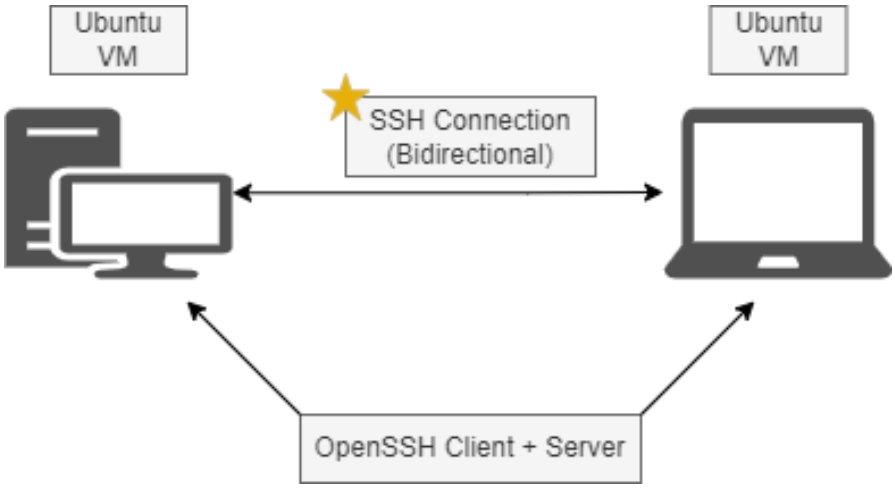
OpenSSH Lab (Setup/Hardening)
By Michael Ambeguia

Lab Purpose: The purpose of this lab is to step up OpenSSH on two of my Ubuntu Linux VMs along with hardening the SSH service for security purposes. SSH is basically another means of logging into a server or client machine so there is not much to cover on its usage.Thus, the main emphasis of the lab is to get hands-on experience hardening OpenSSH. This lab will also serve as a reference for future VM setups so that they can be accessed via SSH securely.

Lab Topology: I have my two Ubuntu VMs. One on my desktop, and the other hosted on my laptop. In this lab both VMs act as clients and servers meaning that both VMs can connect to one another and receive connection requests from one another. **Note: All hardening tasks will be mirrored on both VMs, but in the documentation the screenshots will only show the laptop configuration**



Section #1 Installing OpenSSH on the VMs

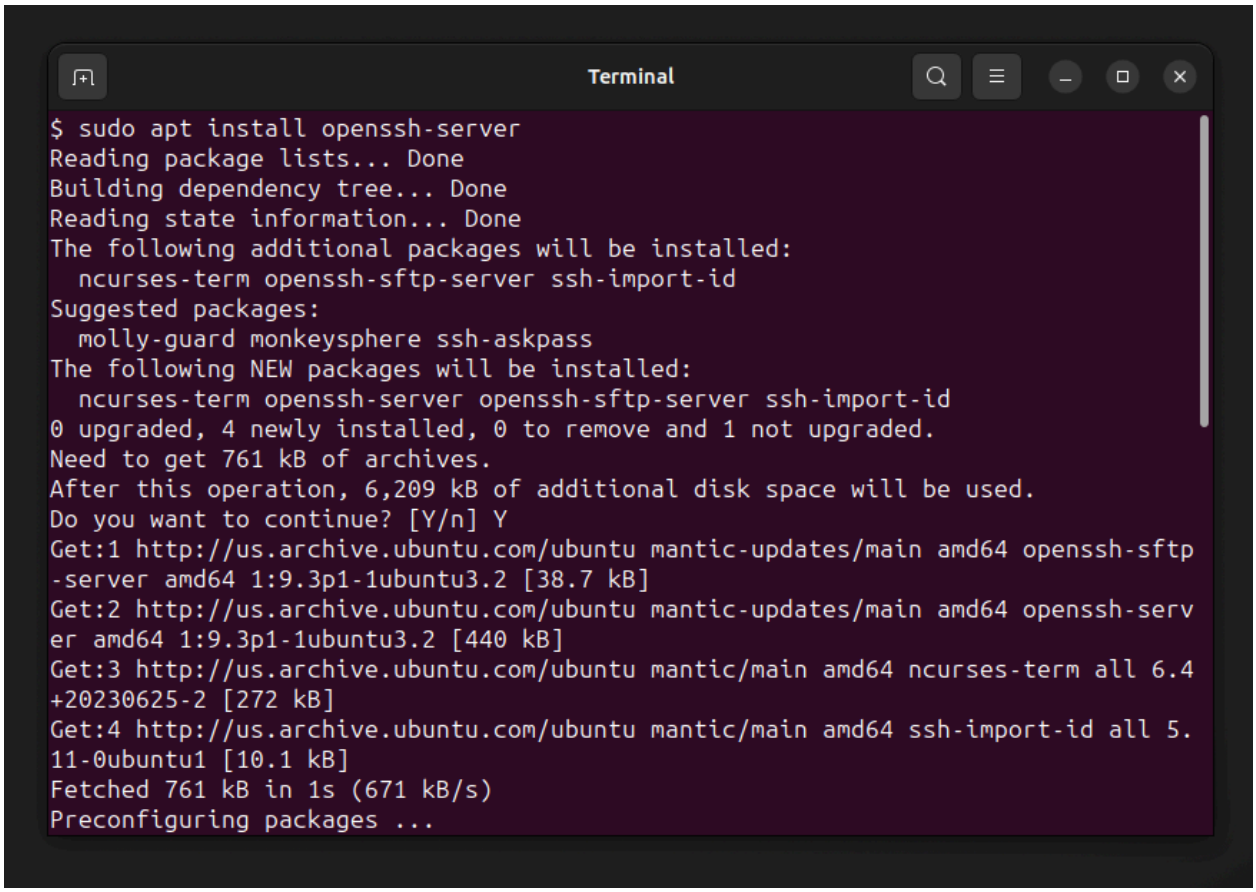
Overview:
I have two Ubuntu VMs, and since I want to have a bidirectional SSH connection established, both need to have the OpenSSH client and server programs installed.

Steps to install OpenSSH

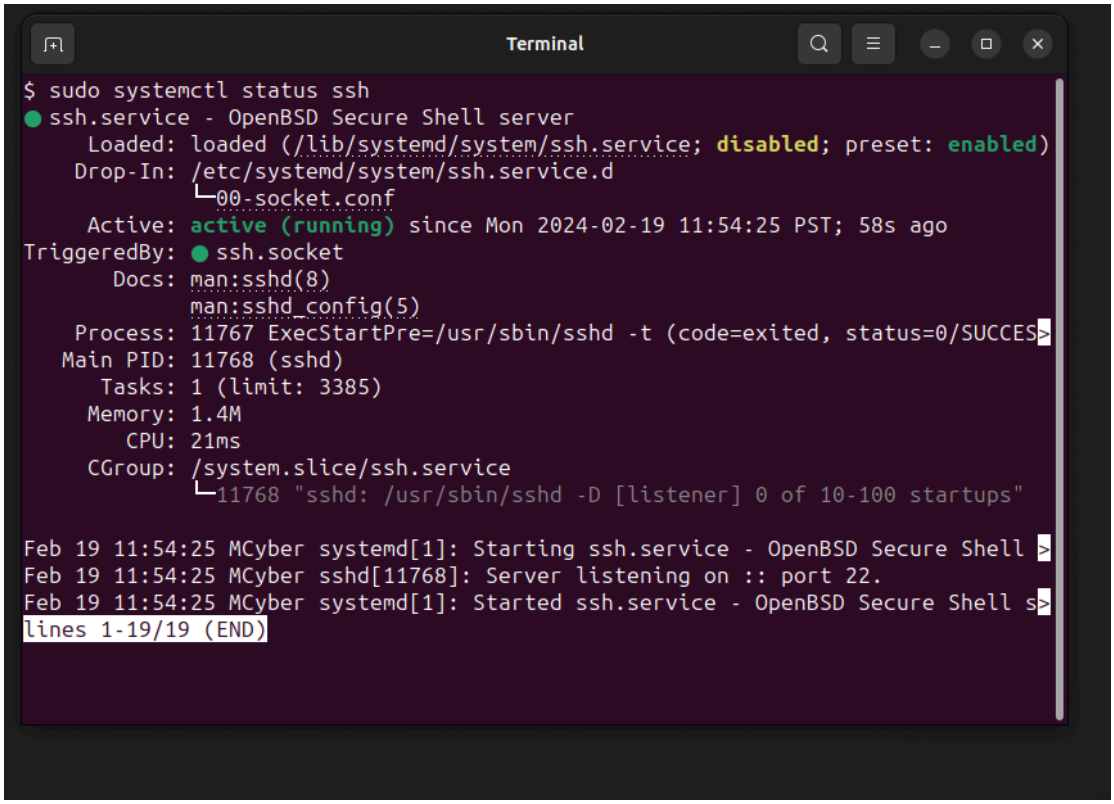
1. Install both OpenSSH client + server on both VMs.
   -1a. Using sudo or root do the following commands:
   Sudo apt update. Sudo apt upgrade.

Purpose: You need to refresh the repository so that you get the up-to-date versions of the packages you want to install.
-1b. Sudo apt install openssh–client. Go through the installation. Then do sudo apt install openssh-server.

SSH Server Installation:



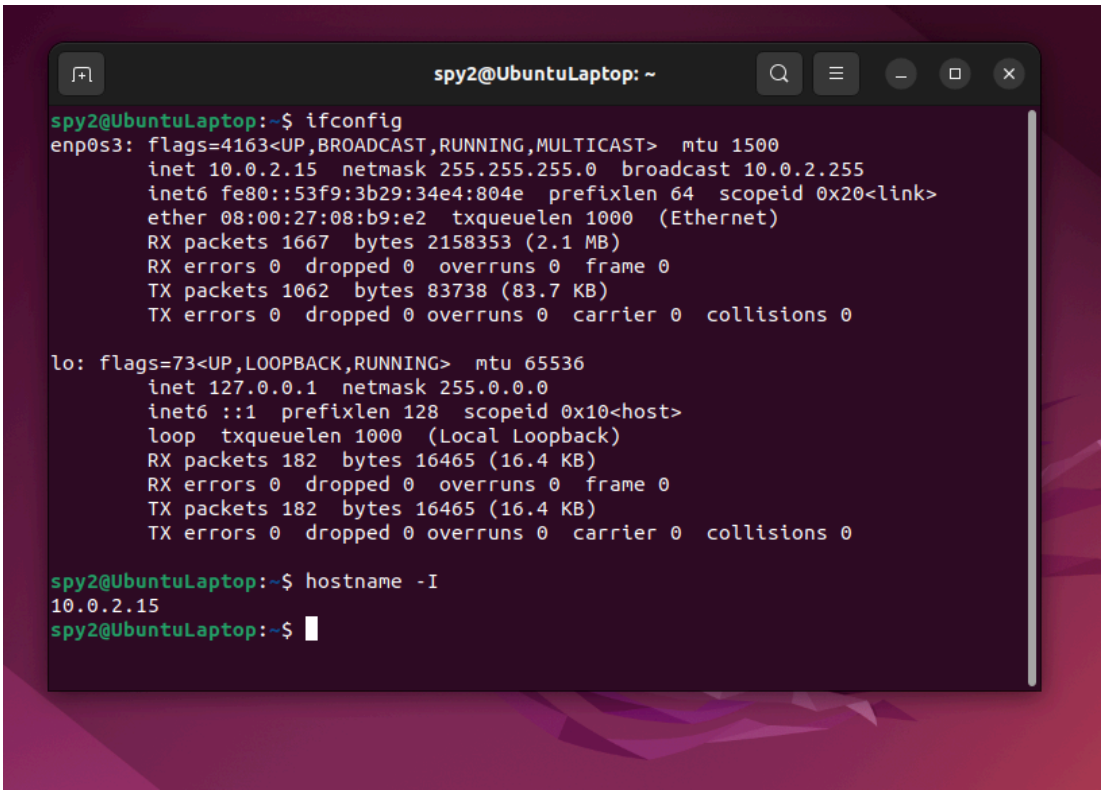
2. Verify that the server is running. Use the following command:
   Sudo systemctl status ssh.
   The server should be up and running! If not, use the sudo systemctl start ssh command!
   It should look like this now:



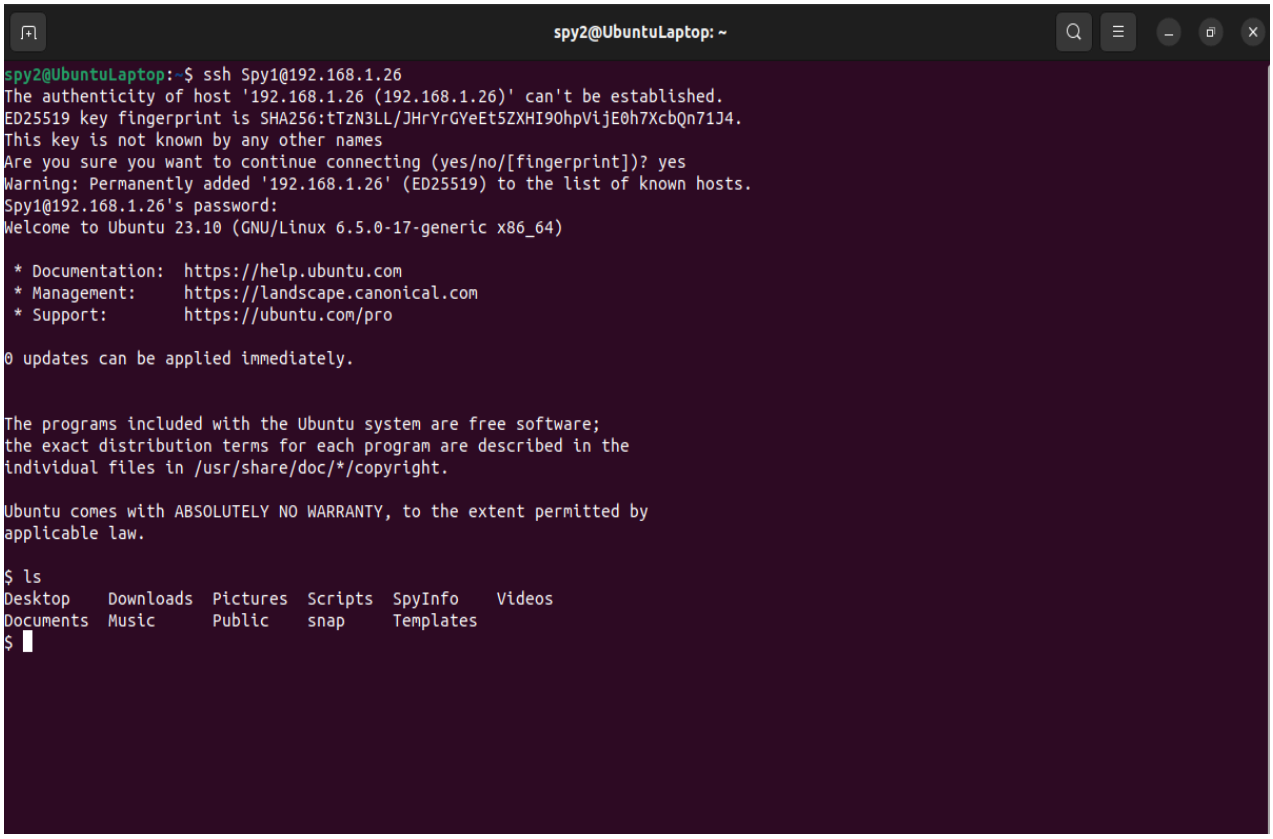
(Note that SSH not enabled so it will not be started at boot time for this lab)

Section #2 Testing and Configuring SSH:

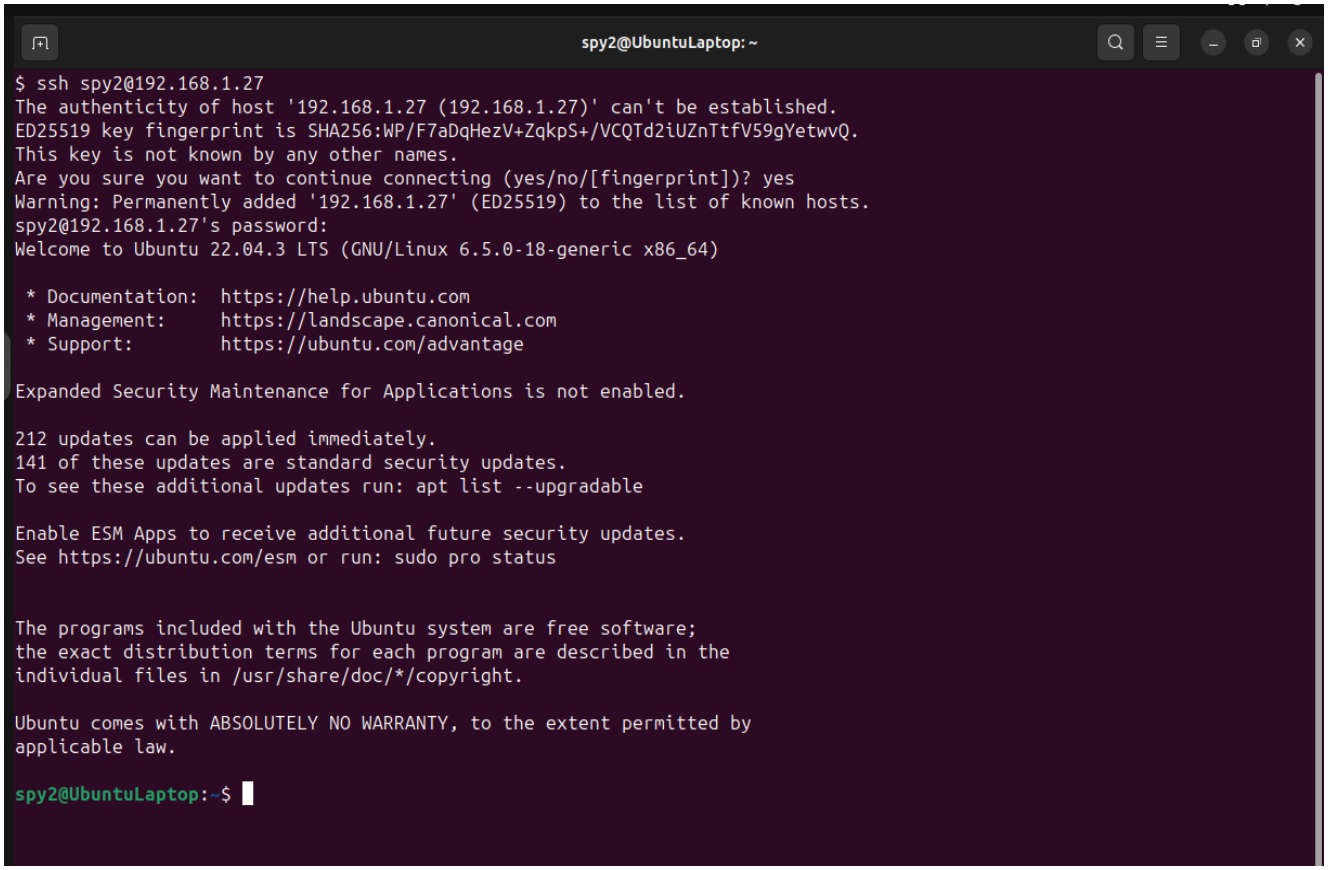
1. Find IP addresses for both devices (write them down or copy/ paste them somewhere)
   - Use the ifconfig command!
     Screenshot (laptop only):



2. Connect to the second VM using SSH:
   Screenshot (Connecting to Desktop VM from Laptop VM):



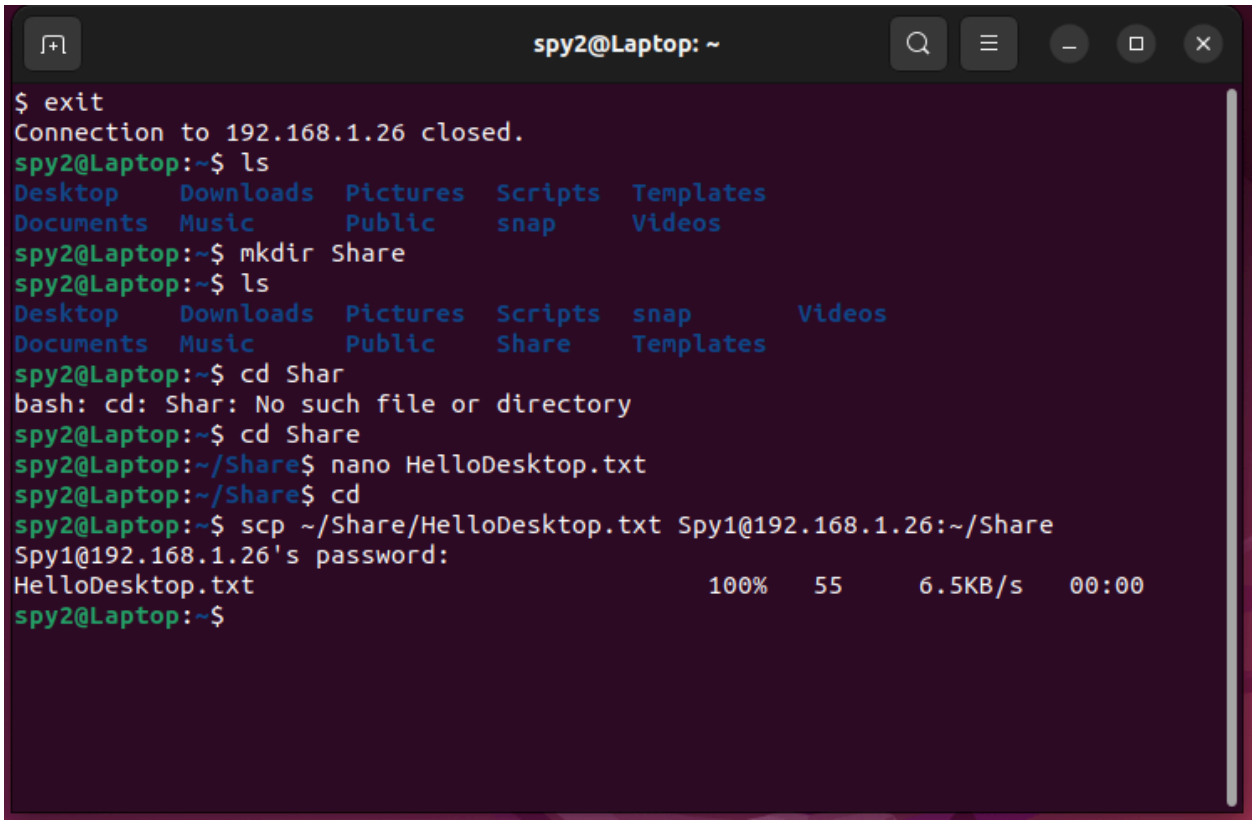
Screenshot (Connecting from desktop to laptop):



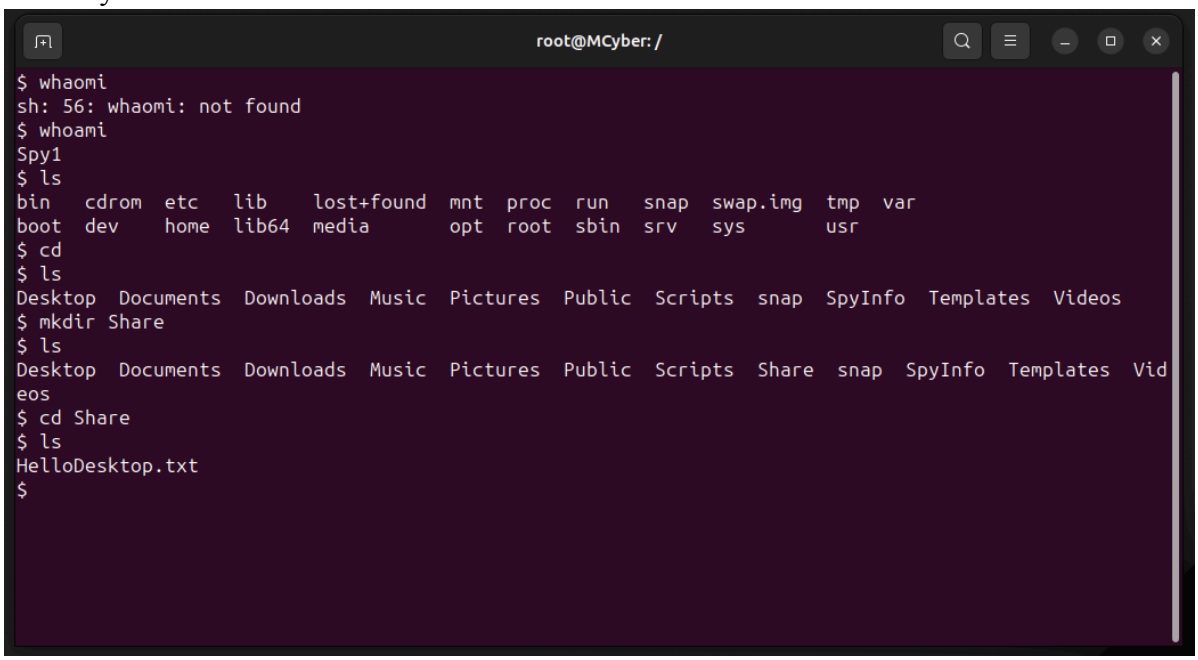
Note: Make sure that your VMs are using bridged network connections! I struggled at first to create a SSH connection because of this! Both of my VMs are now using Bridged Networks.

Part #3 SCP (Secure Copy Protocol):

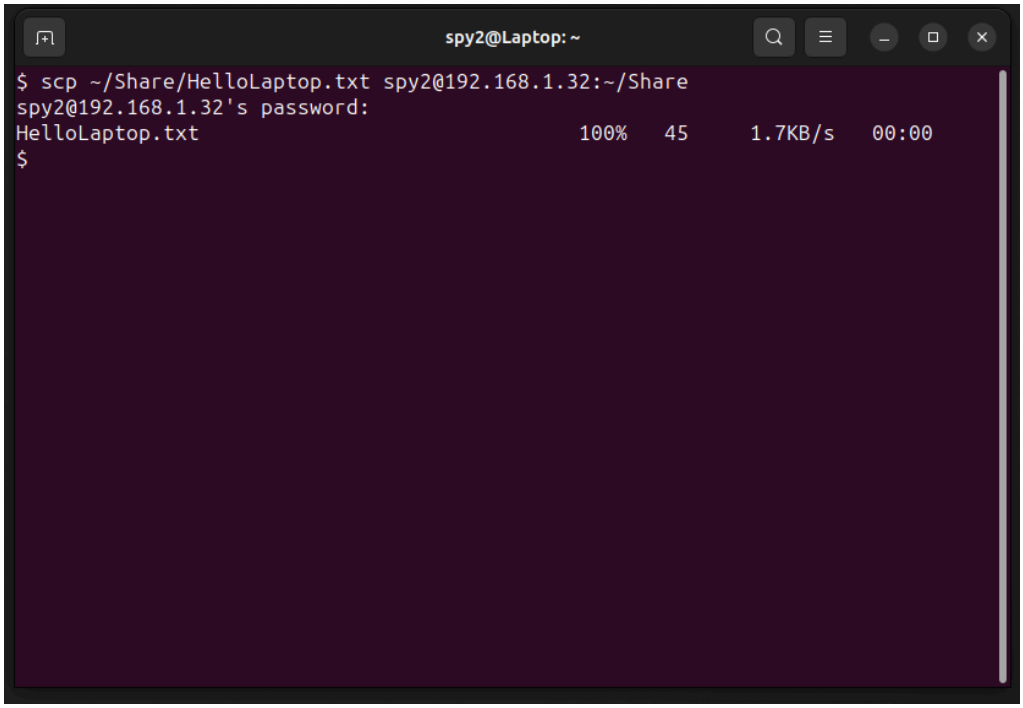
1. Task One: Transfer a file from my laptop VM to my Desktop VM using SCP.
   File transfer from laptop to desktop vm:



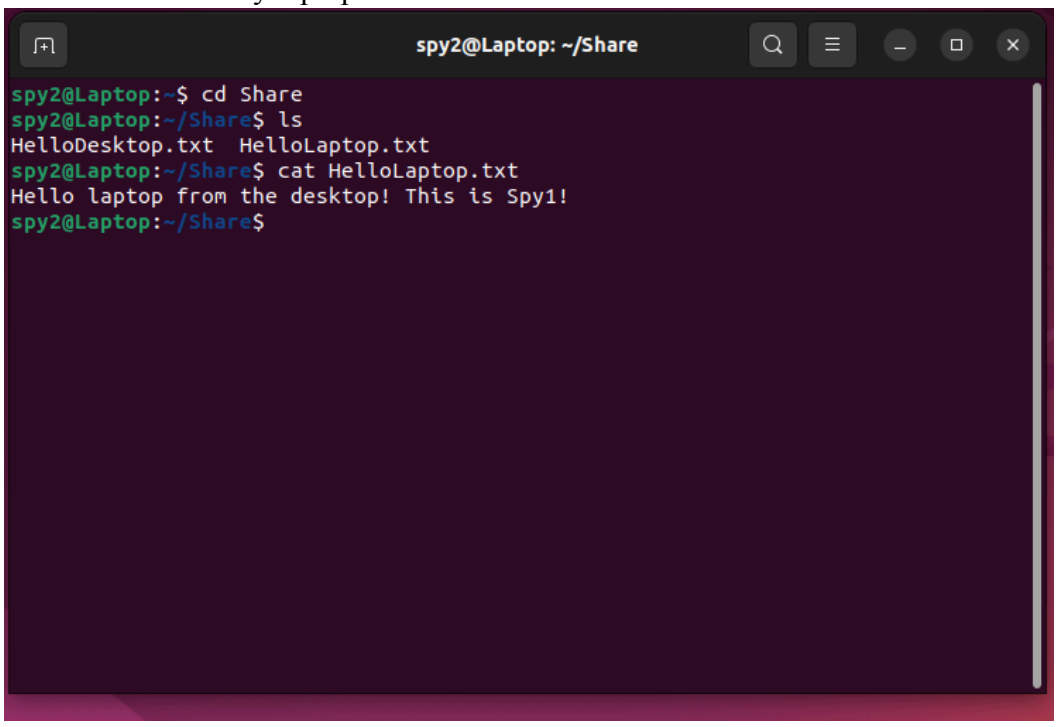
Desktop VM: Now the file from my laptop, HelloDesktop.txt, is located on Spy1's Share directory!



2. Task Two: Use SCP on the desktop to securely move a file from the desktop to the laptop vm.



Now here it is on my laptop VM:



Section #4 SSH Hardening:

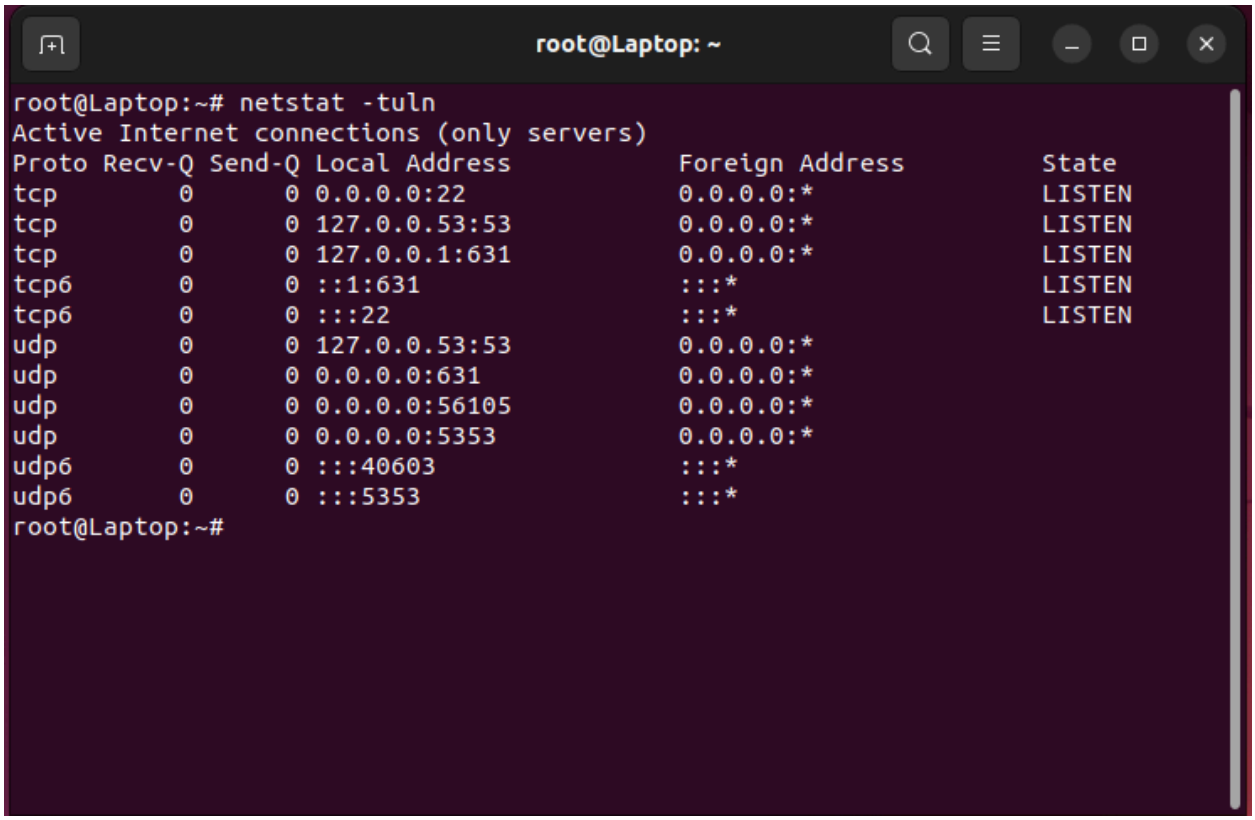
Here are some general guidelines when you harden SSH:

1. Configuration changes must match on the client and server! SSH works like an agreement between the client and the server. The client and server must agree upon common session parameters like what cryptography algorithm will be used to encrypt the session? What hashed-based authentication code will be used to ensure that the session integrity and authenticity is not compromised? Lastly, what algorithm will be used for the key exchange between the client and server? If anything on the client or server does not match up, the SSH session will not work since an agreement can't be reached. Thus for this lab I will be mirroring what I do on both my laptop Ubuntu VM and the Desktop VM. The screenshots though will only be from my laptop though unless stated otherwise.

Hardening Tasks:

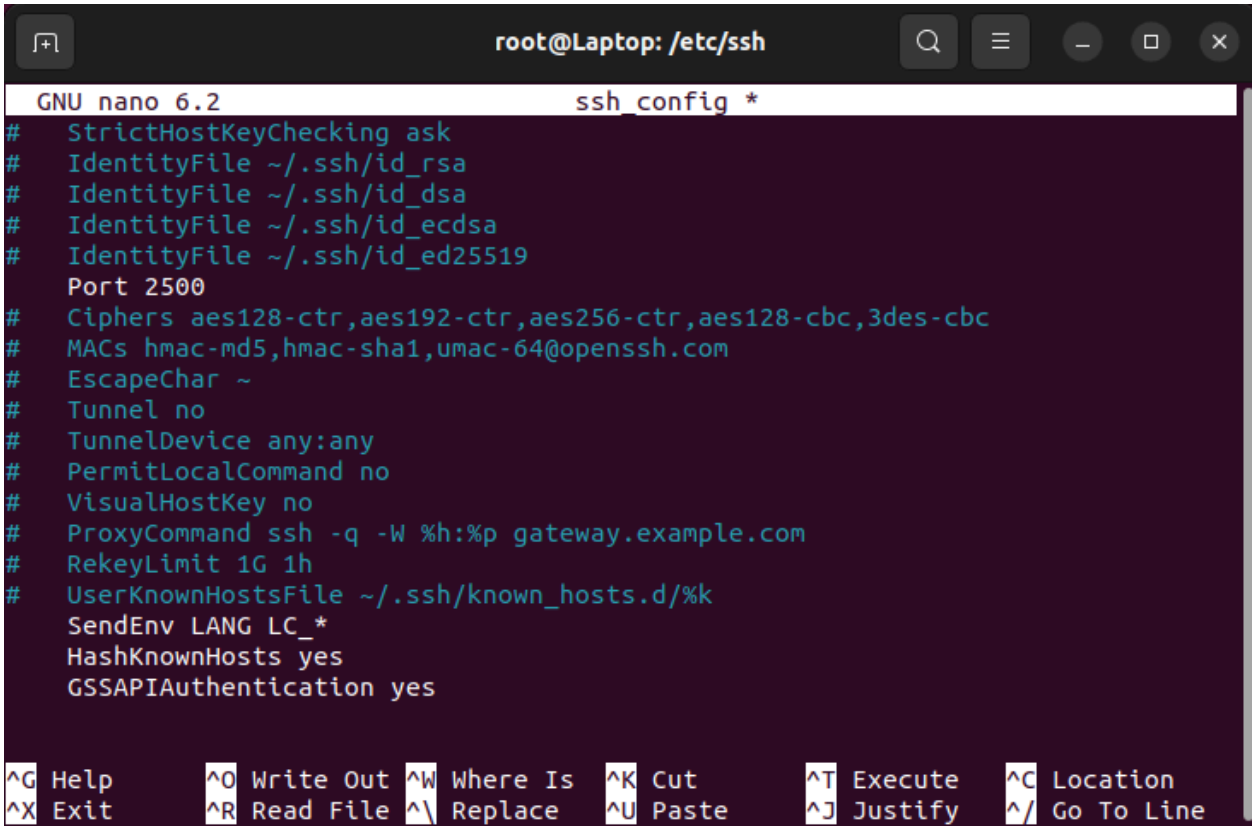
1. Change the default ports.
   Reasoning: You should change the default port because it is common knowledge that the SSH port is port 22. Hackers know this and will want to attack systems that are easy to reach. If you leave the default port as port 22 you are basically making it easier for attackers to attempt remote attacks. Changing the port number to a random value will make it hard for attackers to find. It will take more time for them to do their reconnaissance of your system, and they might simply give up and move on to the next target. Changing the port is an easy way to take a target off your device's back. The only downside is that users must remember what port to use when initiating the SSH connection.

   Steps:
   1. Check for the used ports on your system using the netstat -tuln. The tuln argument stands for tcp, udp, listening, and numerical. This argument will allow you to see what tcp and udp ports are listening for traffic and what their numerical port number is. Knowing what ports are currently up will help you determine what random port you can safely used for the ssh service while preventing any conflicts with other services.
   Screenshot:



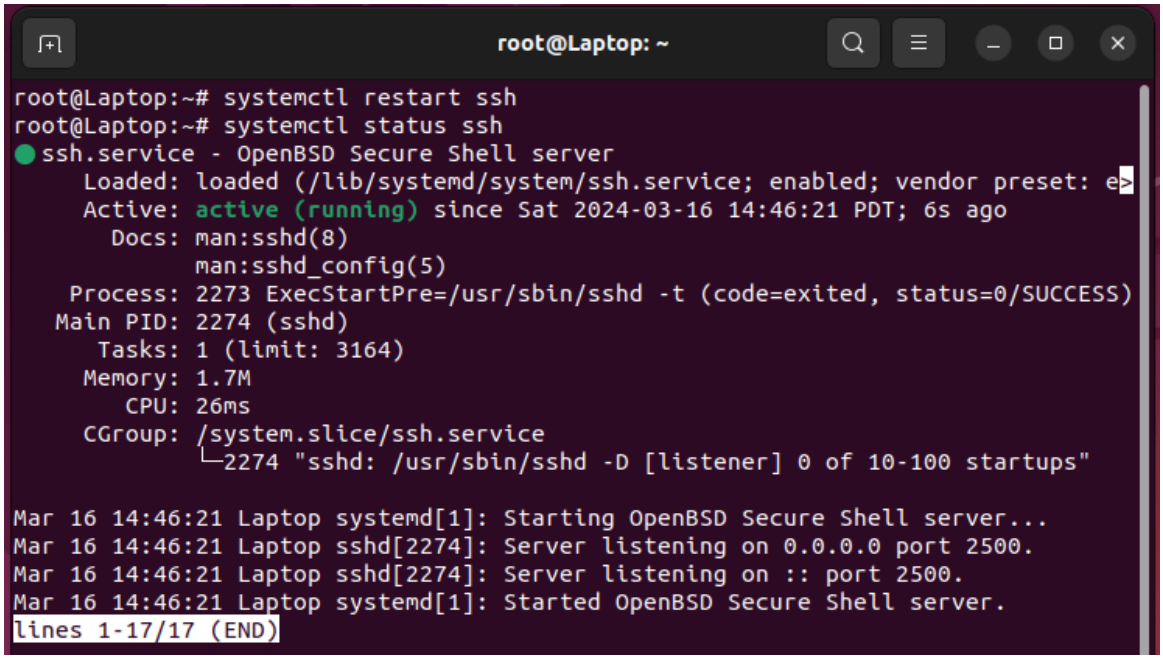
-You can see that port 22 is up and that it is listening. That means that my laptop vm is open for ssh connections.

2. Change the default port on the /etc/ssh/ssh_config file. I will choose port 2500 (a random port that is easy to remember).
Screenshot:

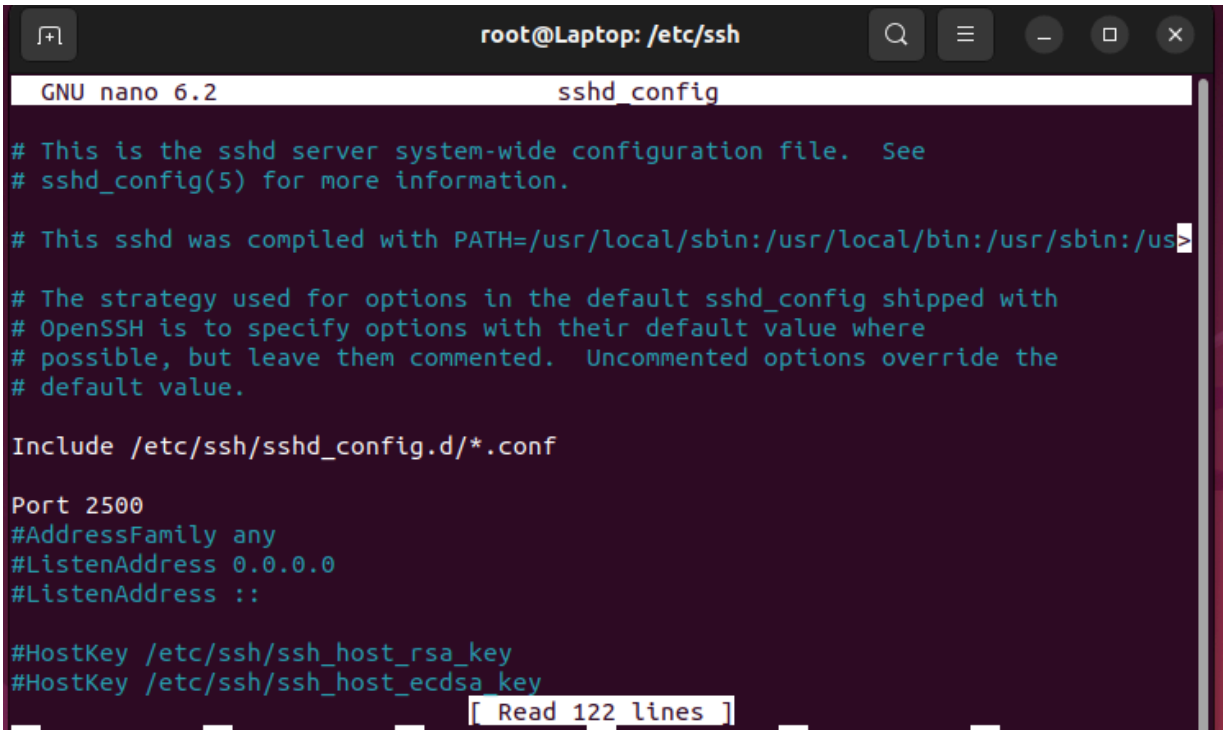


3. Restart the ssh service using systemctl restart ssh.

Screenshot:

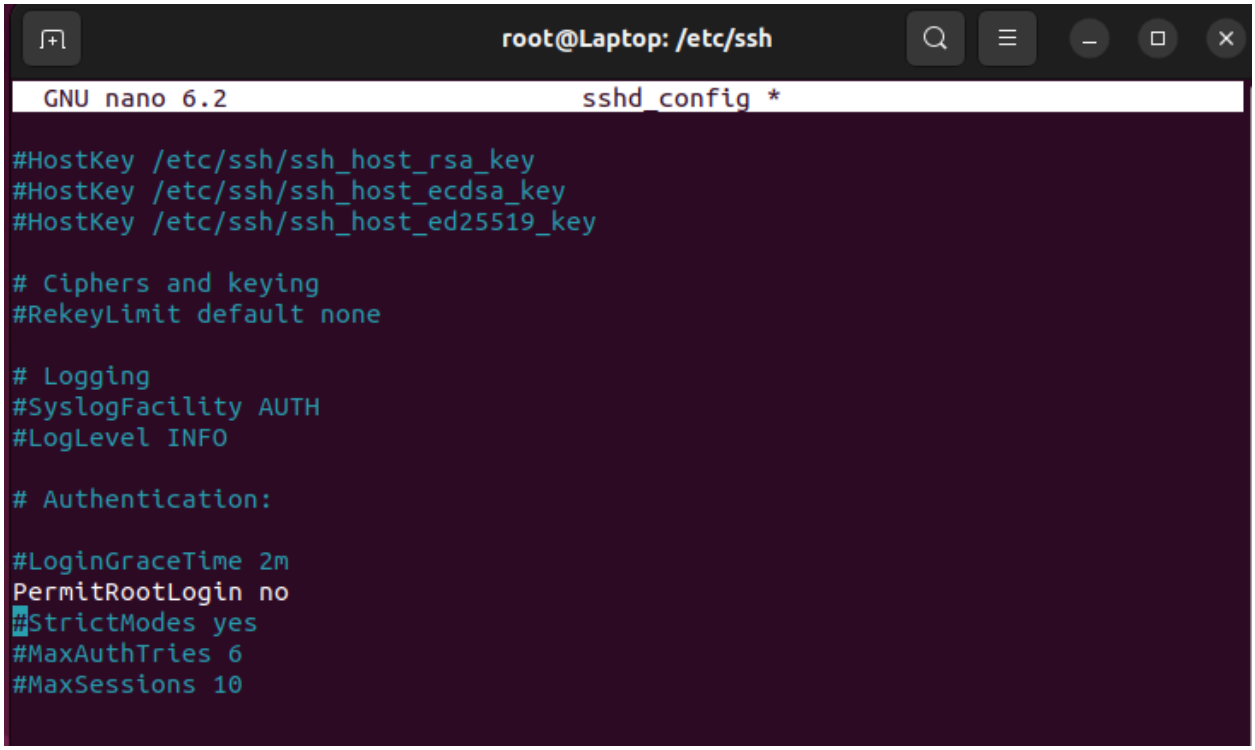


Now the ssh service is listening on port 2500. At first I changed the /etc/ssh/ssh_config file, but if you want the service to listen from port 2500 you need to change the daemon settings on /etc/ssh/sshd_config. Apparently, you do not want to mess with ssh-config since it determines what the client is able to reach. If I kept it at port 2500 that would mean that my client could only connect to port 2500 and not port 22. For my purposes, I do not want that so I reinstated the default port 22.

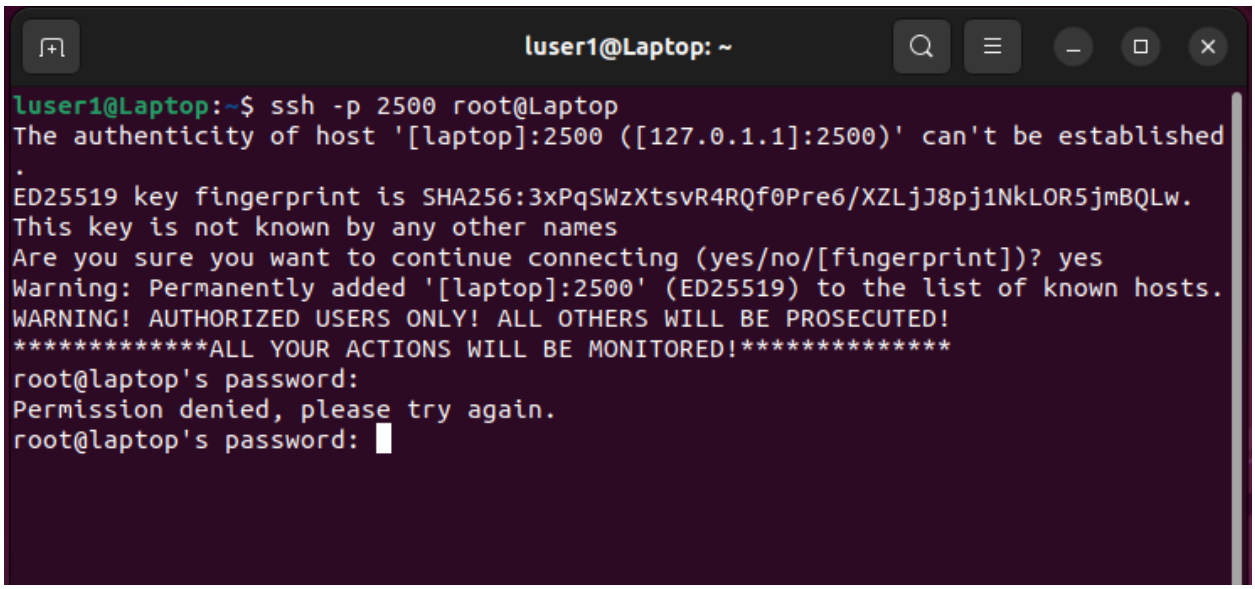


2. Disable Root SSH login
   Reasoning: Disabling root access is vital for many reasons. Allowing root to connect via SSH is deadly since root has unlimited authorization and can do anything on your Linux machine. Under no circumstances should you risk allowing an attacker to use your root account remotely. Root access should be done on premises since you can at least control who is attempting to log in with root privileges physically. Via SSH you do not know if the person using ssh is the actual administrator. You only know that the admin account is trying to connect, but you can't be 100% certain that the actual admin is the one initiating the ssh session with the account. In the office you can at least have confirmation that it is the real admin. Despite the fact that SSH is convenient for admins, privileged actions should be done in person.

   Steps. There is only two things to do:
   1.Change the /etc/ssh/sshd_config file and ban root login via ssh onto the device. After restart the ssh service like before. In fact, always restart services after you mess with their daemon config files to enforce the new parameters!
   Screenshot:



Here is a test:

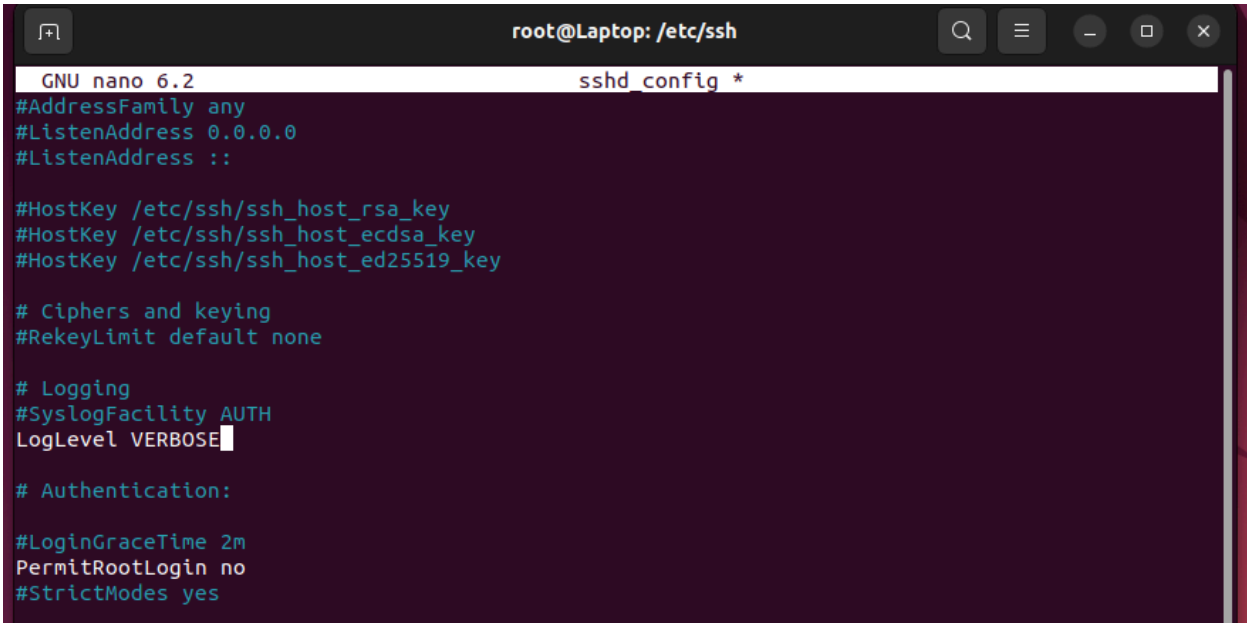


Root can’t log in. I used the correct password, but permission was denied!

3. Setup connection logging
   Reasoning: Logging ssh connections will allow you to know who is trying to connect to the device and it provides key information about the SSH connections parameters. Key

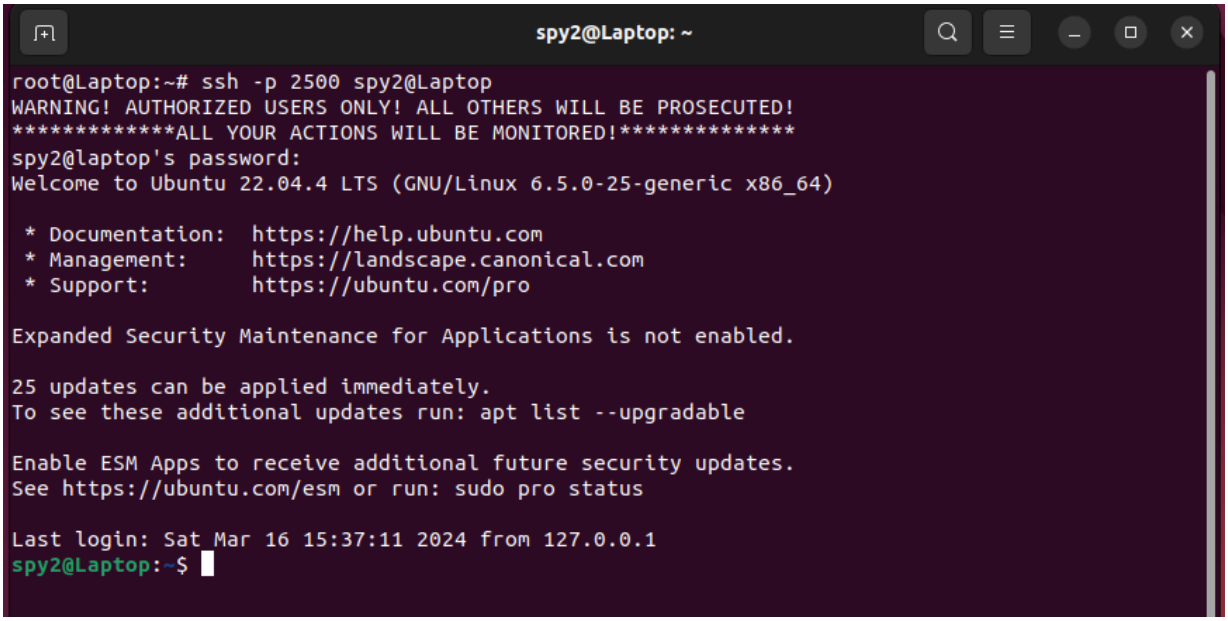
   Steps:

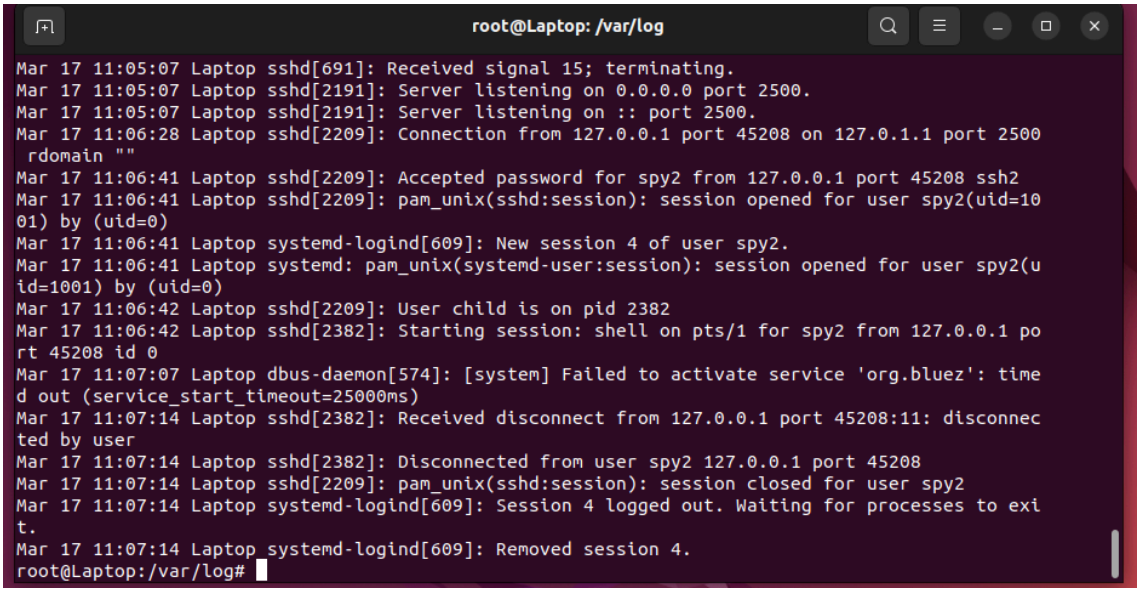
1. Change the log settings in the /etc/ssh/sshd_config file from info to verbose.
   Screenshot:



2. Restart the SSH service.

3. Test the logging. Connect to the device using SSH then check /var/log/auth.log for verbose logging info.
   Screenshot:

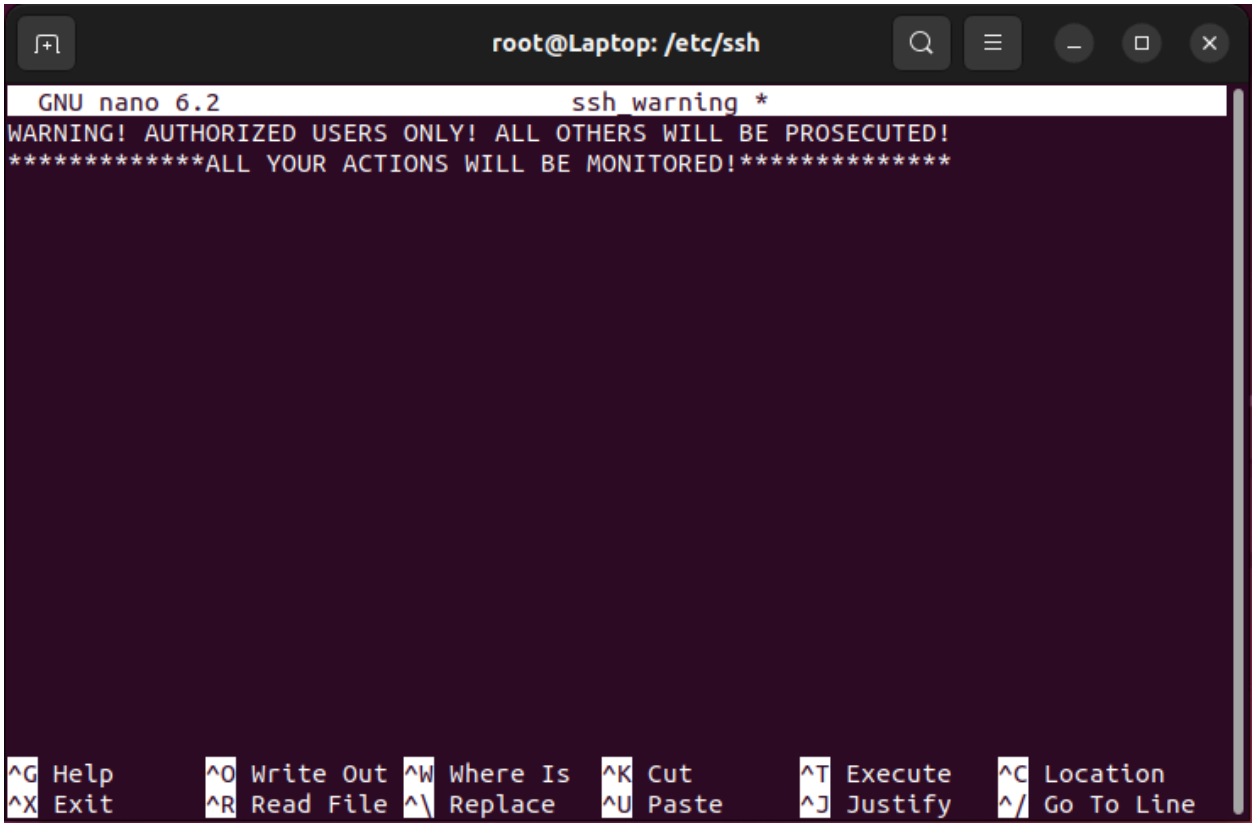




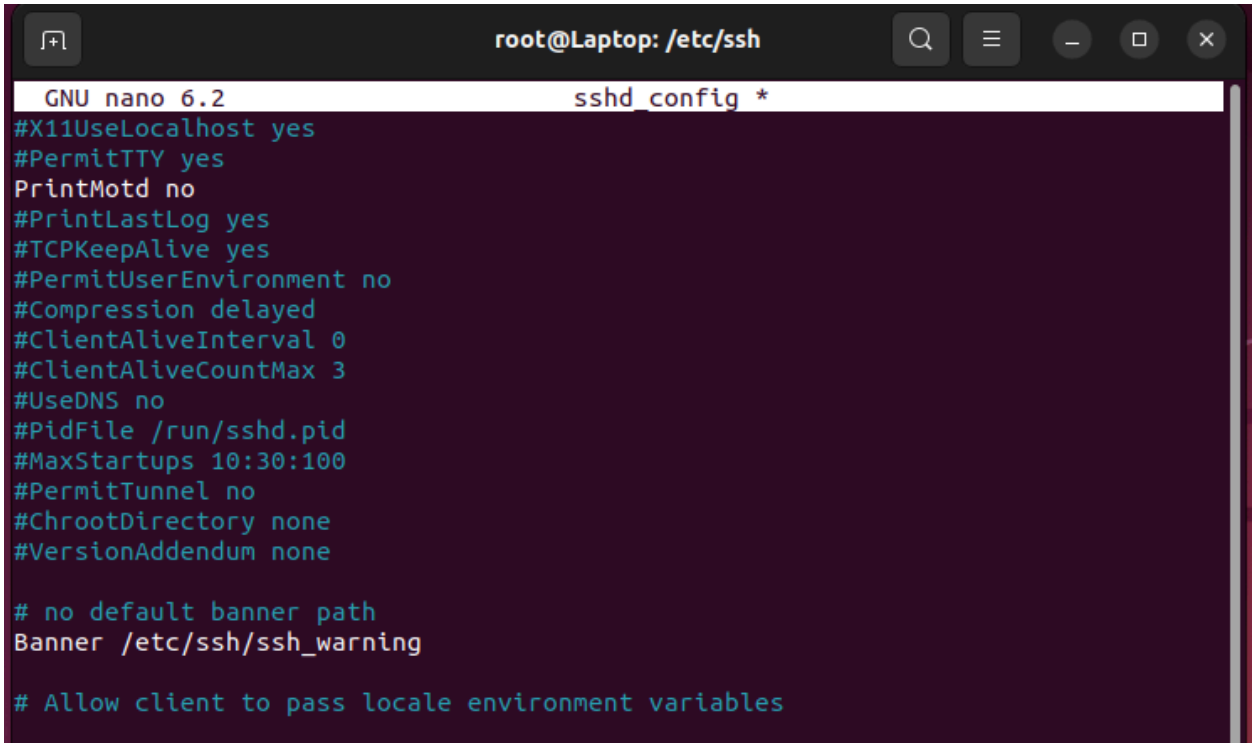
The log file auth.log now tells you more info like when a user connected, disconnected and when they logged out of the SSH session.

4. Setup a custom login banner
   Reasoning: Creating a login banner will help you legally. If you can present a warning to attackers you can be assured that you did your job by informing “trespassers” that they are “trespassing” on “private property”. In the legal world you can’t prosecute someone if there was no “no trespassing” sign in place since the defendant did not know that they were walking onto private property. Same thing in the cyber world. There has to be some warning for unauthorized use. That way in the event that you do get hacked and can identify the culprit, you can prove in court that you gave reasonable warnings and not get your case thrown out. Login banners can also be used to inform unauthorized users that their actions will be monitored as well.

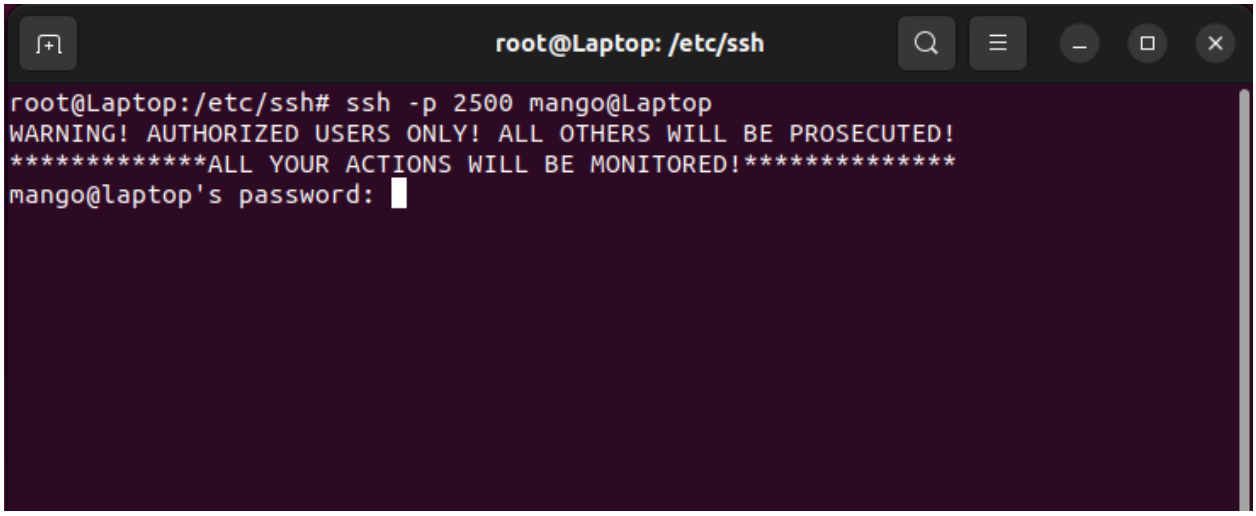
   Steps:
   1.Create a banner in the /etc/ssh directory.
   Screenshot:



2. Add the path of the banner file onto the /etc/ssh/sshd_config file.
Screenshot:



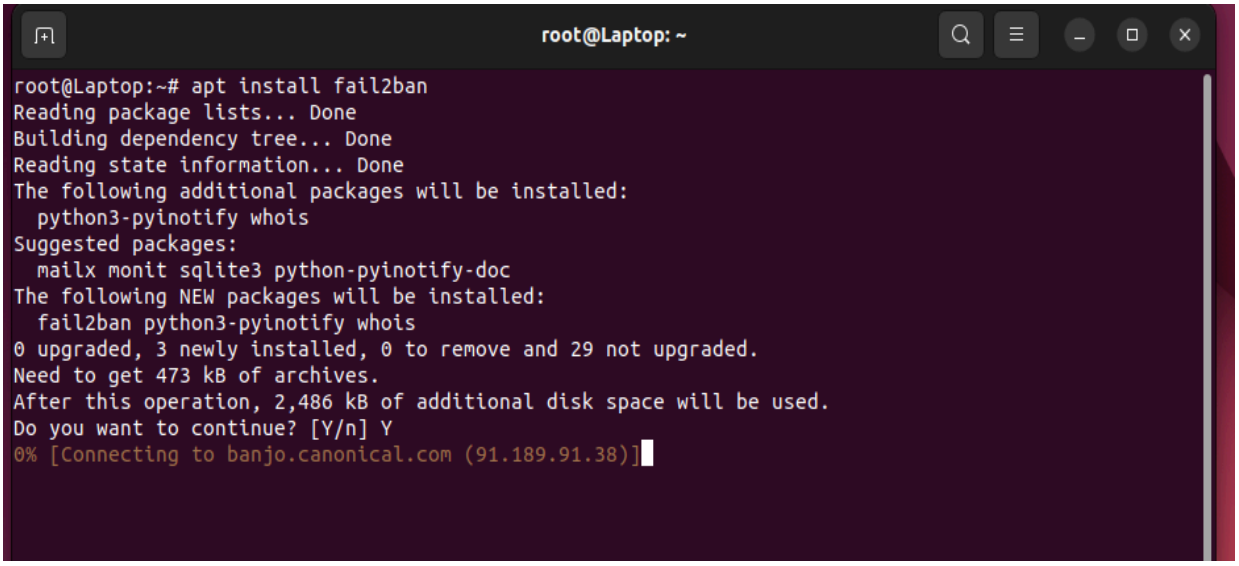
Here you can see that the banner does appear when I attempt to SSH into the laptop vm as Mango:



5. Setup brute force protection for OpenSSH
   Reasoning: Password based SSH authentication can be attacked via brute force methods. Attackers can attempt to try random combinations of usernames and passwords until one works and they compromise your system. If you are using password based SSH authentication there are tools

   Steps:
   1. Make sure fail2ban is installed on the system.

Screenshot:

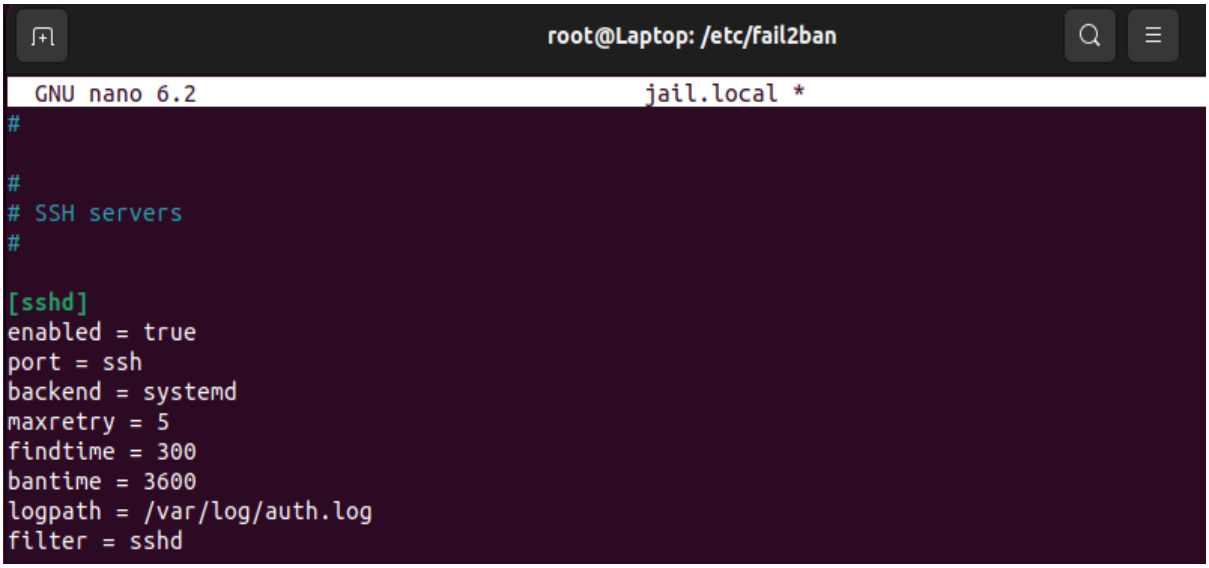


I had to install fail2ban since this Ubuntu VM is new.

2. Edit the/etc/fail2ban/jail.local config file and set the SSH “jail” parameters that will ban a certain ip address after a certain number of failed attempts. This is a multiple step process since jail.config is not created at first and must be created as a copy as jail.conf.

Screenshot:

Now I have a jail.local config file to configure. The reason why you need a jail.local file is because fail2ban updates might erase any custom configurations in the jail.conf file. Time to customize the jail settings on jail.local.



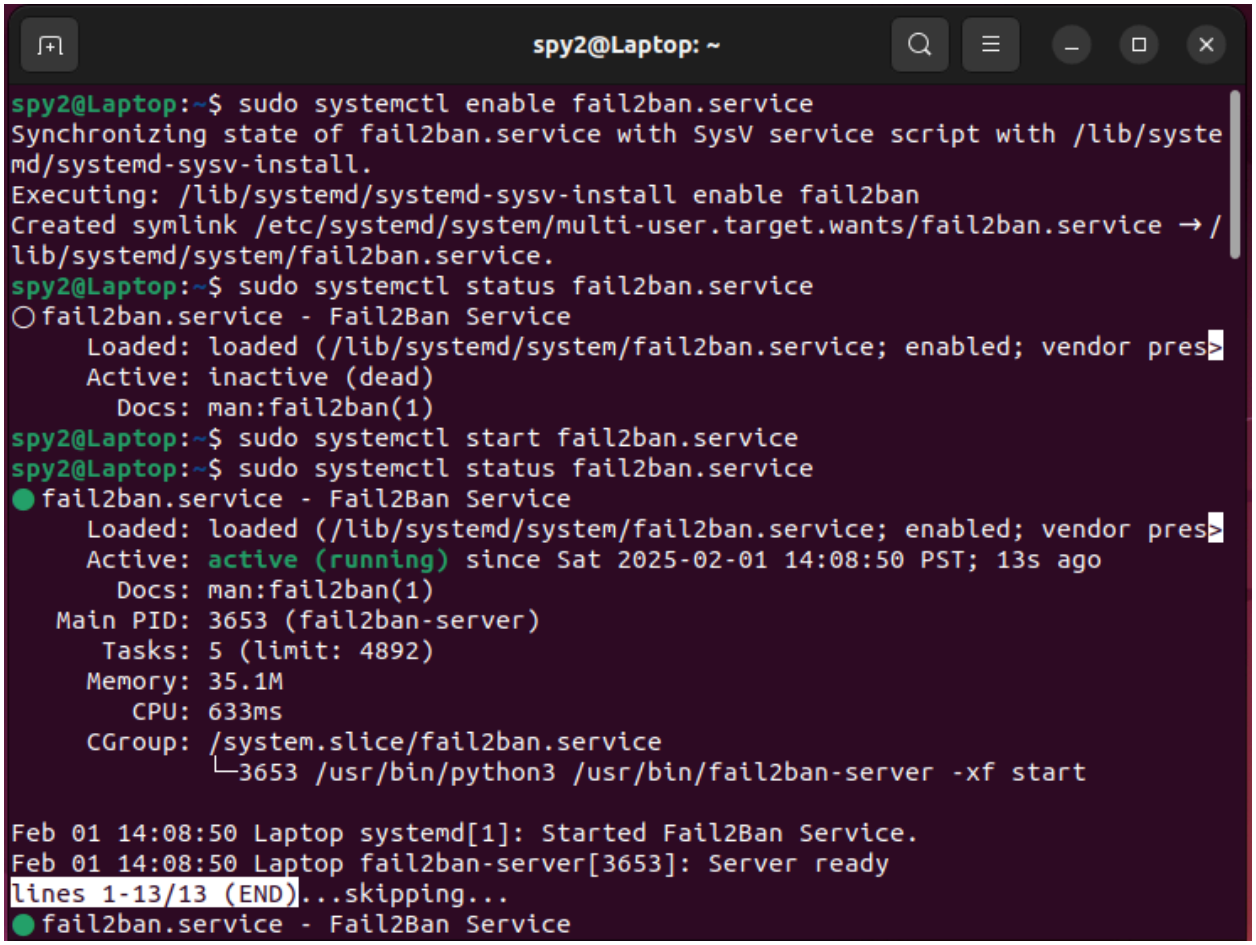
Setting	Value	Reasoning
enabled	true	I want fail2ban to monitor SSH or the sshd service.
port	ssh	I want fail2ban to 0.monitor the ssh port for failed login attempts.
backend	systemd	Systemd is responsible for the sshd service.
maxretry	5	I want to only allow 5 login attempts before an IP address is banned.
findtime	300	In 5 minutes, if 5 attempts are unsuccessful, the IP address will be banned.
bantime	3600	The banned IP address should be banned for

		1 hour.
logpath	/var/log/auth.log	The logs that fail2ban should monitor are in this path.
filter	sshd	Fail2ban will parse sshd log files.

Then as always, restart the fail2ban service.

3. Test Fail2ban using Kali Linux:

I am going to test fail2ban by attempting to brute force my way into my Ubuntu VM using Hydra from my Kali Linux VM. First I will enable the fail2ban service. Then I will perform my attack to test that it is working.



I first enabled the service. Then I had to start the service since enabling it means that it will start after rebooting, and since I don't want to reboot I have to start the service manually. Now my test can commence.

```
spy2@Laptop:~$ sudo fail2ban-client status
Status
|- Number of jail:      1
`- Jail list:   sshd
spy2@Laptop:~$
```

Currently the fail2ban jail for sshd is empty. Not for long though!

```
┌──(kali㉿kali)-[~]
└─$ hydra -t 16 -l spy2 -P /usr/share/wordlists/rockyou.txt -s 2500 192.168.1.30 ssh
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in militar

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-02-01 14:12:39
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommend
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344
[DATA] attacking ssh://192.168.1.30:2500/
[STATUS] 88.00 tries/min, 88 tries in 00:01h, 14344315 to do in 2716:44h, 12 active
   [STATUS] 62.33 tries/min, 187 tries in 00:03h, 14344216 to do in 3835:22h, 12 acti
^CThe session file ./hydra.restore was written. Type "hydra -R" to resume session.

┌──(kali㉿kali)-[~]
└─$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qle
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group def
    link/ether 08:00:27:d2:26:79 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.39/24 brd 192.168.1.255 scope global dynamic noprefixroute eth0
       valid_lft 83695sec preferred_lft 83695sec
    inet6 fe80::aa80:9db2:e3f6:7fb9/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

Now my Kali VM is in the jail!

```
spy2@Laptop:~$ sudo fail2ban-client status sshd
Status for the jail: sshd
|- Filter
|  |- Currently failed: 1
|  |- Total failed:     286
|  `- Journal matches:  _SYSTEMD_UNIT=sshd.service + _COMM=sshd
`- Actions
   |- Currently banned: 1
   |- Total banned:     1
   `- Banned IP list:   192.168.1.39
spy2@Laptop:~$
```

4. Free the Kali VM from the jail.

```
spy2@Laptop:~$ sudo fail2ban-client set sshd unbanip 192.168.1.39
0
spy2@Laptop:~$ sudo fail2ban-client status sshd
Status for the jail: sshd
|- Filter
|  |- Currently failed: 0
|  |- Total failed:     300
|  `- Journal matches:  _SYSTEMD_UNIT=sshd.service + _COMM=sshd
`- Actions
   |- Currently banned: 0
   |- Total banned:     1
   `- Banned IP list:
spy2@Laptop:~$
```