.40.152 obtained from 192.168.40.2, lease time 575
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

worker@SD-pc2:~$ hostname -I
192.168.40.152
worker@SD-pc2:~$ ;
```

Figure 3: Host assigned an IP address by the DHCP server

4.2 FTP Server Configuration

1. First, we must configure the FTP configuration file by adding or modifying the following settings (typically in `/etc/vsftpd.conf`):

```
# Allow write access for local users
write_enable=YES

# Enable login for local system users
local_enable=YES

# Jail local users in their home directories
chroot_local_user=YES

# Enable listening on IPv4
listen=YES

# Disable listening on IPv6
listen_ipv6=NO

# Allow writing inside the chroot jail
```

```
allow_writeable_chroot=YES

# Enable anonymous access
anonymous_enable=YES

# Root directory for anonymous users
anon_root=/home/externalusers

# Anonymous users can't uploading files
anon_upload_enable=NO

# Anonymous users can't creating directories
anon_mkdir_write_enable=NO

# Anonymous users can't renaming or deleting files
anon_other_write_enable=NO
```

2. Next, we create the directory assigned for anonymous users and set it to read-only:

   ```
   $ mkdir -p /home/externalusers && chmod 555 /home/externalusers
   ```

3. Optionally, we can write a `README.txt` file inside the folder to include information that users will see or download.

4. To enable the FTP service, we must expose the necessary ports:

   - **Port 20:** Used for data transfer.
   - **Port 21:** Used for control (commands like `ls`, `login`, etc.).

   ```
   $ EXPOSE 20 21
   ```

5. Finally, start the FTP service:

   ```
   $ service vsftpd start
   ```

To test the server, we can use a client machine with the FTP package installed and follow these steps:

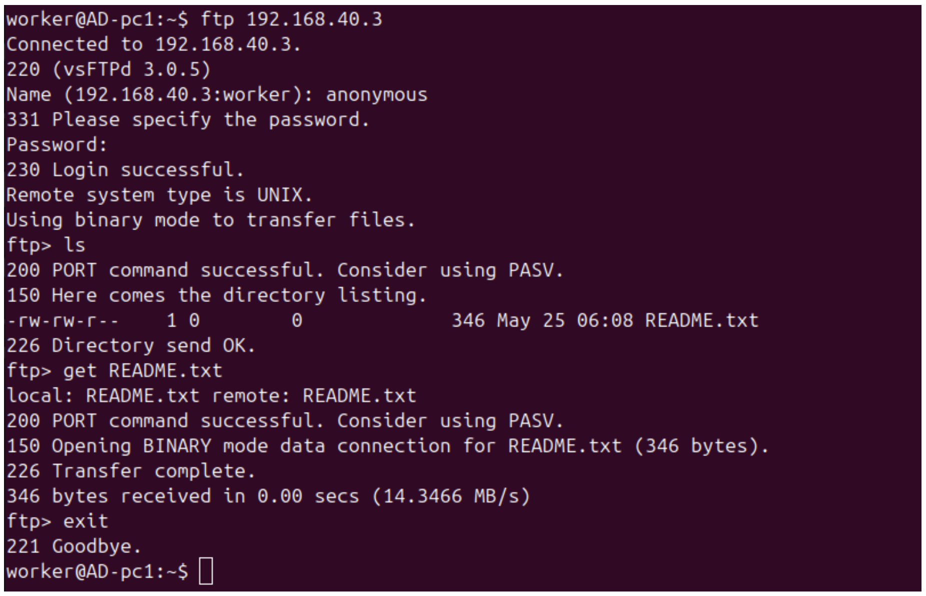
1. Connect to the server using its IP address (`192.168.40.3`):

   ```
   $ ftp <server_ip>
   ```

2. When prompted for a username, enter `anonymous`. When prompted for a password, simply press `ENTER`.

3. To download a file:

   ```
   ftp> get <file_name>
   ```
   (The file will be downloaded to the client's current working directory.)



```
worker@AD-pc1:~$ ftp 192.168.40.3
Connected to 192.168.40.3.
220 (vsFTPd 3.0.5)
Name (192.168.40.3:worker): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rw-rw-r--    1 0        0             346 May 25 06:08 README.txt
226 Directory send OK.
ftp> get README.txt
local: README.txt remote: README.txt
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for README.txt (346 bytes).
226 Transfer complete.
346 bytes received in 0.00 secs (14.3466 MB/s)
ftp> exit
221 Goodbye.
worker@AD-pc1:~$ 
```

Figure 4: User accessing the FTP server and downloading a file

4.3 SSH Server Configuration

4.3.1 SSH Configuraion

1. First, we need to create a user for SSH access using a username and password. In this case, the username will be `admin`, and the password will be `sshp`.

   ```
   $ useradd -m -s /bin/bash admin && echo "admin:sshp" | chpasswd
   && adduser admin sudo
   ```

2. Next, create the required directory for the SSH service:

   ```
   $ mkdir -p /var/run/sshd
   ```

3. Now, edit the SSH configuration file:

   ```
   $ vim /etc/ssh/sshd_config
   ```

4. Modify the following line to enable root login with a password:

   ```
   # Change this line:
   PermitRootLogin prohibit-password

   # To this:
   PermitRootLogin yes
   ```

5. To enable the SSH service, expose port 22:

   ```
   $ EXPOSE 22
   ```

6. Finally, start the SSH service:

   ```
   $ service ssh start
   ```

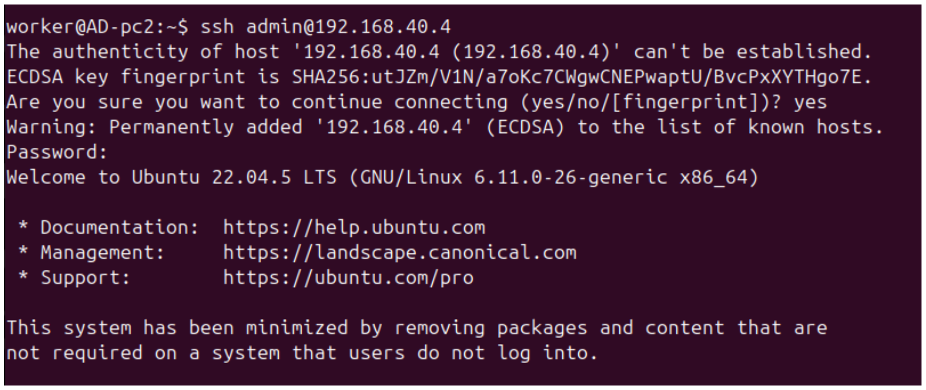
To test the connection to the SSH server, follow these steps:

1. Connect to the server using its IP address (`192.168.40.3`) and the credentials created in step 1:

   - **Username:** admin
   - **Password:** sshp

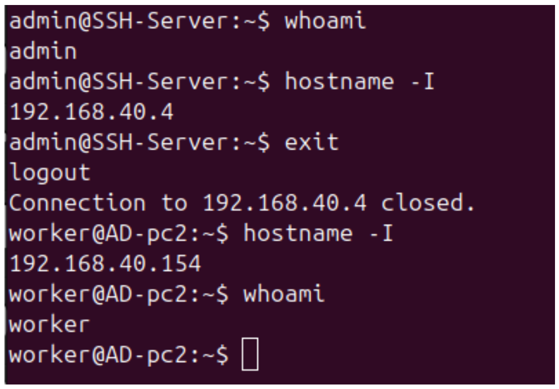
   ```
   $ ssh <username>@<server_ip>
   ```

2. Once connected, you will be inside the server and able to execute commands directly from the remote session.



```
worker@AD-pc2:~$ ssh admin@192.168.40.4
The authenticity of host '192.168.40.4 (192.168.40.4)' can't be established.
ECDSA key fingerprint is SHA256:utJZm/V1N/a7oKc7CWgwCNEPwaptU/BvcPxXYTHgo7E.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.40.4' (ECDSA) to the list of known hosts.
Password:
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.11.0-26-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.
```

Figure 5: User accessing the server via SSH as `admin`



```
admin@SSH-Server:~$ whoami
admin
admin@SSH-Server:~$ hostname -I
192.168.40.4
admin@SSH-Server:~$ exit
logout
Connection to 192.168.40.4 closed.
worker@AD-pc2:~$ hostname -I
192.168.40.154
worker@AD-pc2:~$ whoami
worker
worker@AD-pc2:~$ 
```

Figure 6: Verification

4.3.2 SSH Public and Private Key Authentication

We use a public-private key pair to enable passwordless access to the server. The main idea is to generate the key pair on the host machine, and then send the public key to the server, where it is added to the list of authorized keys.

1. First, we need to configure the `sshd_config` file on the server:

```
# SSH listens on port 22
Port 22

# Disable direct login as root user
PermitRootLogin no

# Do not allow login using a password
PasswordAuthentication no

# Enable login using SSH public/private key pairs
PubkeyAuthentication yes

# Use PAM for additional login control
UsePAM yes

# Configure SFTP to use the sftp-server binary for file
    transfers over SSH
Subsystem sftp /usr/lib/openssh/sftp-server
```

2. After making the changes, we should restart the service to apply them.

```
$ service ssh restart
```

From any host that we want to connect to the server, we should follow these steps:

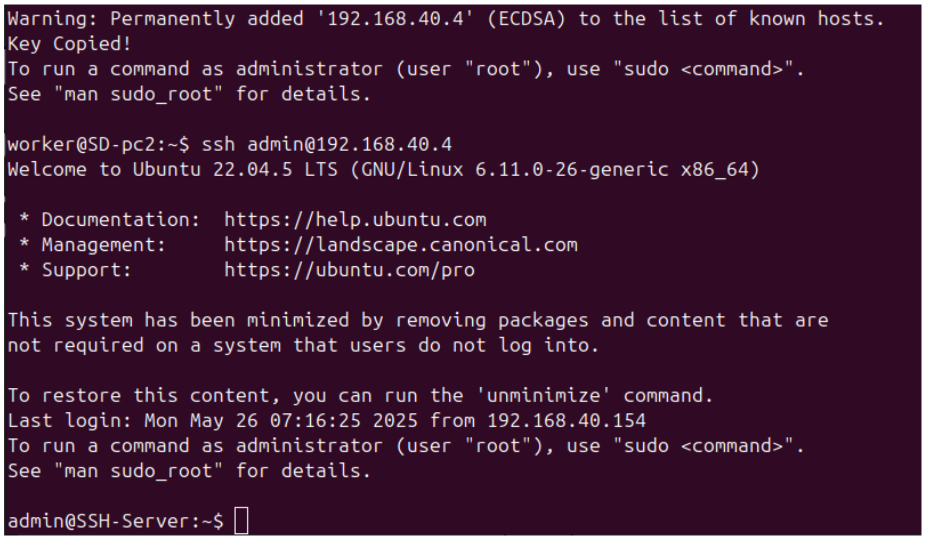
1. First, we need to generate the SSH key pair:

```
# Create .ssh folder if it doesn't exist
mkdir ~/.ssh

# Generate a 4096-bit RSA key pair (-q and -N '' are
    just output settings)
# Private key called "id_rsa" and public key called "
    id_rsa.pub"
ssh-keygen -t rsa -b 4096 -f ~/.ssh/id_rsa -q -N ''

# Set strict permissions to protect the private key and
    SSH folder
chmod 700 ~/.ssh            # (user rwx)
chmod 600 ~/.ssh/id_rsa     # (user rw)
chmod 644 ~/.ssh/id_rsa.pub # (user rw, others r)
```

2. After generating the keys, we need to send the public key to the server. This adds the key to the server's authorized keys file and allows future logins without a password.
   Note: The server password is still required the first time to send the key.

   `$ ssh-copy-id username@<ip_server>`

3. Finally, we should be able to access the server without entering a password.



```
Warning: Permanently added '192.168.40.4' (ECDSA) to the list of known hosts.
Key Copied!
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

worker@SD-pc2:~$ ssh admin@192.168.40.4
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.11.0-26-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Mon May 26 07:16:25 2025 from 192.168.40.154
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

admin@SSH-Server:~$
```

Figure 7: User accessing the server via SSH as `admin`, without a password

4.4 Web Server Configuration

1. First, we need to copy all web files to the directory `/var/www/html`, which is the default root directory for Apache. **Note:** The folder `web_page` contains the web files.

   ```
   $ mv  /web_page/index.html /var/www/html/index.html
   ```

   ```
   $ mv  /web_page/style.css /var/www/html/style.css
   ```

2. Next, we must set the correct permissions so that users can access the folder and its content:

   ```
   $ chmod -R 755 /var/www/html
   ```

3. Expose port 80, which is the default for HTTP traffic:

   ```
   $ EXPOSE 80
   ```

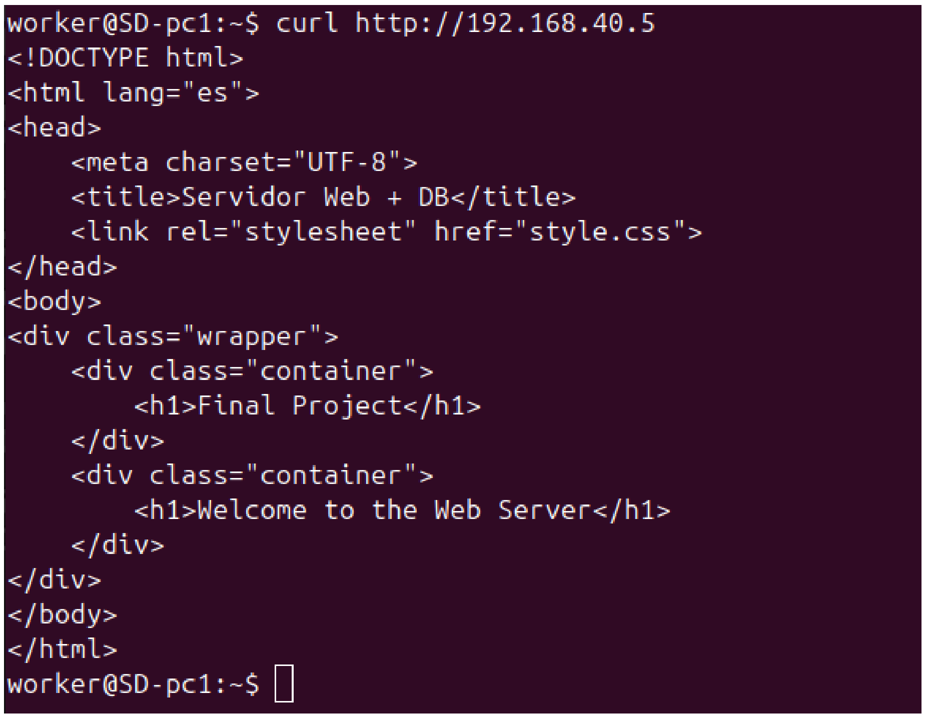
4. Finally, start the Apache web service:

   ```
   $ service apache2 start
   ```

After completing these steps, we can test the server by connecting to its IP address (`192.168.40.5`) using the `curl` command or by opening a browser:

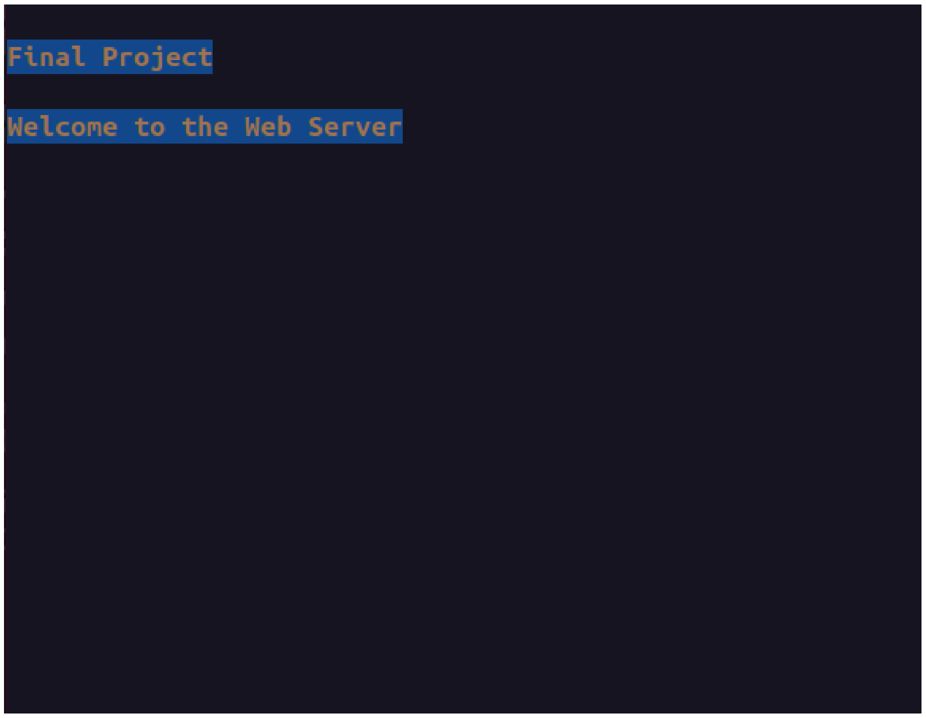
- **Using curl:** `$ curl http://<server_ip>`
- **Using a browser (lynx):** `$ lynx http://<server_ip>`

If you use `curl`, the raw HTML content of the `index.html` file will be displayed in the terminal. If you use a browser, the visual webpage will be rendered.



```
worker@SD-pc1:~$ curl http://192.168.40.5
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Servidor Web + DB</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
<div class="wrapper">
    <div class="container">
        <h1>Final Project</h1>
    </div>
    <div class="container">
        <h1>Welcome to the Web Server</h1>
    </div>
</div>
</body>
</html>
worker@SD-pc1:~$ 
```

Figure 8: Testing connection to the web server using the **curl** method



```
Final Project

Welcome to the Web Server
```

Figure 9: Testing connection to the web server using the **Lynx** browser

5 Database Server Configuration

5.1 Start the database

We use the official `mysql:8.0` Docker image to run the database server. This makes the setup easier and more portable.

Configuration Steps:

1. Create a Dockerfile or use a simple `docker run` command:

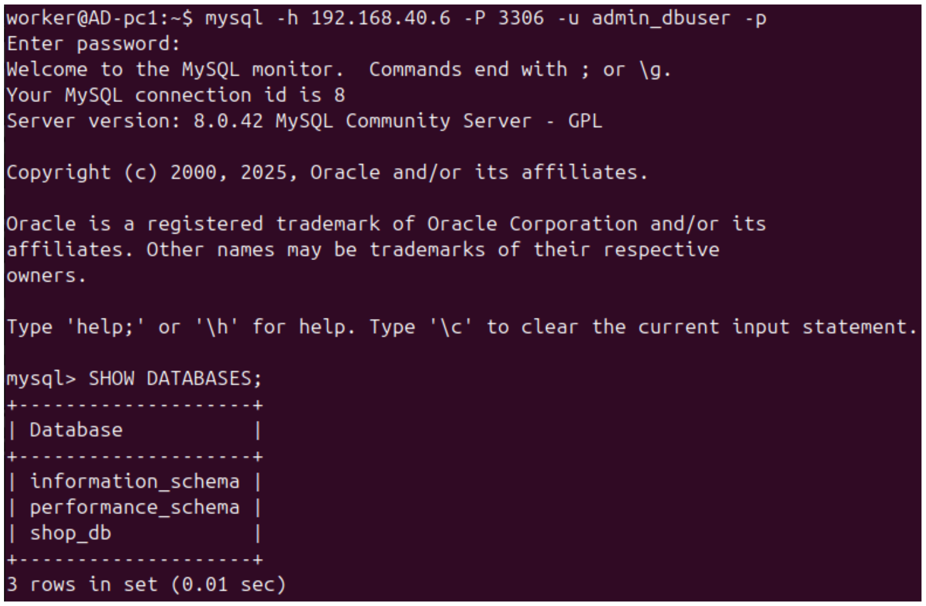
```
docker run -d \
  --name mysql_server \
  -e MYSQL_ROOT_PASSWORD=dbp \
  -e MYSQL_USER=admin_dbuser \
  -e MYSQL_PASSWORD=dbp \
  -e MYSQL_DATABASE=shop_db
  -p 3306:3306 \
  mysql:8.0
```

2. **Environment variables explained:**
   - MYSQL_ROOT_PASSWORD: root password for MySQL
   - MYSQL_DATABASE: the default database created at startup
   - MYSQL_USER: the user you want to create
   - MYSQL_PASSWORD: the password for that user
3. **To access the database from another PC:**
   - Use a MySQL client
   - Set the host IP to the address of the Database Server (192.168.40.6)
   - Port: 3306
   - Username: admin_dbuser
   - Password: dbp
   - Database: servers_credentials_db

   Now, if we want to connect to the database server, we just need to run:

   ```
   $ mysql -h 192.168.40.6 -P 3306 -u admin_dbuser -p
   ```

   After that, we should enter the database password, and we're in.



```
worker@AD-pc1:~$ mysql -h 192.168.40.6 -P 3306 -u admin_dbuser -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.42 MySQL Community Server - GPL

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| performance_schema |
| shop_db            |
+--------------------+
3 rows in set (0.01 sec)
```

Figure 10: Accessing the database using the MySQL service from another host

5.2 Database Used

We use a simple database that includes a list of buyers and the products they purchased. Each buyer is associated with an ID card number and a referenced product.

Listing 1: Physical Model

```
-- Create database
CREATE DATABASE IF NOT EXISTS shop_db;
USE shop_db;

-- Products table
CREATE TABLE products (
    id INT AUTO_INCREMENT PRIMARY KEY,
    product_name VARCHAR(100) NOT NULL
);

-- Buyers table
CREATE TABLE buyers (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    id_card VARCHAR(20) NOT NULL UNIQUE,
    product_id INT,
    FOREIGN KEY (product_id) REFERENCES products(id)
);

-- Insert products
INSERT INTO products (product_name) VALUES
('Laptop'), ('Headphones'), ('Smartphone');

-- Insert buyers
INSERT INTO buyers (name, id_card, product_id) VALUES
('Carlos P rez', '0102030405', 1),
('Ana G mez', '0607080910', 3),
('Luis Torres', '1122334455', 2);
```

6 NAT Configuration

6.1 Router Configuration

For the network to have access to the internet, we need to configure a NAT module. To do this, we must use a router. In our case, we used a Cisco C7200 router.

We connected the main switch (remember that we previously separated a gateway IP for this purpose) to the interface `FastEthernet 0/0`, and the NAT module to the interface `FastEthernet 1/0`.

To begin the configuration, we start the router and execute the following commands:

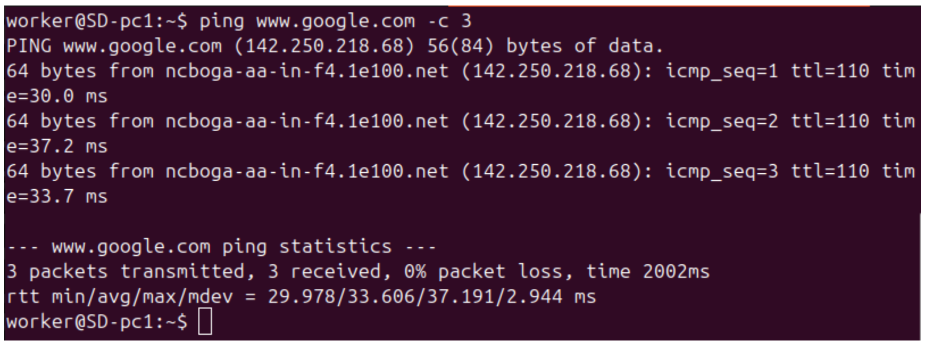
```
-- This command shows the state and information of each
    router interface
R1# show ip interface brief

-- Enter terminal configuration mode
R1# config terminal

-- Select the interface connected to the NAT module
R1(config)# interface FastEthernet 1/0

-- Request an IP via DHCP from the real network and set it
    as the NAT outside
R1(config-if)# ip address dhcp
R1(config-if)# ip nat outside

-- Activate the interface and exit configuration mode
R1(config-if)# no shutdown
R1(config-if)# exit

-- Now select the interface connected to the main switch
R1(config)# interface FastEthernet 0/0

-- Assign the reserved gateway IP and mask, and set it as
    the NAT inside
R1(config-if)# ip address 192.168.40.1 255.255.255.0
R1(config-if)# ip nat inside

-- Activate the interface and exit configuration mode
R1(config-if)# no shutdown
R1(config-if)# exit

-- Permit any IP traffic through an access list
R1(config)# access-list 1 permit any

-- Configure NAT to use the outside interface and overload (
    PAT)
R1(config)# ip nat inside source list 1 interface
    FastEthernet 1/0 overload

-- Enable domain name resolution to avoid DNS issues
R1(config)# ip domain-lookup

-- Exit to privileged EXEC mode
R1(config)# end

-- Test the connection
R1# ping google.com
```

```
-- or
R1# ping 8.8.8.8
```

6.2 Test Internet conecction

Remember that in the net we configure a DHCP server can assign a ip automatic for each pc oin the network, so yo can anter in a pc and probe:

`$ ping www.google.com -c 3`



```
worker@SD-pc1:~$ ping www.google.com -c 3
PING www.google.com (142.250.218.68) 56(84) bytes of data.
64 bytes from ncboga-aa-in-f4.1e100.net (142.250.218.68): icmp_seq=1 ttl=110 tim
e=30.0 ms
64 bytes from ncboga-aa-in-f4.1e100.net (142.250.218.68): icmp_seq=2 ttl=110 tim
e=37.2 ms
64 bytes from ncboga-aa-in-f4.1e100.net (142.250.218.68): icmp_seq=3 ttl=110 tim
e=33.7 ms

--- www.google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 29.978/33.606/37.191/2.944 ms
worker@SD-pc1:~$ 
```

Figure 11: Testing internet connectivity by pinging **Google**

7 User and Group Management

For better management of each server, we must configure users and groups. This way, we can have more control over who connects to the server and what permissions they have. Depending on the type of service implemented, some servers may only require an admin user for configuration, while others—such as the FTP server—require users grouped by permission level.

(**Note:** Each admin user has their own group, and for the SSH server, user authentication does not require group validation (4.3.2).)

7.1 Configure Admin User

To configure an admin user for each server, follow these steps:

1. First, we add the user and assign them the Bash shell:

   `$ sudo useradd -m -s /bin/bash admin`

2. Then, we assign a password (if the command did not prompt for it):

   `$ sudo passwd admin`

3. Finally, we add the user to the **sudo** group to grant administrative privileges:

   ```
   $ sudo usermod -aG sudo admin
   ```

Admin users are configured independently for each server:

Server	Username	Password
DHCP	admin	dhcpp
FTP	admin	ftpp
SSH	admin	sshp
WEB	admin	webp
DATABASE	admin	dbp

Table 2: Admin users and their passwords for each server

7.2 FTP Users

For the FTP server, we created three groups to define different permission levels and a user for each group:

- `ftpuser1`: Can only read/download files.
- `ftpuser2`: Can read and upload files.
- `ftpuser3`: Has full access: read, upload, delete.

1. Create the groups:

   ```
   $ sudo groupadd ftp_read
   ```

   ```
   $ sudo groupadd ftp_write
   ```

   ```
   $ sudo groupadd ftp_admin
   ```

2. Add the shell `/sbin/nologin` to the system so FTP users can't log in interactively:

   ```
   $ echo "/sbin/nologin" | sudo tee -a /etc/shells
   ```

3. Create each user, assigning them to their group and restricting shell access:

   ```
   $ sudo useradd -m -d /home/ftpuser1 -s /sbin/nologin -g ftp_read
   ftpuser1
   ```

   ```
   $ sudo useradd -m -d /home/ftpuser2 -s /sbin/nologin -g ftp_write
   ftpuser2
   ```

   ```
   $ sudo useradd -m -d /home/ftpuser3 -s /sbin/nologin -g ftp_admin
   ftpuser3
   ```

4. Set the passwords:

   ```
   $ echo "ftpuser1:user1p" | sudo chpasswd
   ```

   ```
   $ echo "ftpuser2:user2p" | sudo chpasswd
   ```

   ```
   $ echo "ftpuser3:user3p" | sudo chpasswd
   ```

5. Adjust the permissions of their home folders:

   - `ftpuser1`:

     ```
     $ chown root:ftp_read /home/ftpuser1
     $ chmod 750 /home/ftpuser1
     ```

   - `ftpuser2`:

     ```
     $ chown ftpuser2:ftp_write /home/ftpuser2
     $ chmod 770 /home/ftpuser2
     ```

   - `ftpuser3`:

     ```
     $ chown ftpuser3:ftp_admin /home/ftpuser3
     $ chmod 770 /home/ftpuser3
     ```

6. Create a file `vsftpd.userlist` listing all users allowed to access via FTP:

   ```
   anonymous
   ftpuser1
   ftpuser2
   ftpuser3
   ```

7. Create the anonymous download directory:

   ```
   $ sudo mkdir -p /home/externalusers && sudo chmod 755 /home/externalusers
   ```

8. Edit the configuration file `/etc/vsftpd.conf` to reflect user/group access:

   ```
   # Listen IPv4
   listen=YES
   # Don't listen IPv6
   listen_ipv6=NO

   # Allow anonymous login and working folder
   anonymous_enable=YES
   anon_root=/home/externalusers

   # Allow local users and permissions like upload, delete,
       and download
   local_enable=YES
   write_enable=YES

   # Lock users in their home directory
   chroot_local_user=YES
   ```

```
# Allow writing even when chrooted
allow_writeable_chroot=YES

# Enable user allow/deny list
userlist_enable=YES
# Path to the user list file
userlist_file=/etc/vsftpd.userlist
# Only listed users can log in
userlist_deny=YES

# Anonymous users can't upload, create folders, rename/
    delete/etc.
anon_upload_enable=NO
anon_mkdir_write_enable=NO
anon_other_write_enable=NO

# Enable transfer logs
xferlog_enable=YES
# Use server's local time in logs
use_localtime=YES
# Show directory message on login
dirmessage_enable=YES
# Use port 20 for data transfers
connect_from_port_20=YES
```

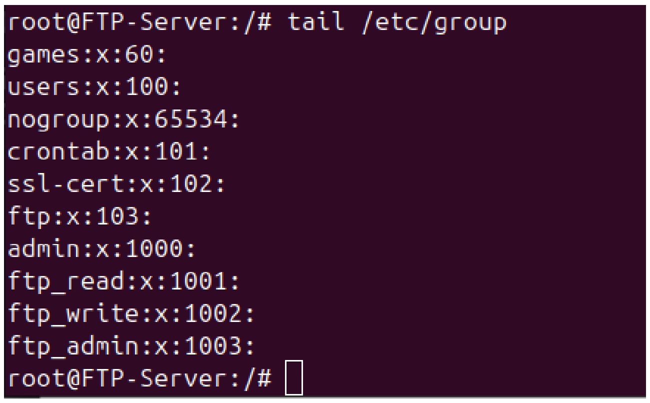
The groups and users configured for the FTP server are:

Group	**Permissions**
`ftp_read`	Download only
`ftp_write`	Download and upload
`ftp_admin`	Download, upload, and delete

Table 3: Permission levels for each FTP group

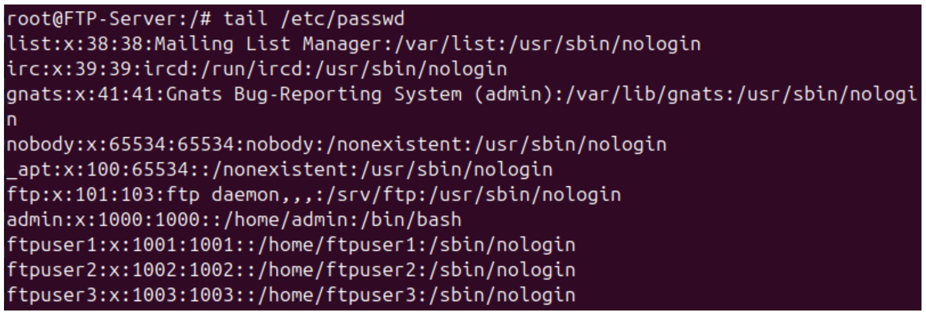
Group	**Username**	**Password**
`ftp_read`	ftpuser1	user1p
`ftp_write`	ftpuser2	user2p
`ftp_admin`	ftpuser3	user3p

Table 4: FTP users and their assigned groups



```
root@FTP-Server:/# tail /etc/group
games:x:60:
users:x:100:
nogroup:x:65534:
crontab:x:101:
ssl-cert:x:102:
ftp:x:103:
admin:x:1000:
ftp_read:x:1001:
ftp_write:x:1002:
ftp_admin:x:1003:
root@FTP-Server:/#
```

Figure 12: User groups configured on the FTP server



```
root@FTP-Server:/# tail /etc/passwd
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologi
n
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
ftp:x:101:103:ftp daemon,,,:/srv/ftp:/usr/sbin/nologin
admin:x:1000:1000::/home/admin:/bin/bash
ftpuser1:x:1001:1001::/home/ftpuser1:/sbin/nologin
ftpuser2:x:1002:1002::/home/ftpuser2:/sbin/nologin
ftpuser3:x:1003:1003::/home/ftpuser3:/sbin/nologin
```

Figure 13: User accounts configured on the FTP server

8 Shell Scripting for Automation

To facilitate some tasks on the host PCs, we created a few scripts: (**NOTE:** Remember that you should give execution permission to each `.sh` file)

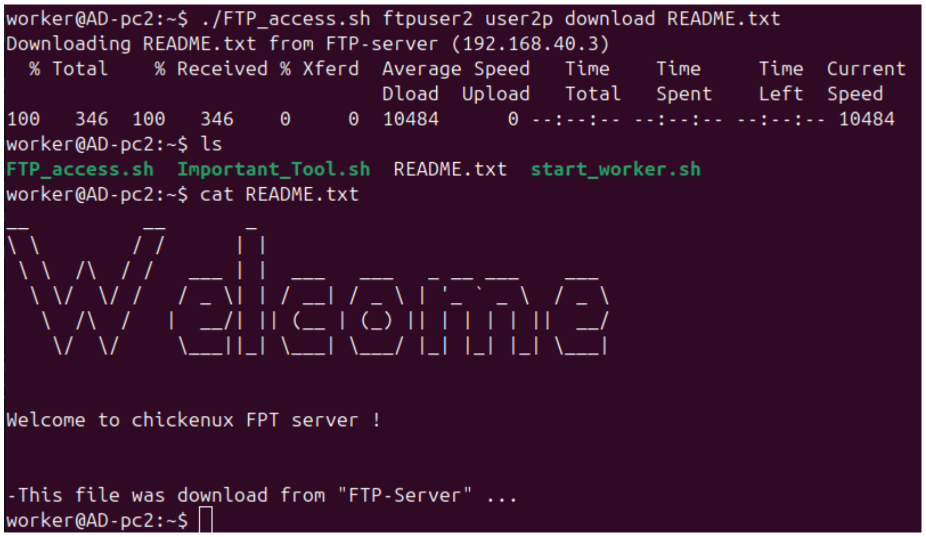
8.1 FTP Connection Script

This script handles direct access to the FTP server. It can be executed with just a username and password to list files, or with an additional command and filename to download or upload. Supported commands are `list`, `download <file>`, and `upload <file>`. The server's IP address is assumed to be as specified in Table 1.

File name: `FTP_access.sh`

```
#!/usr/bin/bash
# Daniel Troya
# In this part, we verify if the number of arguments is
   correct.
if [ $# -ne 2 ] && [ $# -ne 4 ]; then
    echo "Usage schema:"
    echo "./FTP_access.sh <username> <password> # JUST list
       files"
    echo "./FTP_access.sh <username> <password> download <
       file>"
    echo "./FTP_access.sh <username> <password> upload <file
       >"
    exit 1
fi

# Here, we assign the position of the arguments entered into
    variables to use in the program
USER="$1"
PASS="$2"
CMD="$3"
FILE="$4"

# IP of the FTP server
SERVER="192.168.40.3"

# Here, we verify the entered command and write the
   functionality for each one. If the command is wrong, we
   return an error message.
if [ $# -eq 2 ]; then
    echo "Listing files on FTP-server ($SERVER) as $USER:"
    curl -u "$USER:$PASS" ftp://$SERVER/
else
    case "$CMD" in
        download)
            echo "Downloading $FILE from FTP-server ($SERVER
               )"
            curl -u "$USER:$PASS" ftp://$SERVER/$FILE -O
            ;;
        upload)
            if [ ! -f "$FILE" ]; then
                echo "Error: File '$FILE' does not exist
                   locally."
                exit 1
            fi
            echo "Uploading $FILE to FTP server ($SERVER)...
               "
            curl -T "$FILE" -u "$USER:$PASS" ftp://$SERVER/
            ;;
        *)
            echo "Invalid command: $CMD"
            echo "Use: download or upload"
```

```
                ;;
        esac
fi
```



```
worker@AD-pc2:~$ ./FTP_access.sh ftpuser2 user2p download README.txt
Downloading README.txt from FTP-server (192.168.40.3)
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   346  100   346    0     0  10484      0 --:--:-- --:--:-- --:--:-- 10484
worker@AD-pc2:~$ ls
FTP_access.sh  Important_Tool.sh  README.txt  start_worker.sh
worker@AD-pc2:~$ cat README.txt
__          __        _
\ \        / /       | |
 \ \  /\  / /   ___  | |  ___   ___   _ __ ___    ___
  \ \/  \/ /   / _ \ | | / __| / _ \ | '_ ` _ \  / _ \
   \  /\  /   |  __/ | || (__ | (_) || | | | | ||  __/
    \/  \/     \___| |_| \___| \___/ |_| |_| |_| \___|


Welcome to chickenux FPT server !


-This file was download from "FTP-Server" ...
worker@AD-pc2:~$ 
```

Figure 14: Execution of the script **FTP_access.sh**

8.2 User, Group, and Permission Management Script

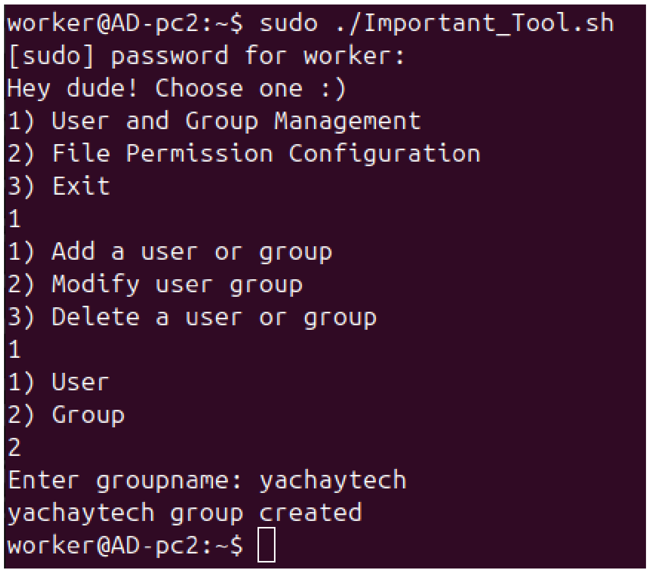
This script helps the user create users and groups, add users to groups, and more. It also allows modifying file permissions or changing the owner of a specific file.(Note: This script must be executed with superuser privileges)

File name: `Important_Tool.sh`

```
#!/usr/bin/bash
# Daniel Troya

# Create the main menu with different options
echo "Hey dude! Choose one :)"
echo "1) User and Group Management"
echo "2) File Permission Configuration"
echo "3) Exit"
read op

# We split the script into functions so it's easier to
    understand.
# First we define the functions, and then we call them in
    the main execution.
```

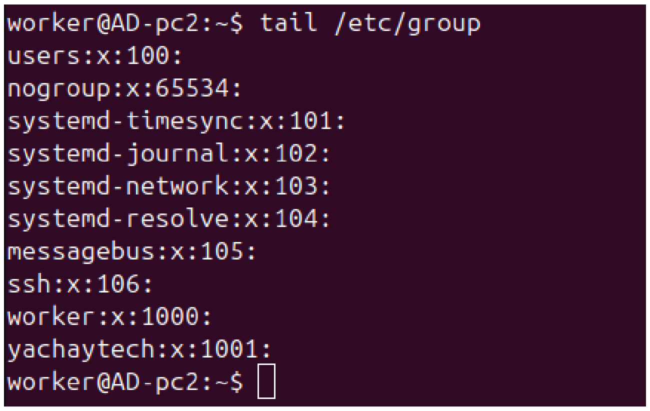
```
# ====> Option 1 <====

# This function creates a user or a group. If the option is
    invalid, it shows an error message.
add_usser_or_group(){
    echo "1) User"
    echo "2) Group"
    read addchoice
    case $addchoice in
        1)
            read -p "Enter username: " name
            sudo adduser $name
            echo "$name user created"
            ;;
        2)
            read -p "Enter groupname: " name
            sudo groupadd $name
            echo "$name group created"
            ;;
        *)
            echo "Error: Wrong option"
            ;;
    esac
}

# This function adds a user to a specific group.
Modify(){
    read -p "Enter username: " uname
    read -p "Enter groupname: " gname
    sudo usermod -g $uname $gname
    echo "$uname added to $gname"
}

# This function deletes a user or a group. If the choice is
    invalid, it shows an error.
Delete(){
    echo "1) User"
    echo "2) Group"
    read addchoice
    case $addchoice in
        1)
            read -p "Enter username: " name
            sudo userdel -r $name
            echo "$name deleted"
            ;;
        2)
            read -p "Enter groupname: " name
            sudo groupdel $name
            echo "$name group deleted"
            ;;
        *)
            echo "Error: Wrong option"
            ;;
    esac
}

# ====> Option 2 <====

# This function asks for a file or folder and changes its
    permission (read/write/execute).
Chanperm(){
    read -p "Enter a file path: " pth
    read -p "Enter a permission number: " perm
    chmod $perm $pth
}

# This function changes the owner and group of a file or
    folder.
Chanowner(){
    read -p "Enter a file path: " pth
    read -p "Enter the owner: " owner
    read -p "Enter the group: " grp
    chown $owner:$grp $pth
}

# MAIN FUNCTION
# Here we call all the functions based on the  u s e r s
    choice, and show error messages for invalid input.
case $op in
    1)
        echo "1) Add a user or group"
        echo "2) Modify user group"
        echo "3) Delete a user or group"
        read c1
        case $c1 in
            1) add_usser_or_group ;;
            2) Modify ;;
            3) Delete ;;
            *) echo "Error: Wrong option" ;;
        esac
        ;;
    2)
        echo "1) Change file permissions"
        echo "2) Change file ownership"
        read c2
        case $c2 in
            1) Chanperm ;;
            2) Chanowner ;;
            *) echo "Error: Wrong option" ;;
        esac
```

```
            ;;
      3)
            exit 0
            ;;
      *)
            echo "Error: Invalid option"
            ;;
esac
```



```
worker@AD-pc2:~$ sudo ./Important_Tool.sh
[sudo] password for worker:
Hey dude! Choose one :)
1) User and Group Management
2) File Permission Configuration
3) Exit
1
1) Add a user or group
2) Modify user group
3) Delete a user or group
1
1) User
2) Group
2
Enter groupname: yachaytech
yachaytech group created
worker@AD-pc2:~$
```

Figure 15: Execution of the script **Important_Tool.sh**



```
worker@AD-pc2:~$ tail /etc/group
users:x:100:
nogroup:x:65534:
systemd-timesync:x:101:
systemd-journal:x:102:
systemd-network:x:103:
systemd-resolve:x:104:
messagebus:x:105:
ssh:x:106:
worker:x:1000:
yachaytech:x:1001:
worker@AD-pc2:~$
```

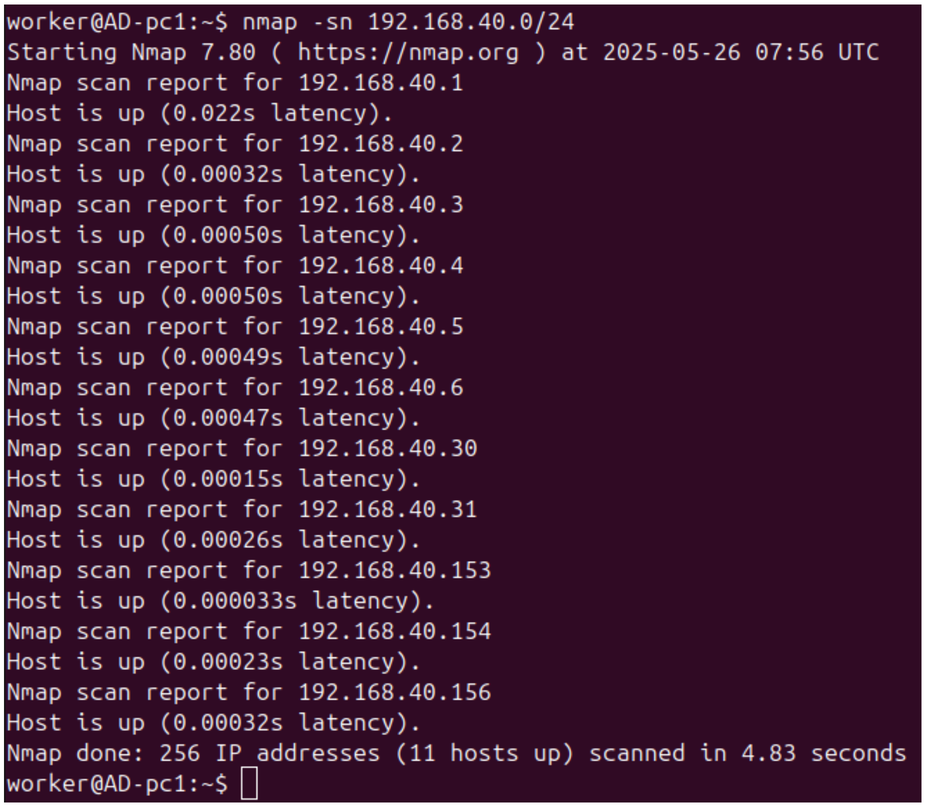
Figure 16: Verification of group creation

9 Vulnerability Testing with Nmap

Using `nmap`, we started by detecting all the devices connected to the network:

- **Discover hosts in the entire network:**

  ```
  $ nmap -sn <network_ip>/<mask>
  ```



```
worker@AD-pc1:~$ nmap -sn 192.168.40.0/24
Starting Nmap 7.80 ( https://nmap.org ) at 2025-05-26 07:56 UTC
Nmap scan report for 192.168.40.1
Host is up (0.022s latency).
Nmap scan report for 192.168.40.2
Host is up (0.00032s latency).
Nmap scan report for 192.168.40.3
Host is up (0.00050s latency).
Nmap scan report for 192.168.40.4
Host is up (0.00050s latency).
Nmap scan report for 192.168.40.5
Host is up (0.00049s latency).
Nmap scan report for 192.168.40.6
Host is up (0.00047s latency).
Nmap scan report for 192.168.40.30
Host is up (0.00015s latency).
Nmap scan report for 192.168.40.31
Host is up (0.00026s latency).
Nmap scan report for 192.168.40.153
Host is up (0.000033s latency).
Nmap scan report for 192.168.40.154
Host is up (0.00023s latency).
Nmap scan report for 192.168.40.156
Host is up (0.00032s latency).
Nmap done: 256 IP addresses (11 hosts up) scanned in 4.83 seconds
worker@AD-pc1:~$
```

Figure 17: Discovered hosts in the **192.168.40.0/24** network

- **Obtain detailed information about a specific host:**
  - **Port Scan:**

    ```
    $ nmap -p- <host_ip>
    ```

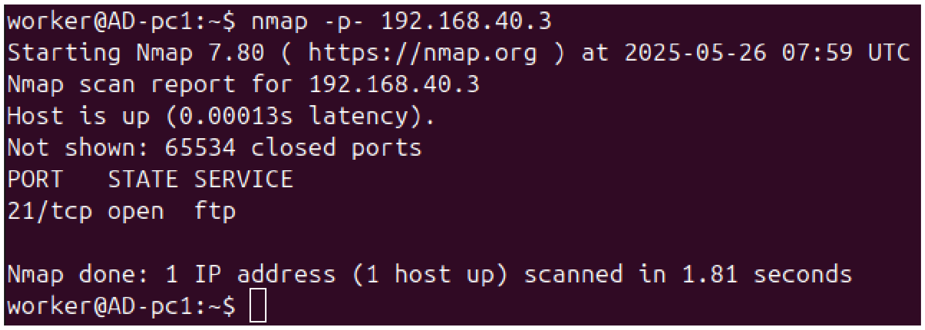

```
worker@AD-pc1:~$ nmap -p- 192.168.40.3
Starting Nmap 7.80 ( https://nmap.org ) at 2025-05-26 07:59 UTC
Nmap scan report for 192.168.40.3
Host is up (0.00013s latency).
Not shown: 65534 closed ports
PORT   STATE SERVICE
21/tcp open  ftp

Nmap done: 1 IP address (1 host up) scanned in 1.81 seconds
worker@AD-pc1:~$ 
```

Figure 18: Port scan performed on the **192.168.40.3** host

- **Operating System Detection and Service Info:**
  `$ nmap -A <host_ip>`

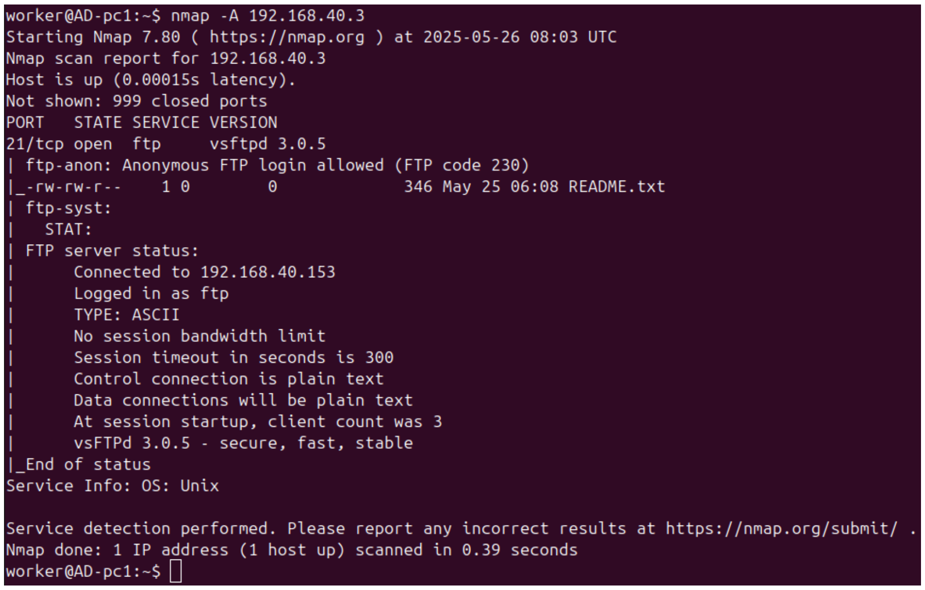
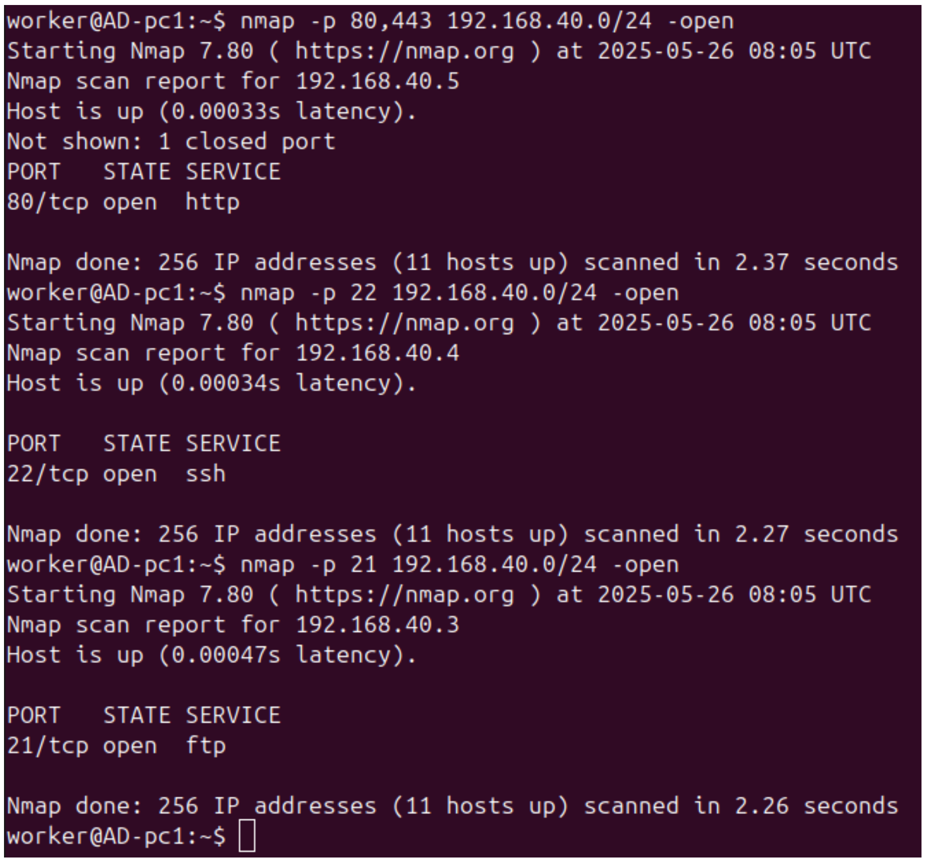

```
worker@AD-pc1:~$ nmap -A 192.168.40.3
Starting Nmap 7.80 ( https://nmap.org ) at 2025-05-26 08:03 UTC
Nmap scan report for 192.168.40.3
Host is up (0.00015s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 3.0.5
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_-rw-rw-r--    1 0        0             346 May 25 06:08 README.txt
| ftp-syst:
|   STAT:
| FTP server status:
|      Connected to 192.168.40.153
|      Logged in as ftp
|      TYPE: ASCII
|      No session bandwidth limit
|      Session timeout in seconds is 300
|      Control connection is plain text
|      Data connections will be plain text
|      At session startup, client count was 3
|      vsFTPd 3.0.5 - secure, fast, stable
|_End of status
Service Info: OS: Unix

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 0.39 seconds
worker@AD-pc1:~$ 
```

Figure 19: Operating system detected on the **192.168.40.3** host

- **Service-specific scans:** If we want to get information about specific services, we can specify the port numbers:
  - **Web Server:** `$ nmap -p 80,443 <network_ip>/<mask> -open`
  - **SSH Server:** `$ nmap -p 22 <network_ip>/<mask> -open`
  - **FTP Server:** `$ nmap -p 21 <network_ip>/<mask> -open`



```
worker@AD-pc1:~$ nmap -p 80,443 192.168.40.0/24 -open
Starting Nmap 7.80 ( https://nmap.org ) at 2025-05-26 08:05 UTC
Nmap scan report for 192.168.40.5
Host is up (0.00033s latency).
Not shown: 1 closed port
PORT   STATE SERVICE
80/tcp open  http

Nmap done: 256 IP addresses (11 hosts up) scanned in 2.37 seconds
worker@AD-pc1:~$ nmap -p 22 192.168.40.0/24 -open
Starting Nmap 7.80 ( https://nmap.org ) at 2025-05-26 08:05 UTC
Nmap scan report for 192.168.40.4
Host is up (0.00034s latency).

PORT   STATE SERVICE
22/tcp open  ssh

Nmap done: 256 IP addresses (11 hosts up) scanned in 2.27 seconds
worker@AD-pc1:~$ nmap -p 21 192.168.40.0/24 -open
Starting Nmap 7.80 ( https://nmap.org ) at 2025-05-26 08:05 UTC
Nmap scan report for 192.168.40.3
Host is up (0.00047s latency).

PORT   STATE SERVICE
21/tcp open  ftp

Nmap done: 256 IP addresses (11 hosts up) scanned in 2.26 seconds
worker@AD-pc1:~$ 
```

Figure 20: Scanning for specific services in the **192.168.40.0/24** network

9.1 Security Measures

While using `nmap`, we encountered situations where the scan reported limited or no results. This was due to the fact that some servers had been secured properly. These protections included:

- **Service Obfuscation:** Some ports were intentionally hidden or not responding.
- **Firewall/IPS:** These devices blocked scan attempts and filtered incoming requests.

This does not mean that the scan failed, but rather that some hosts on the network had defense mechanisms in place. In such cases, we can improve our scanning techniques by:

- Scanning specific ports or protocols.
- Using more advanced scan types (e.g., `-sS`, `-sV`, `-O`).
- Timing and evasion techniques to bypass detection.

10 Challenges and Troubleshooting

During the implementation of the different services, we encountered various issues that were challenging to identify and resolve. Below are the most notable ones and how we fixed them.

10.1 DHCP Configuration Issues

When the DHCP server was implemented and tested, it appeared to be correctly configured, but it was not assigning IP addresses to hosts in the network. After reviewing the configuration file `dhcpd.conf`, everything seemed correct. However, the issue was that we had not restarted the DHCP service after making changes. The problem was resolved by simply restarting the service with:

```
$ service isc-dhcp-server start
```

10.2 FTP Configuration Issues

The FTP server was the most problematic to configure. We attempted to define specific users allowed to access the server. Initially, we tried enabling access for the `anonymous` user, but we kept receiving access errors.

After rechecking the `vsftpd.conf` file, we confirmed that anonymous access was enabled. However, upon reviewing the **system logs**, we noticed that the server was not recognizing the connection attempt when entering the username. The root of the problem was in the user list restrictions. Specifically, we had enabled the following configuration:

```
# Enable user allow/deny list
userlist_enable=YES
# Path to the user list file
userlist_file=/etc/vsftpd.userlist
```

By default, the FTP service does not allow the `anonymous` user if the user list is active. Therefore, we needed to explicitly include `anonymous` in the file `/etc/vsftpd.userlist` to grant access.

10.3 SSH Configuration Issues

In the SSH server, we encountered permission-related problems. When trying to access the web server via the `curl` command, the request was denied. Upon investigation, we found that the issue was due to incorrect folder permissions—the directory `/var/www/html` was only accessible to the `root` user.

The problem was easily resolved by changing the directory permissions to make it accessible to other users:

```
$ chmod -R 755 /var/www/html
```

(**Note**: With `755`, the group and others are allowed to read and execute the web folder, but not write to it.)

11 Conclusions

In general, we implemented each service in a controlled environment for security reasons. We noticed that each server has a different configuration style from the location of configuration files to the types of protocols used. It was interesting to realize that we could also identify the vulnerabilities of each server, precisely because we configured them ourselves.

Permissions played a key role in the functionality of every service. Depending on the permissions set, users could perform different actions or be restricted from accessing certain parts of the system. This highlights how powerful and critical permission configuration can be, as it directly controls the level of access available to users.

The next potential step would be to implement more advanced security mechanisms such as firewalls, IP tables with restrictions, and user roles with specific permissions to manage servers. In real-world scenarios, these servers may store large amounts of sensitive data or run essential services for businesses. Losing control over them, or being attacked by hackers, could lead to a **CRITICAL ERROR** and a serious security incident, especially if user data is compromised.

In summary, the implementation and management of servers is a crucial task in the real world. Much of the internet relies on properly configured services like these, and many of them are implemented on Linux due to its versatility and strong security. Ultimately, everything depends on what you want to achieve and how securely you choose to implement it.

Appendices

If you want to consult any configuration file used in this project, you can find them in the main folder, where each virtual machine is created through Docker. You need to be aware that:

- **Dockerfile:** Used to create each host. You will also find useful command examples, most of which are commented to make the file easier to read.

- **Bash Scripts:** Each host has a different bash script, depending on the services implemented. All scripts are commented to help you understand their purpose and functionality.

- **Configuration Files:** Remember that each service has its own configuration file (usually ending in `.conf`) to modify default settings according to our requirements.
- **Other Files:** There may be other files not mentioned above. These are typically unique to each host's folder or Docker build context. These files are usually imported or referenced from the Dockerfile, so we recommend starting by reviewing the `Dockerfile` for each host.

References

[1] A. AbdulGhaffar, S. K. Paul, and A. Matrawy, "An analysis of dhcp vulnerabilities, attacks, and countermeasures," in *2023 Biennial Symposium on Communications (BSC)*, 2023, pp. 119–124.

- Microsoft Docs — DHCP Overview: https://learn.microsoft.com/en-us/windows-server/networking/technologies/dhcp/dhcp-top
- Hostinger — What is FTP: https://www.hostinger.com/tutorials/what-is-ftp
- Wikipedia — HTTPS: https://en.wikipedia.org/wiki/HTTPS
- UpGuard — File Transfer Vulnerabilities: https://www.upguard.com/blog/file-transfer
- Microsoft Docs — What is SQL Server: https://learn.microsoft.com/en-us/sql/sql-server/what-is-sql-server?view=sql-server-ver16
- DataSunrise — 10 Common Database Vulnerabilities: https://www.datasunrise.com/potential-db-threats/10-common-vulnerabilities/
- Cloudflare — What is SSH: https://www.cloudflare.com/learning/access-management/what-is-ssh/
- CSIRT Bolivia — Vulnerability in SSH: https://www.csirt.gob.bo/es/avisos-de-seguridad/vulnerabilidad-en-ssh-permite-los-atacantes-tomar-el-control-del-servidor
- GNS3 Documentation: https://docs.gns3.com/docs/
- Docker Docs — Overview: https://docs.docker.com/get-started/docker-overview/
- Docker — What is a Container: https://www.docker.com/resources/what-container/

- Ubuntu Server Documentation:
  https://documentation.ubuntu.com/server/
- Nmap — Official Documentation:
  https://nmap.org/book/man.html#man-description
- Cisco Networking Topologies:
  https://ccnadesdecero.es/topologias-red-lan-y-wan/
- DigitalOcean — SSH Key-Based Authentication:
  https://www.digitalocean.com/community/tutorials/how-to-configure-ssh-key-based-authentication-on-a-linux-server
- NAT Virtual Interface Explanation:
  https://networklessons.com/ip-services/nat-virtual-interface
- Cisco — IOS Basic Commands:
  https://www.cisco.com/E-Learning/bulk/public/tac/cim/cib/using_cisco_ios_software/07_basic_commands_tasks.htm