Linux Security Hardening Project – By Priant Singh

This is a small security project I did on my Ubuntu virtual machine to make it safer. I used simple Linux commands to update the system, create a safer user, secure SSH, set up a firewall, install Fail2Ban, and check open ports. Here is everything I did and what each command means.

1. Updated and Upgraded the System

Command:

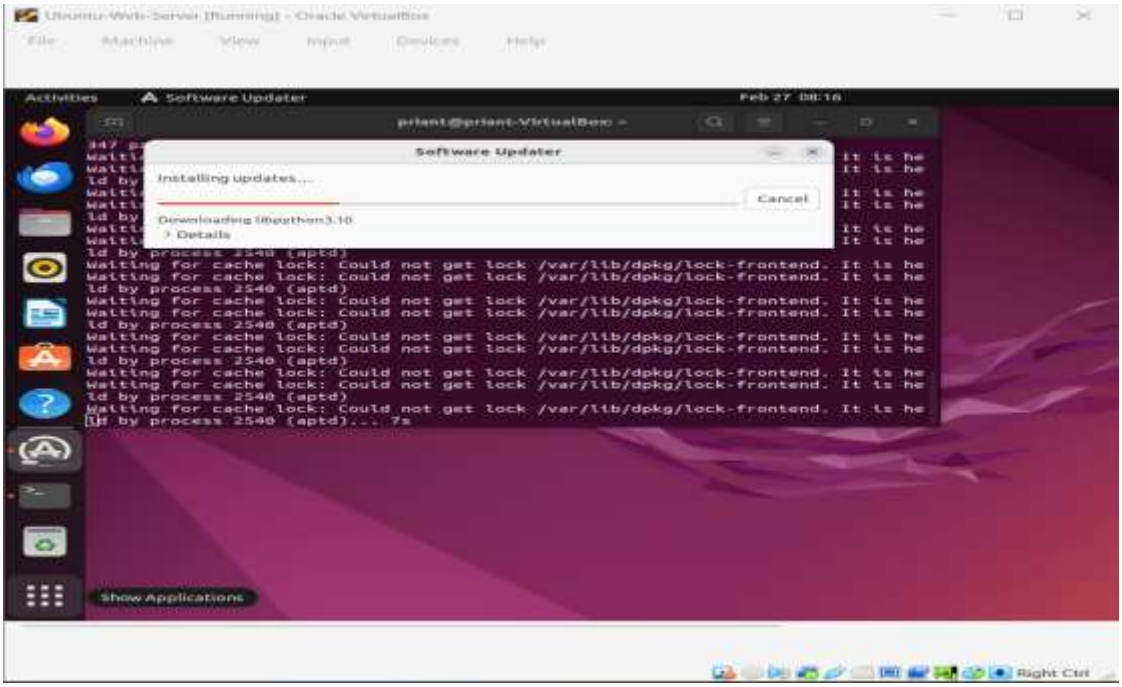
sudo apt update && sudo apt upgrade -y

What it does (simple explanation):

- **apt update** checks for the latest updates available.
- **apt upgrade** installs the updates.
- **sudo** runs the command as admin.
- **-y** automatically answers “yes”.

Why I did this:

Keeping the system updated is one of the easiest and most important security steps because it fixes known vulnerabilities.



2. Created a New Non-Root User (Least Privilege)

Commands:

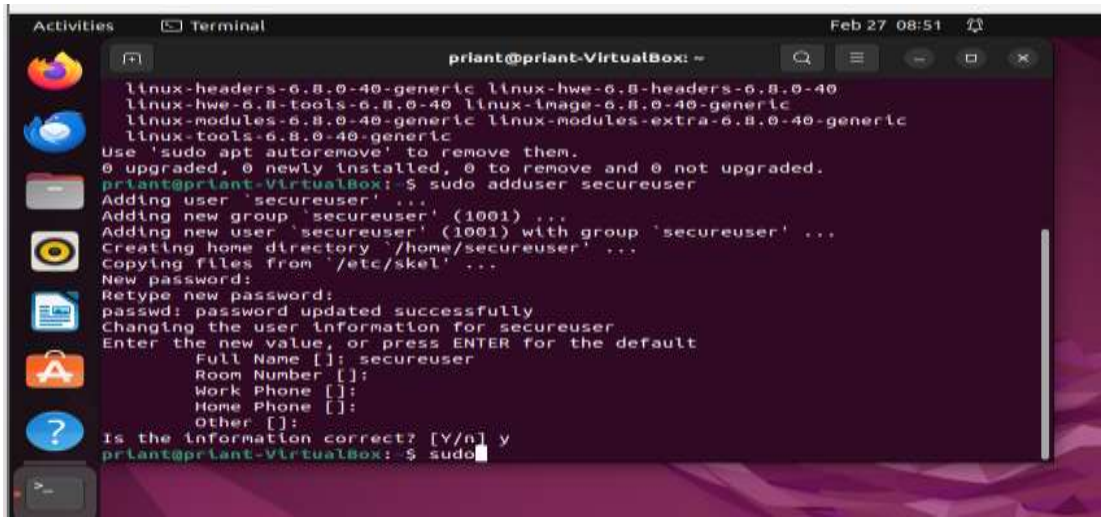
sudo adduser secureuser
sudo usermod -aG sudo secureuser

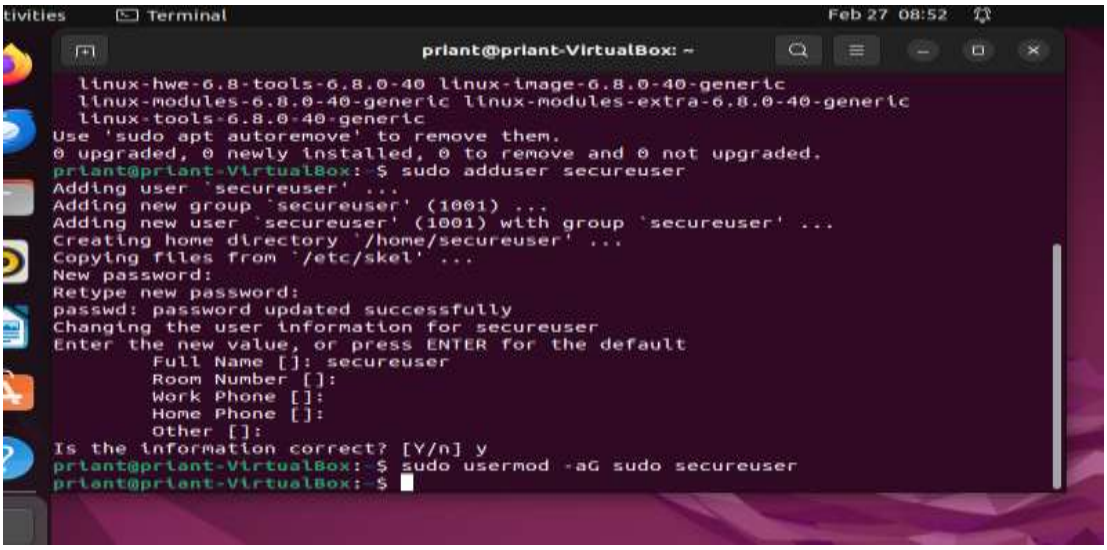
What they mean:

- **adduser secureuser** creates a new user named “secureuser”.
- **usermod -aG sudo secureuser** adds that user to the sudo (admin) group.

Why I did this:

It is unsafe to use the root account for everything. Creating a normal user and only using sudo when needed reduces risk.





3. Installed OpenSSH Server

Command:

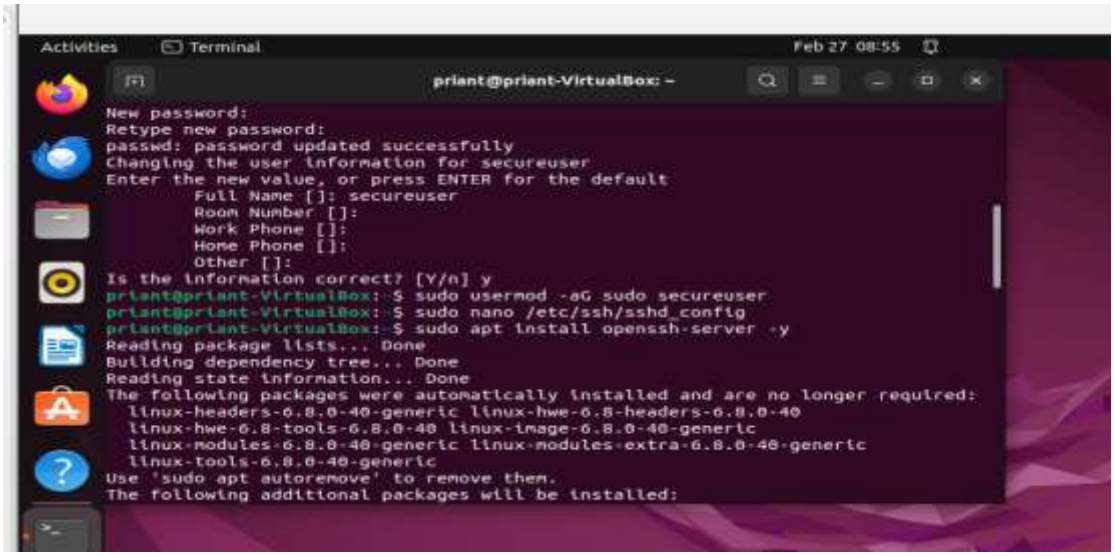
sudo apt install openssh-server -y

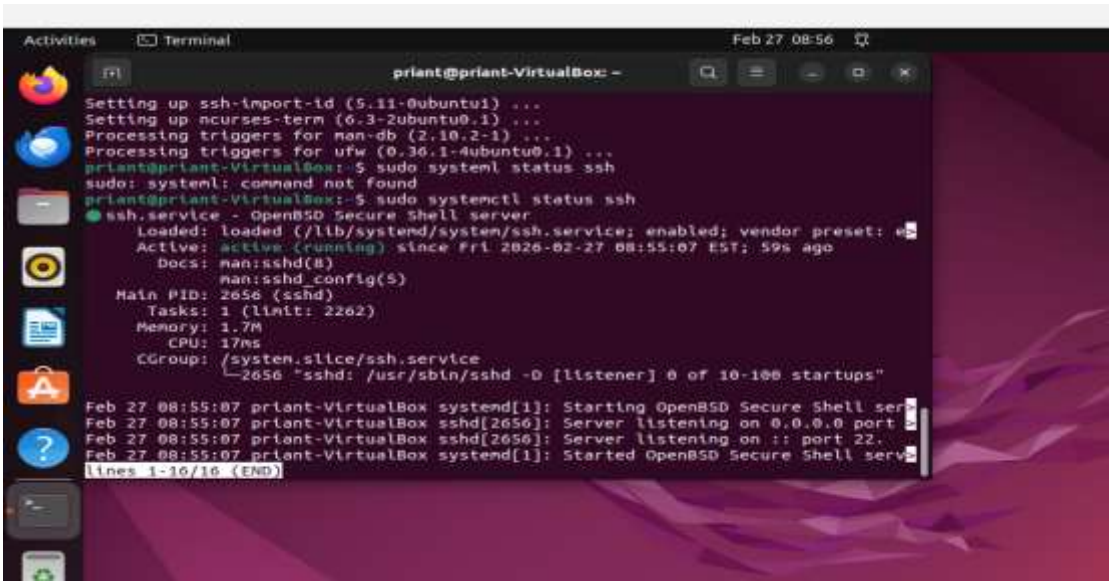
What it does:

- Installs the SSH service so I can secure it.
- Creates the /etc/ssh/sshd_config file.

Why I did this:

SSH allows remote access, so it must be installed before securing it.





4. Secured SSH by Disabling Root Login

Open SSH config:

sudo nano /etc/ssh/sshd_config

Edited this line:

Original:

#PermitRootLogin prohibit-password

Changed to:

PermitRootLogin no

What this means:

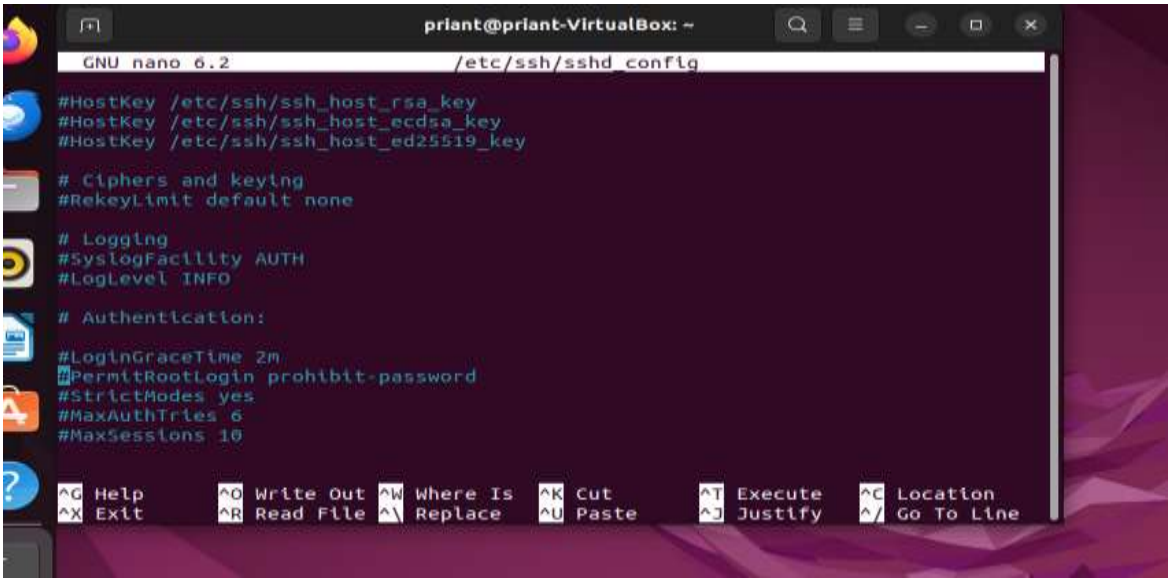
- **nano** opens the file for editing.
- Changing **PermitRootLogin no** stops anyone from logging in as root through SSH.

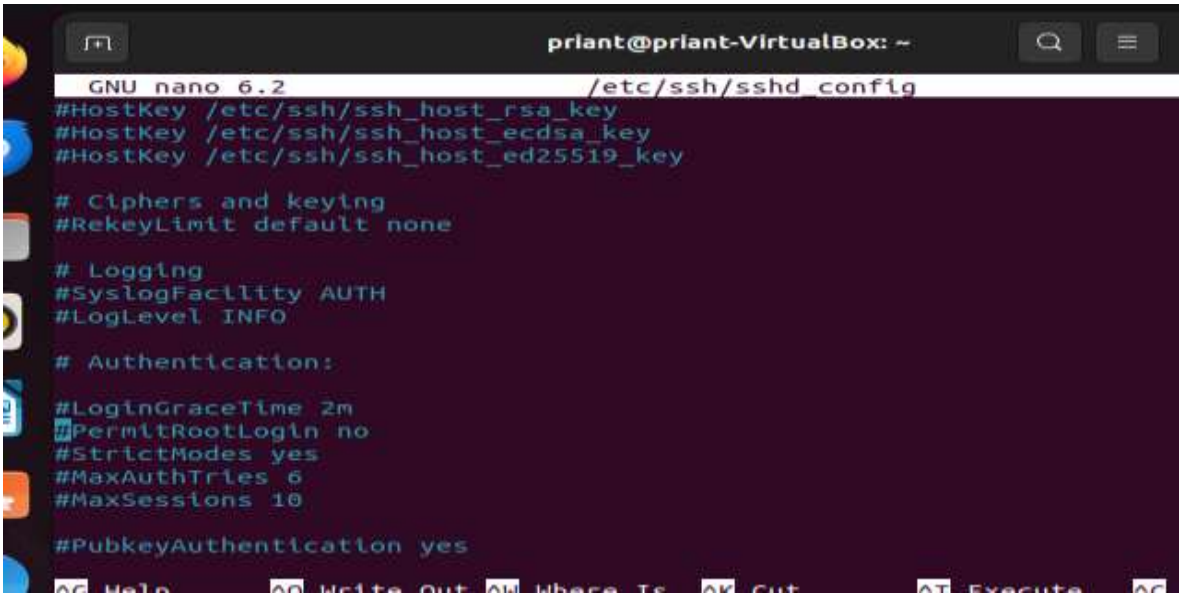
Why I did this:

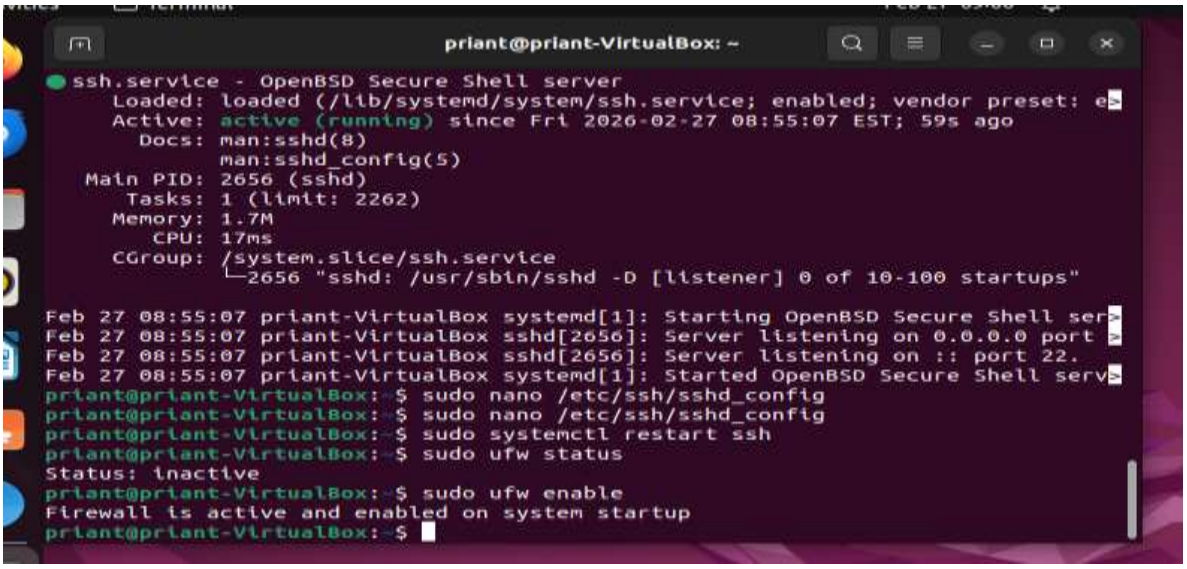
Disabling root login is a big security improvement because attackers always try the "root" user first.

Restarted SSH:

sudo systemctl restart ssh







5. Enabled and Configured the Firewall (UFW)

Commands:

sudo ufw enable
sudo ufw allow OpenSSH
sudo ufw allow 80
sudo ufw status

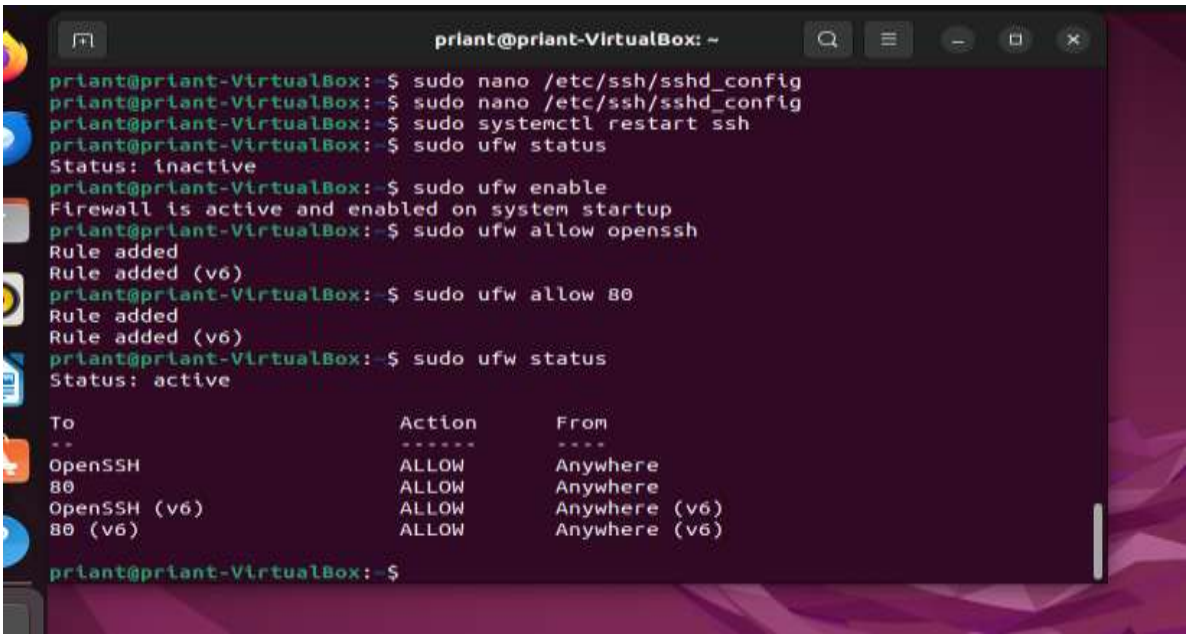
What they mean:

- **ufw enable** turns on the firewall.

- **ufw allow OpenSSH** allows SSH so I don't lock myself out (port 22).
- **ufw allow 80** allows web traffic if I ever run a website.
- **ufw status** shows which ports are allowed.

Why I did this:

A firewall blocks unwanted network traffic and only allows the ports I want open.



6. Installed and Enabled Fail2Ban (Brute-Force Protection)

Commands:

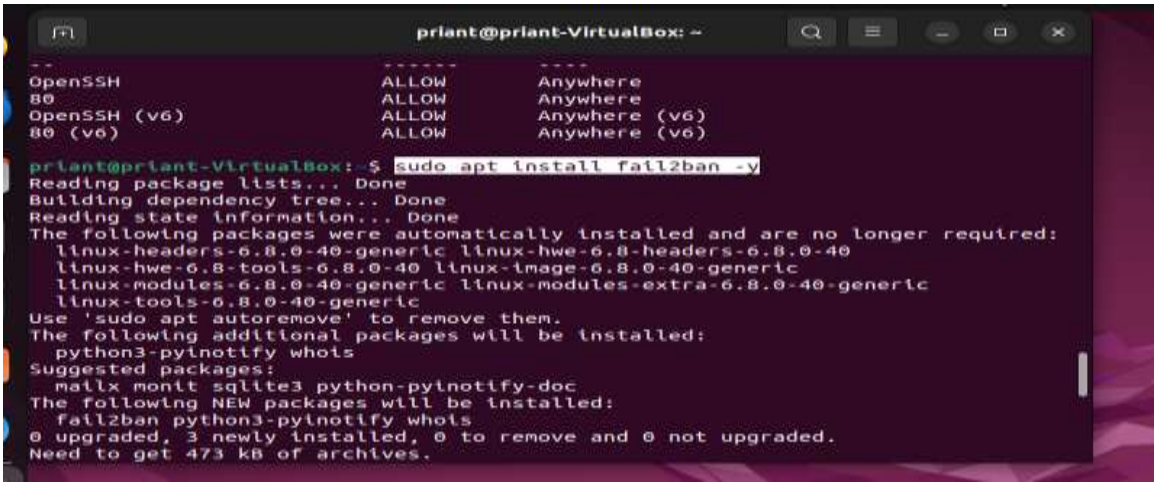
sudo apt install fail2ban -y
sudo systemctl start fail2ban
sudo systemctl enable fail2ban
sudo cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local

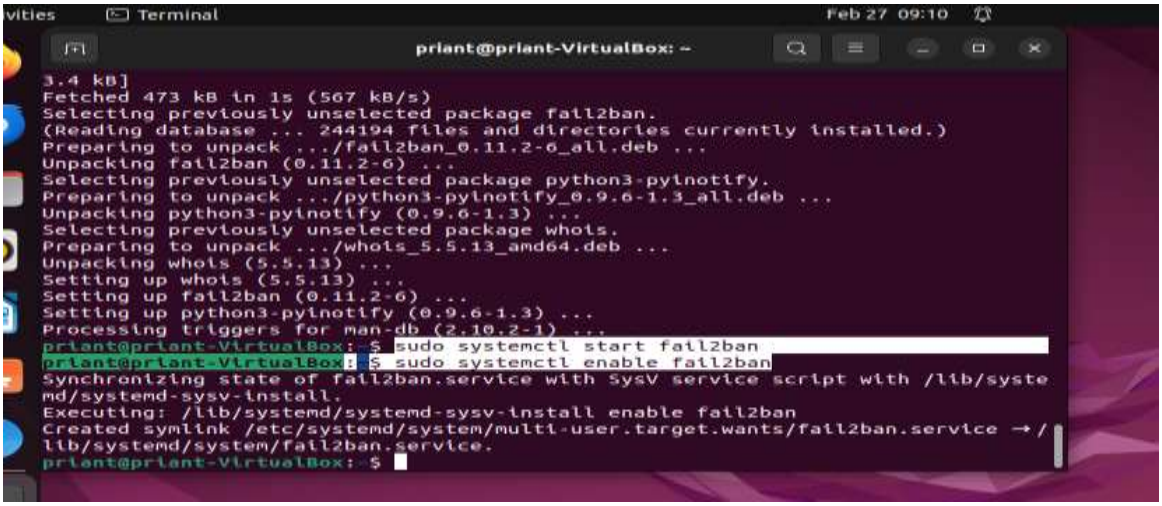
What they mean:

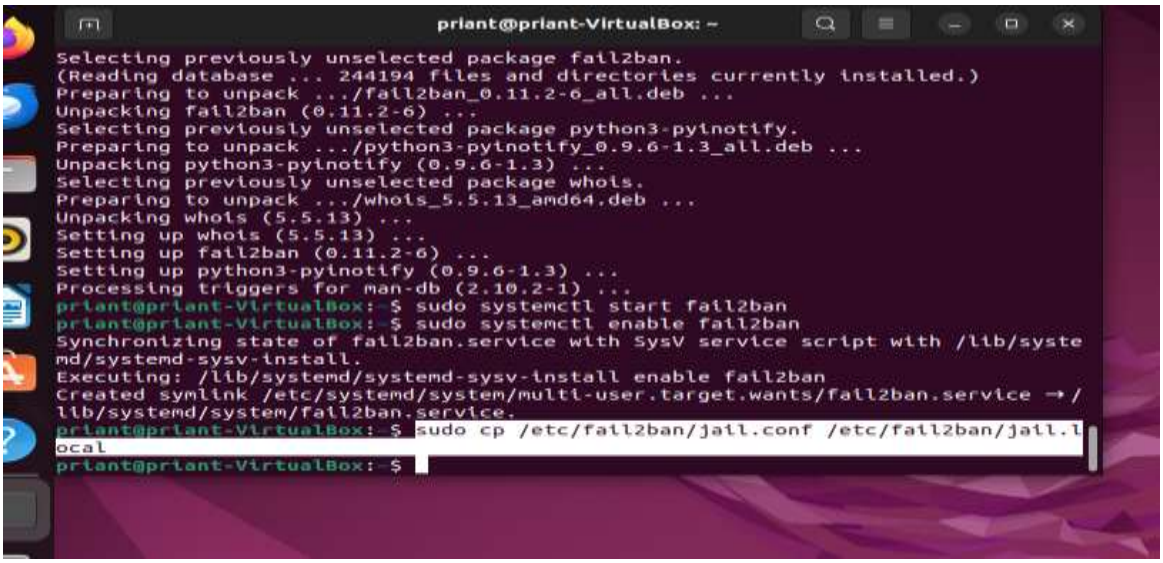
- **fail2ban** watches for repeated failed login attempts.
- **start fail2ban** runs it immediately.
- **enable fail2ban** makes it start automatically on boot.
- **copying jail.conf** creates a safe editable config file.

Why I did this:

Fail2Ban automatically blocks IP addresses that try to brute-force passwords, adding an extra security layer.







7. Checked Open Ports (Network Exposure Check)

Command:

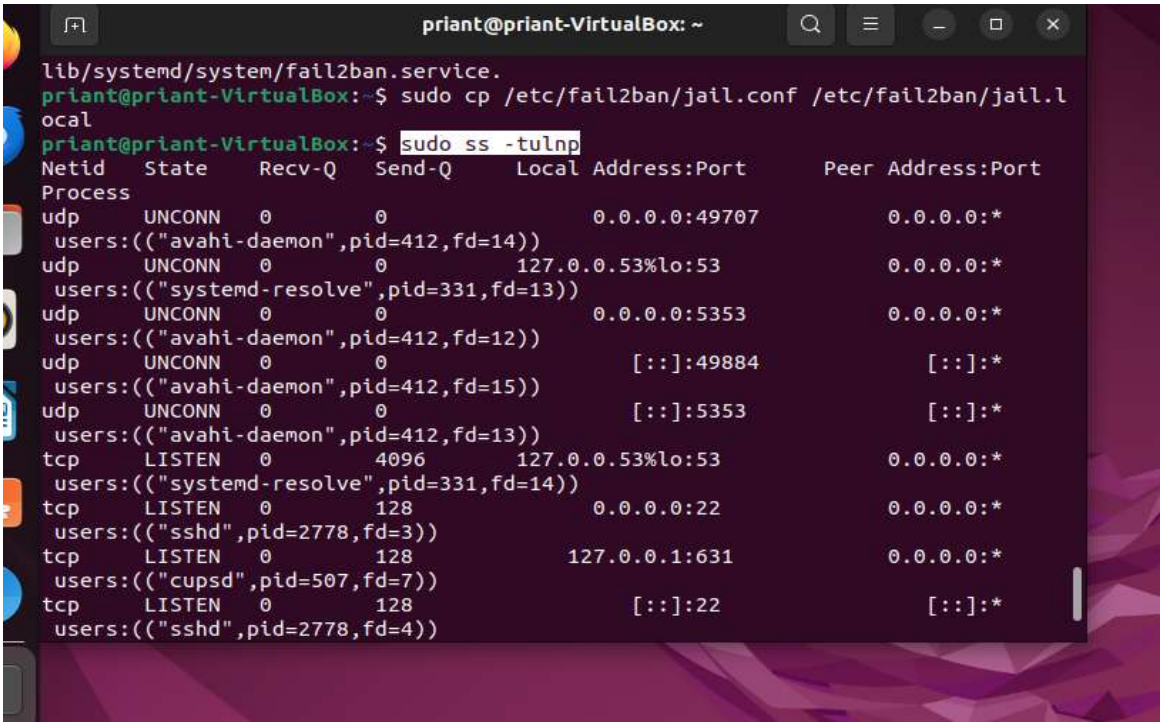
sudo ss -tulnp

What it means:

- **ss** shows open ports and network connections.
- **-t** shows TCP ports.
- **-u** shows UDP ports.
- **-l** shows listening ports.
- **-n** shows numeric output.
- **-p** shows which program is using the port.

Why I did this:

It helps me see exactly which services are visible to the network and if anything unnecessary is open.



Final Summary

I completed a Linux Security Hardening mini project where I updated the system, created a safer user, disabled root SSH login, set up the firewall, installed Fail2Ban, and checked open ports. This helped me understand how to reduce attack surface, protect SSH access, and secure a Linux machine in a simple and hands-on way.