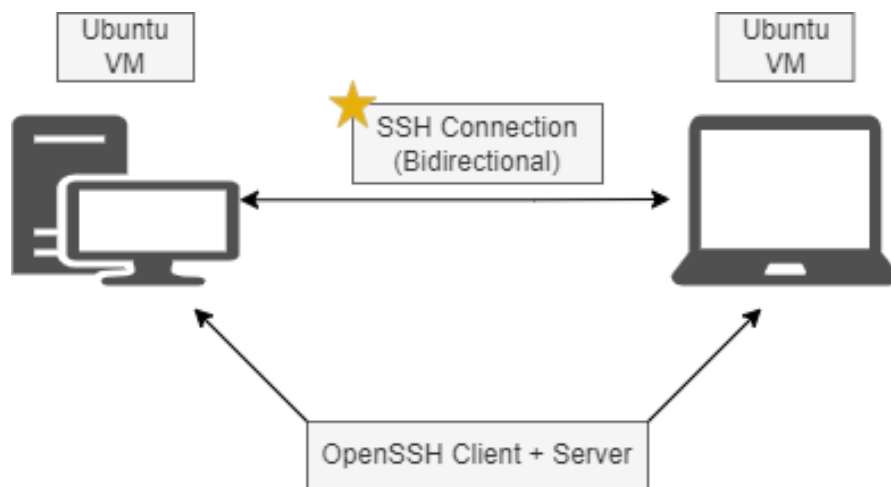


# OpenSSH Lab (Setup/Hardening)

## 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.

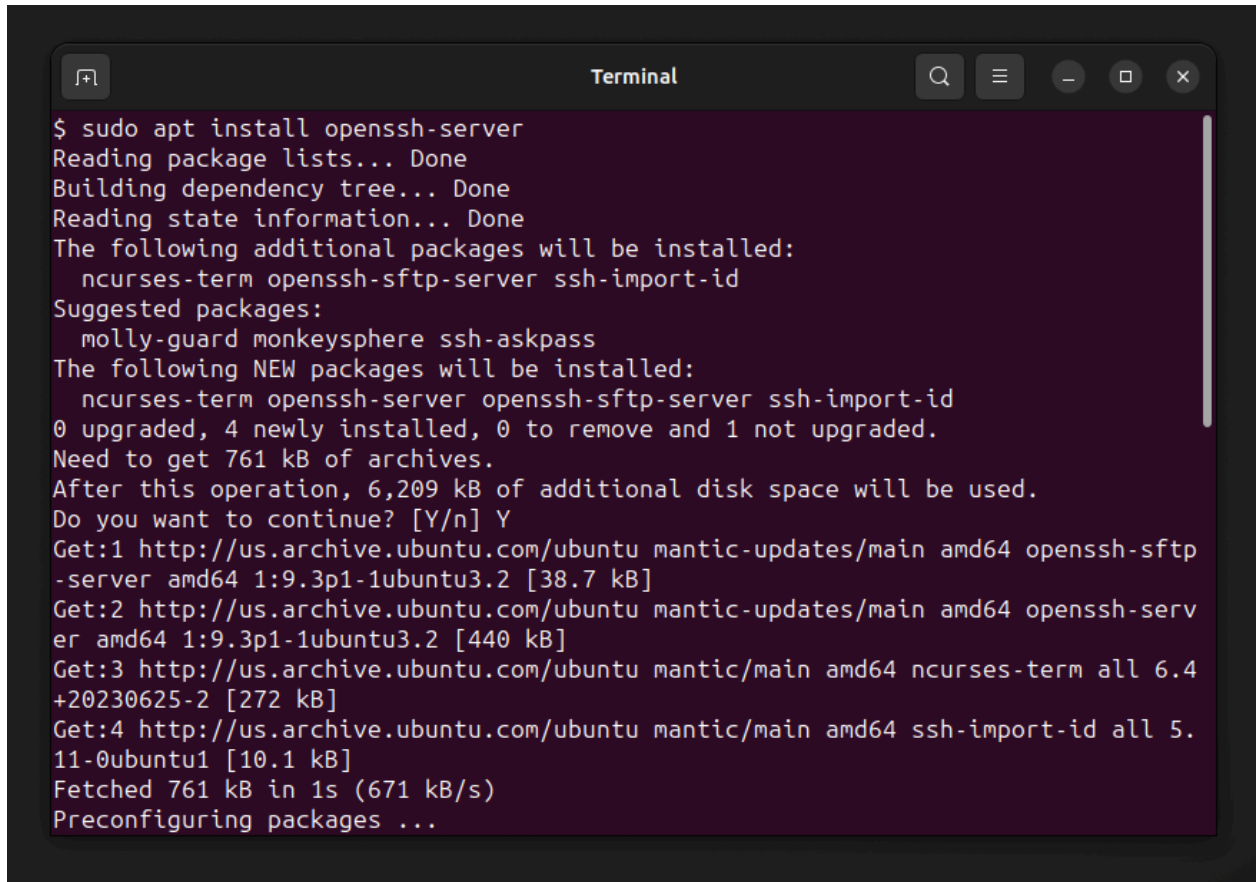
**Steps to install OpenSSH**

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:

A terminal window titled "Terminal" with a dark background and light text. It shows the command \$ sudo apt install openssh-server and its output. The output indicates that several additional packages will be installed along with the requested package. It shows the progress of downloading these packages from the Ubuntu repository, including the size of each package and the total disk space required. The installation is confirmed by typing 'Y' at the prompt. The terminal window has standard Ubuntu window controls at the top: a search icon, a menu icon, and window management buttons (minimize, maximize, close).

```
$ sudo apt install openssh-server
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  ncurses-term openssh-sftp-server ssh-import-id
Suggested packages:
  molly-guard monkeysphere ssh-askpass
The following NEW packages will be installed:
  ncurses-term openssh-server openssh-sftp-server ssh-import-id
0 upgraded, 4 newly installed, 0 to remove and 1 not upgraded.
Need to get 761 kB of archives.
After this operation, 6,209 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://us.archive.ubuntu.com/ubuntu mantic-updates/main amd64 openssh-sftp
-server amd64 1:9.3p1-1ubuntu3.2 [38.7 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu mantic-updates/main amd64 openssh-serv
er amd64 1:9.3p1-1ubuntu3.2 [440 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu mantic/main amd64 ncurses-term all 6.4
+20230625-2 [272 kB]
Get:4 http://us.archive.ubuntu.com/ubuntu mantic/main amd64 ssh-import-id all 5.
11-0ubuntu1 [10.1 kB]
Fetched 761 kB in 1s (671 kB/s)
Preconfiguring packages ...
```

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:

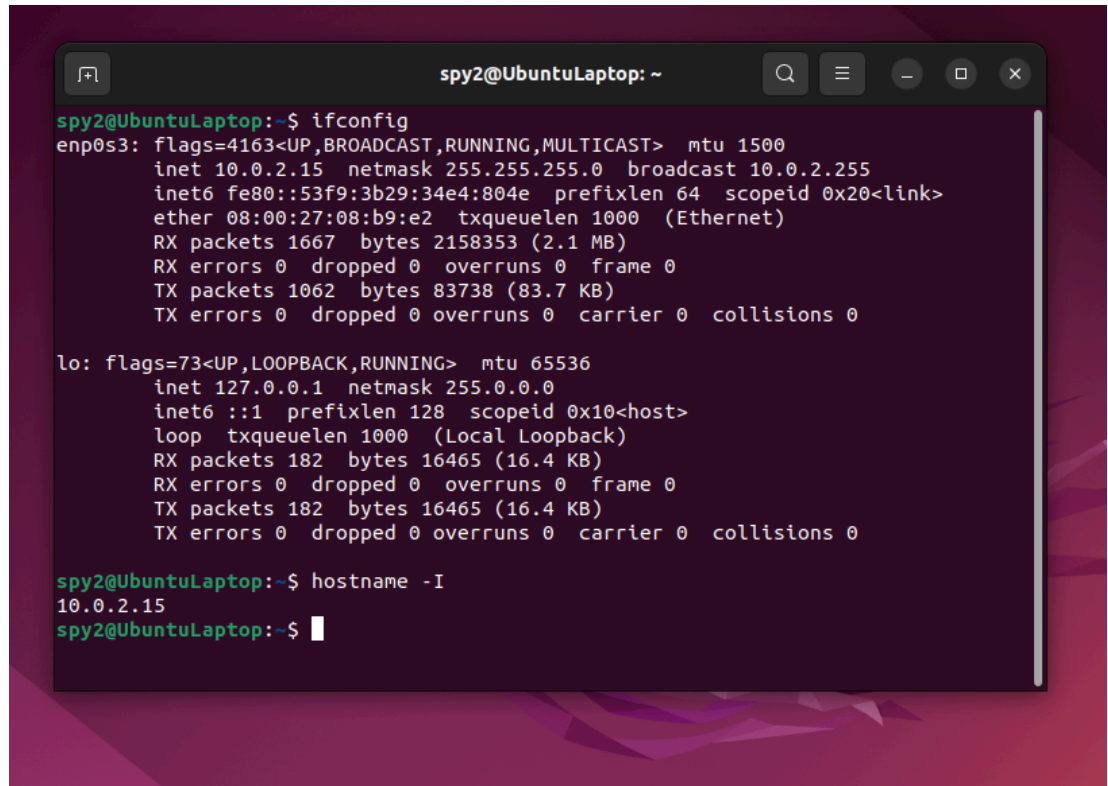
```
Terminal
$ sudo systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; disabled; preset: enabled)
   Drop-In: /etc/systemd/system/ssh.service.d
            └─00-socket.conf
   Active: active (running) since Mon 2024-02-19 11:54:25 PST; 58s ago
 TriggeredBy: ● ssh.socket
    Docs: man:sshd(8)
          man:sshd_config(5)
   Process: 11767 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
   Main PID: 11768 (sshd)
     Tasks: 1 (limit: 3385)
    Memory: 1.4M
       CPU: 21ms
    CGroup: /system.slice/ssh.service
            └─11768 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

Feb 19 11:54:25 MCyber systemd[1]: Starting ssh.service - OpenBSD Secure Shell
Feb 19 11:54:25 MCyber sshd[11768]: Server listening on :: port 22.
Feb 19 11:54:25 MCyber systemd[1]: Started ssh.service - OpenBSD Secure Shell
lines 1-19/19 (END)
```

(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):

A terminal window titled 'spy2@UbuntuLaptop: ~' with standard window controls. The terminal shows the output of the 'ifconfig' command for two interfaces: 'enp0s3' (Ethernet) and 'lo' (Loopback). The 'enp0s3' interface has IP 10.0.2.15, netmask 255.255.255.0, and MAC address 08:00:27:08:b9:e2. The 'lo' interface has IP 127.0.0.1 and netmask 255.0.0.0. Below this, the 'hostname -I' command is executed, returning '10.0.2.15'.

```
spy2@UbuntuLaptop:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
    inet6 fe80::53f9:3b29:34e4:804e  prefixlen 64  scopeid 0x20<link>
    ether 08:00:27:08:b9:e2  txqueuelen 1000  (Ethernet)
    RX packets 1667  bytes 2158353 (2.1 MB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 1062  bytes 83738 (83.7 KB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
    inet 127.0.0.1  netmask 255.0.0.0
    inet6 ::1  prefixlen 128  scopeid 0x10<host>
    loop txqueuelen 1000  (Local Loopback)
    RX packets 182  bytes 16465 (16.4 KB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 182  bytes 16465 (16.4 KB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

spy2@UbuntuLaptop:~$ hostname -I
10.0.2.15
spy2@UbuntuLaptop:~$
```

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

```
spy2@UbuntuLaptop: ~  
spy2@UbuntuLaptop:~$ ssh Spy1@192.168.1.26  
The authenticity of host '192.168.1.26 (192.168.1.26)' can't be established.  
ED25519 key fingerprint is SHA256:tTzN3LL/JHrYrGYeEt5ZXHI90hpVijE0h7XcbQn71J4.  
This key is not known by any other names  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added '192.168.1.26' (ED25519) to the list of known hosts.  
Spy1@192.168.1.26's password:  
Welcome to Ubuntu 23.10 (GNU/Linux 6.5.0-17-generic x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:        https://ubuntu.com/pro  
  
0 updates can be applied immediately.  
  
The programs included with the Ubuntu system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by  
applicable law.  
  
$ ls  
Desktop    Downloads  Pictures   Scripts   SpyInfo    Videos  
Documents  Music      Public    snap      Templates  
$
```

Screenshot (Connecting from desktop to laptop):

```
spy2@UbuntuLaptop: ~  
$ ssh spy2@192.168.1.27  
The authenticity of host '192.168.1.27 (192.168.1.27)' can't be established.  
ED25519 key fingerprint is SHA256:WP/F7aDqHezV+ZqkpS+/VCQTd2iUZnTtfV59gYetwvQ.  
This key is not known by any other names.  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added '192.168.1.27' (ED25519) to the list of known hosts.  
spy2@192.168.1.27's password:  
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 6.5.0-18-generic x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:        https://ubuntu.com/advantage  
  
Expanded Security Maintenance for Applications is not enabled.  
  
212 updates can be applied immediately.  
141 of these updates are standard security updates.  
To see these additional updates run: apt list --upgradable  
  
Enable ESM Apps to receive additional future security updates.  
See https://ubuntu.com/esm or run: sudo pro status  
  
The programs included with the Ubuntu system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by  
applicable law.  
  
spy2@UbuntuLaptop:~$
```

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):

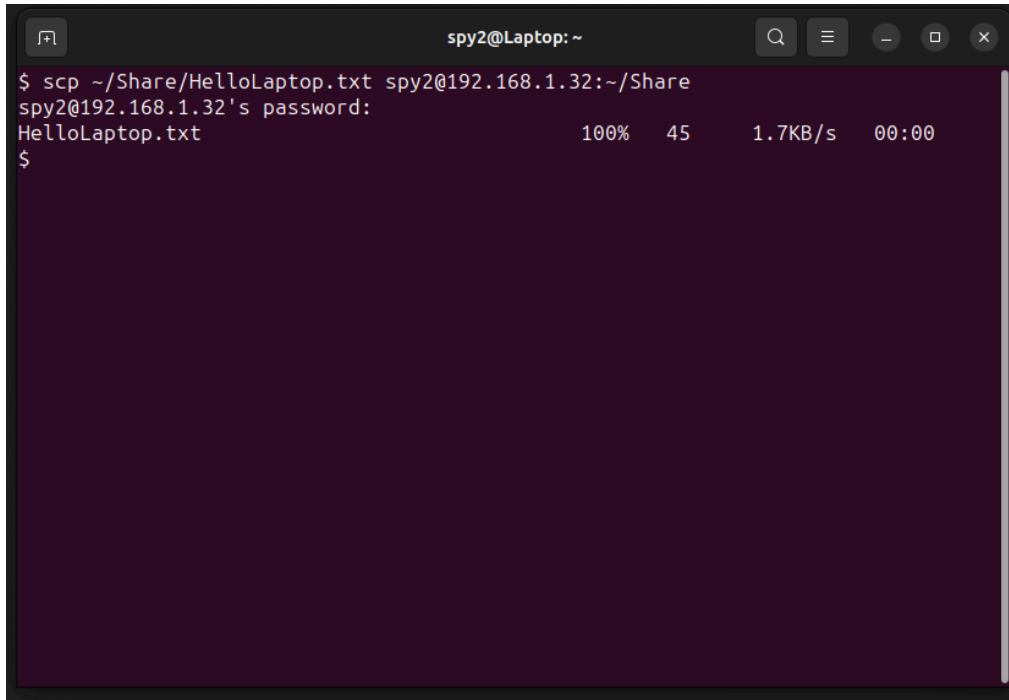
1. Task One: Transfer a file from my laptop VM to my Desktop VM using SCP.  
File transfer from laptop to desktop vm:

```
spy2@Laptop: ~  
$ exit  
Connection to 192.168.1.26 closed.  
spy2@Laptop:~$ ls  
Desktop  Downloads  Pictures  Scripts  Templates  
Documents Music      Public    snap      Videos  
spy2@Laptop:~$ mkdir Share  
spy2@Laptop:~$ ls  
Desktop  Downloads  Pictures  Scripts  snap      Videos  
Documents Music      Public    Share     Templates  
spy2@Laptop:~$ cd Shar  
bash: cd: Shar: No such file or directory  
spy2@Laptop:~$ cd Share  
spy2@Laptop:~/Share$ nano HelloDesktop.txt  
spy2@Laptop:~/Share$ cd  
spy2@Laptop:~$ scp ~/Share/HelloDesktop.txt Spy1@192.168.1.26:~/Share  
Spy1@192.168.1.26's password:  
HelloDesktop.txt                                100%  55    6.5KB/s   00:00  
spy2@Laptop:~$
```

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

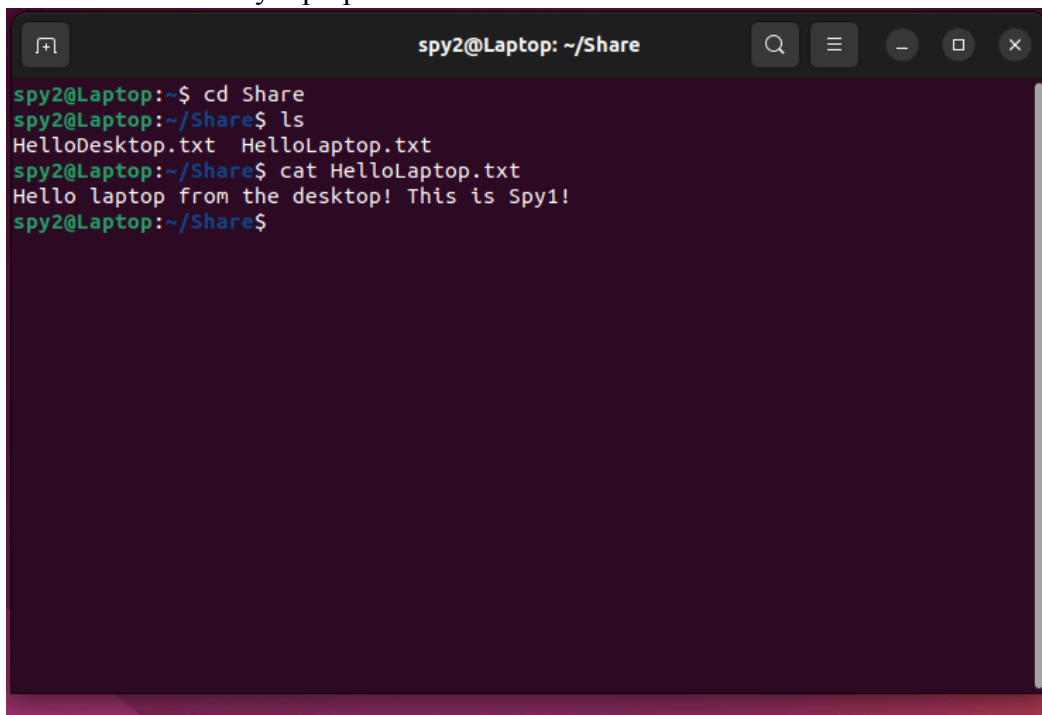
```
root@MCyber: /  
$ whaomi  
sh: 56: whaomi: not found  
$ whoami  
Spy1  
$ ls  
bin  cdrom  etc  lib  lost+found  mnt  proc  run  snap  swap.img  tmp  var  
boot  dev  home  lib64  media  opt  root  sbin  srv  sys  usr  
$ cd  
$ ls  
Desktop  Documents  Downloads  Music  Pictures  Public  Scripts  snap  SpyInfo  Templates  Videos  
$ mkdir Share  
$ ls  
Desktop  Documents  Downloads  Music  Pictures  Public  Scripts  Share  snap  SpyInfo  Templates  Videos  
$ cd Share  
$ ls  
HelloDesktop.txt  
$
```

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



```
spy2@Laptop: ~  
$ scp ~/Share/HelloLaptop.txt spy2@192.168.1.32:~/Share  
spy2@192.168.1.32's password:  
HelloLaptop.txt                                100% 45    1.7KB/s   00:00  
$
```

Now here it is on my laptop VM:



```
spy2@Laptop: ~/Share  
spy2@Laptop:~$ cd Share  
spy2@Laptop:~/Share$ ls  
HelloDesktop.txt  HelloLaptop.txt  
spy2@Laptop:~/Share$ cat HelloLaptop.txt  
Hello laptop from the desktop! This is Spy1!  
spy2@Laptop:~/Share$
```

## 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 use for the ssh service while preventing any conflicts with other services.  
Screenshot:

```
root@Laptop: ~  
root@Laptop:~# netstat -tuln  
Active Internet connections (only servers)  
Proto Recv-Q Send-Q Local Address           Foreign Address         State  
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN  
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN  
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN  
tcp6       0      0 :::1:631                :::*                    LISTEN  
tcp6       0      0 :::22                   :::*                    LISTEN  
udp        0      0 127.0.0.53:53           0.0.0.0:*                 
udp        0      0 0.0.0.0:631             0.0.0.0:*                 
udp        0      0 0.0.0.0:56105           0.0.0.0:*                 
udp        0      0 0.0.0.0:5353            0.0.0.0:*                 
udp6       0      0 :::40603                :::*                      
udp6       0      0 :::5353                 :::*                      
root@Laptop:~#
```

-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:

```
root@Laptop: /etc/ssh
GNU nano 6.2 ssh_config *
# StrictHostKeyChecking ask
# IdentityFile ~/.ssh/id_rsa
# IdentityFile ~/.ssh/id_dsa
# IdentityFile ~/.ssh/id_ecdsa
# IdentityFile ~/.ssh/id_ed25519
Port 2500
# Ciphers aes128-ctr,aes192-ctr,aes256-ctr,aes128-cbc,3des-cbc
# MACs hmac-md5,hmac-sha1,umac-64@openssh.com
# EscapeChar ~
# Tunnel no
# TunnelDevice any:any
# PermitLocalCommand no
# VisualHostKey no
# ProxyCommand ssh -q -W %h:%p gateway.example.com
# RekeyLimit 1G 1h
# UserKnownHostsFile ~/.ssh/known_hosts.d/%k
SendEnv LANG LC_*
HashKnownHosts yes
GSSAPIAuthentication yes

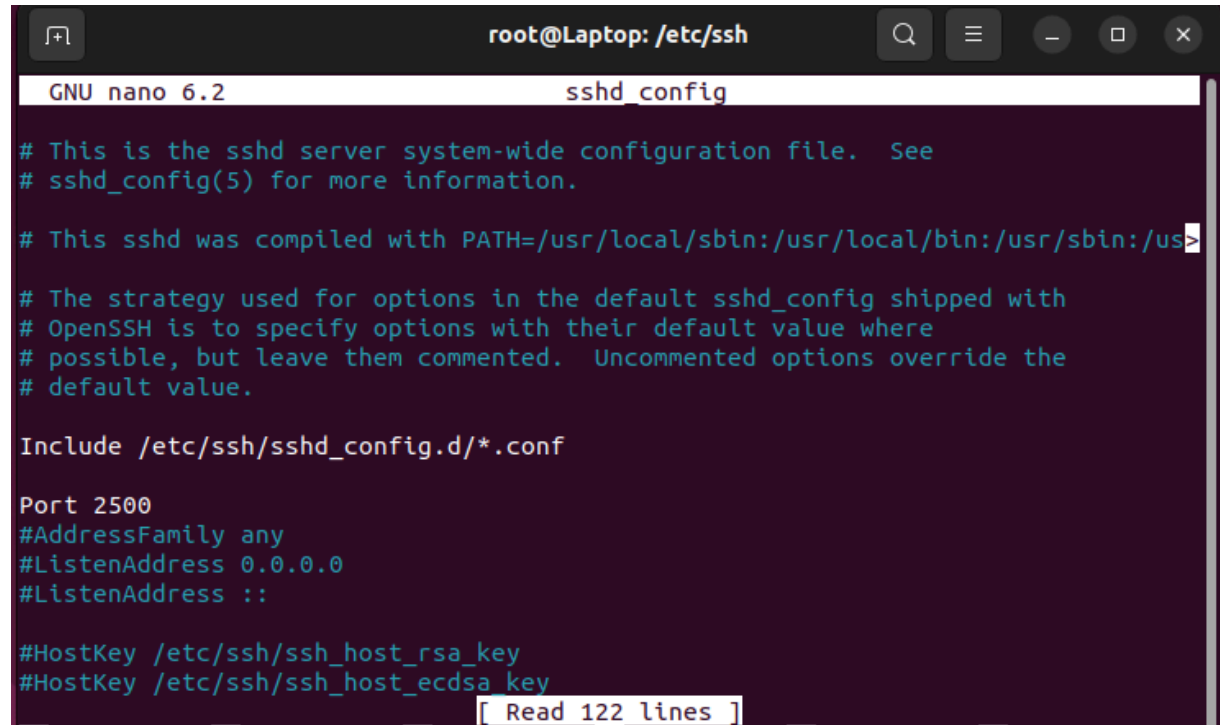
^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line
```

3. Restart the ssh service using systemctl restart ssh.  
Screenshot:

```
root@Laptop: ~
root@Laptop:~# systemctl restart ssh
root@Laptop:~# systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2024-03-16 14:46:21 PDT; 6s ago
     Docs: man:sshd(8)
           man:sshd_config(5)
   Process: 2273 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
  Main PID: 2274 (sshd)
    Tasks: 1 (limit: 3164)
   Memory: 1.7M
      CPU: 26ms
   CGroup: /system.slice/ssh.service
           └─2274 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

Mar 16 14:46:21 Laptop systemd[1]: Starting OpenBSD Secure Shell server...
Mar 16 14:46:21 Laptop sshd[2274]: Server listening on 0.0.0.0 port 2500.
Mar 16 14:46:21 Laptop sshd[2274]: Server listening on :: port 2500.
Mar 16 14:46:21 Laptop systemd[1]: Started OpenBSD Secure Shell server.
lines 1-17/17 (END)
```

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.

A screenshot of a terminal window titled 'root@Laptop: /etc/ssh'. The terminal shows the contents of the `/etc/ssh/sshd_config` file being edited with GNU nano 6.2. The file contains several commented-out lines and one active line: `Port 2500`. The terminal window has a dark background with light-colored text. The nano editor interface is visible at the top, showing the file name and line numbers. The bottom of the terminal shows the prompt `[ Read 122 lines ]`.

```
root@Laptop: /etc/ssh
GNU nano 6.2 sshd_config

# This is the sshd server system-wide configuration file.  See
# sshd_config(5) for more information.

# This sshd was compiled with PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr
# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented.  Uncommented options override the
# default value.

Include /etc/ssh/sshd_config.d/*.conf

Port 2500
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_ecdsa_key

[ Read 122 lines ]
```

## 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:

```
root@Laptop: /etc/ssh
GNU nano 6.2          sshd_config *

#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_ecdsa_key
#HostKey /etc/ssh/ssh_host_ed25519_key

# Ciphers and keying
#RekeyLimit default none

# Logging
#SyslogFacility AUTH
#LogLevel INFO

# Authentication:

#LoginGraceTime 2m
PermitRootLogin no
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10
```

Here is a test:

```
luser1@Laptop: ~
luser1@Laptop:~$ ssh -p 2500 root@Laptop
The authenticity of host '[laptop]:2500 ([127.0.1.1]:2500)' can't be established
.
ED25519 key fingerprint is SHA256:3xPqSWzXtsvR4RQf0Pre6/XZLjJ8pj1NkLOR5jmBQLw.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[laptop]:2500' (ED25519) to the list of known hosts.
WARNING! AUTHORIZED USERS ONLY! ALL OTHERS WILL BE PROSECUTED!
*****ALL YOUR ACTIONS WILL BE MONITORED!*****
root@laptop's password:
Permission denied, please try again.
root@laptop's password: █
```

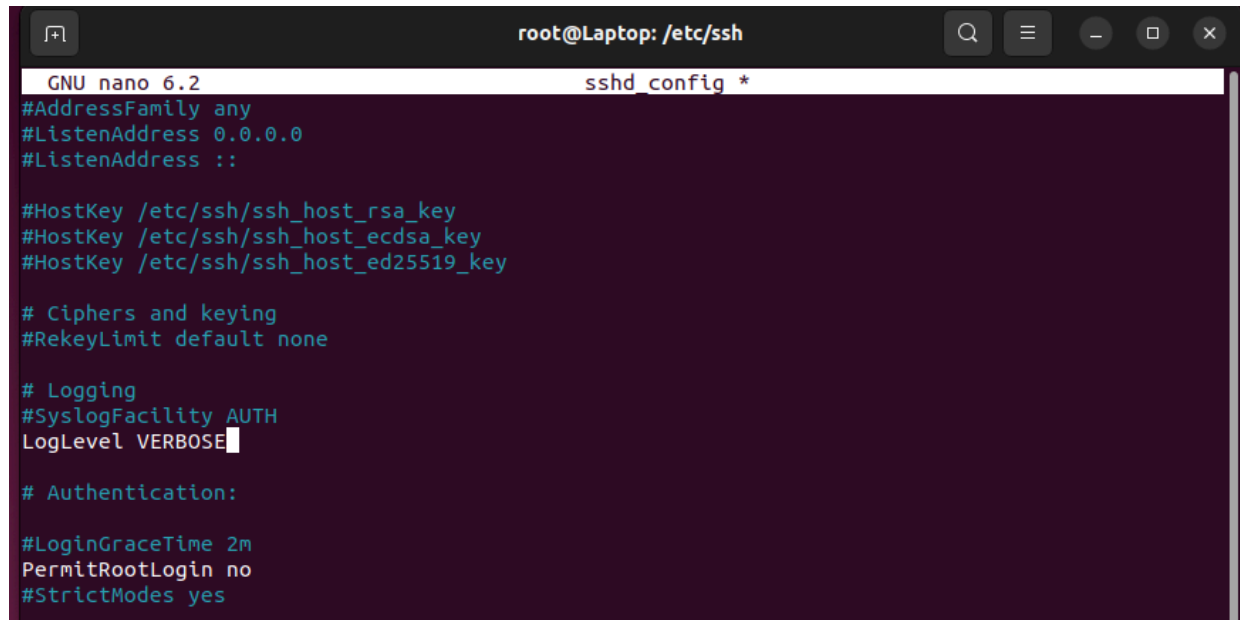
Root can't log in. I used the correct password, but permission was denied!

### 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:



```
root@Laptop: /etc/ssh
GNU nano 6.2 sshd_config *
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_ecdsa_key
#HostKey /etc/ssh/ssh_host_ed25519_key

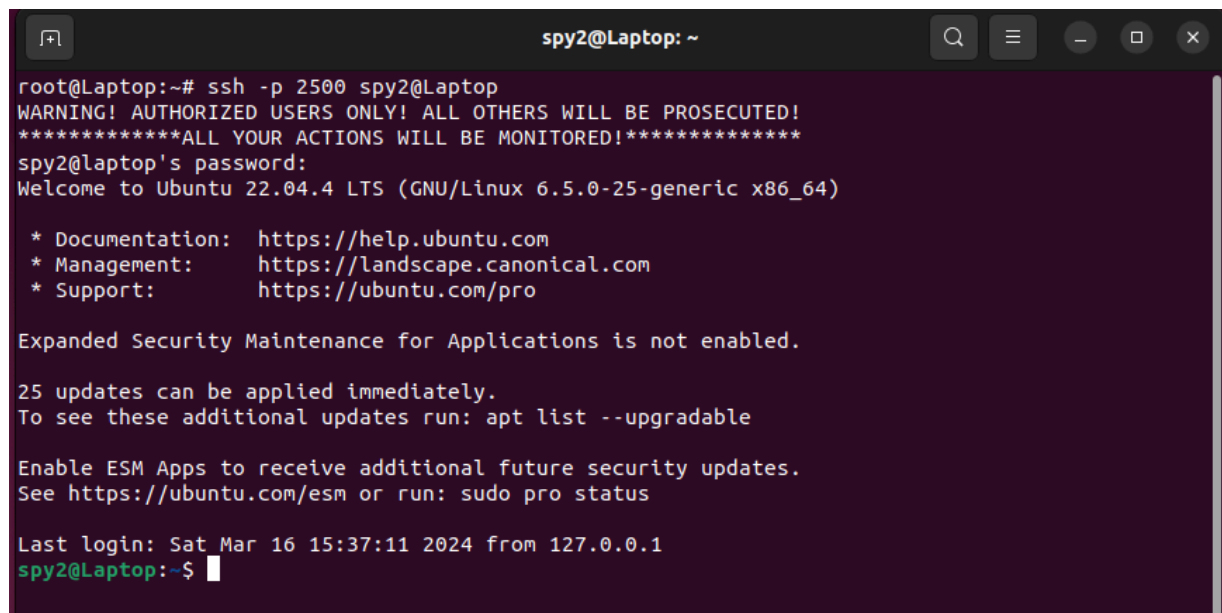
# Ciphers and keying
#RekeyLimit default none

# Logging
#SyslogFacility AUTH
LogLevel VERBOSE

# Authentication:

#LoginGraceTime 2m
PermitRootLogin no
#StrictModes yes
```

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:



```
spy2@Laptop: ~
root@Laptop:~# ssh -p 2500 spy2@Laptop
WARNING! AUTHORIZED USERS ONLY! ALL OTHERS WILL BE PROSECUTED!
*****ALL YOUR ACTIONS WILL BE MONITORED!*****
spy2@laptop's password:
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.5.0-25-generic x86_64)

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

Expanded Security Maintenance for Applications is not enabled.

25 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Last login: Sat Mar 16 15:37:11 2024 from 127.0.0.1
spy2@Laptop:~$
```

```
root@Laptop: /var/log
Mar 17 11:05:07 Laptop sshd[691]: Received signal 15; terminating.
Mar 17 11:05:07 Laptop sshd[2191]: Server listening on 0.0.0.0 port 2500.
Mar 17 11:05:07 Laptop sshd[2191]: Server listening on :: port 2500.
Mar 17 11:06:28 Laptop sshd[2209]: Connection from 127.0.0.1 port 45208 on 127.0.1.1 port 2500
rdomain ""
Mar 17 11:06:41 Laptop sshd[2209]: Accepted password for spy2 from 127.0.0.1 port 45208 ssh2
Mar 17 11:06:41 Laptop sshd[2209]: pam_unix(sshd:session): session opened for user spy2(uid=1001) by (uid=0)
Mar 17 11:06:41 Laptop systemd-logind[609]: New session 4 of user spy2.
Mar 17 11:06:41 Laptop systemd: pam_unix(systemd-user:session): session opened for user spy2(uid=1001) by (uid=0)
Mar 17 11:06:42 Laptop sshd[2209]: User child is on pid 2382
Mar 17 11:06:42 Laptop sshd[2382]: Starting session: shell on pts/1 for spy2 from 127.0.0.1 port 45208 id 0
Mar 17 11:07:07 Laptop dbus-daemon[574]: [system] Failed to activate service 'org.bluez': timed out (service_start_timeout=25000ms)
Mar 17 11:07:14 Laptop sshd[2382]: Received disconnect from 127.0.0.1 port 45208:11: disconnected by user
Mar 17 11:07:14 Laptop sshd[2382]: Disconnected from user spy2 127.0.0.1 port 45208
Mar 17 11:07:14 Laptop sshd[2209]: pam_unix(sshd:session): session closed for user spy2
Mar 17 11:07:14 Laptop systemd-logind[609]: Session 4 logged out. Waiting for processes to exit.
Mar 17 11:07:14 Laptop systemd-logind[609]: Removed session 4.
root@Laptop: /var/log#
```

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.

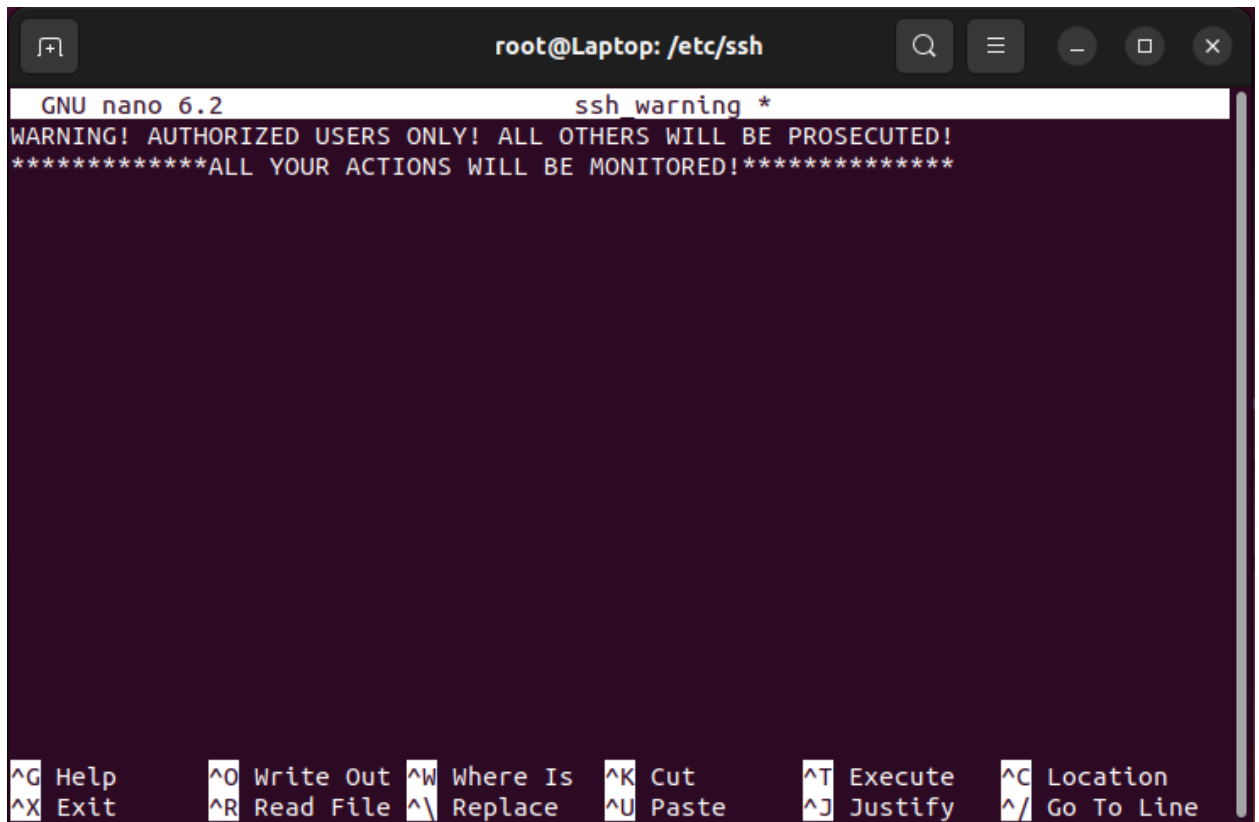
#### 4. Setup a custom login banner

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:

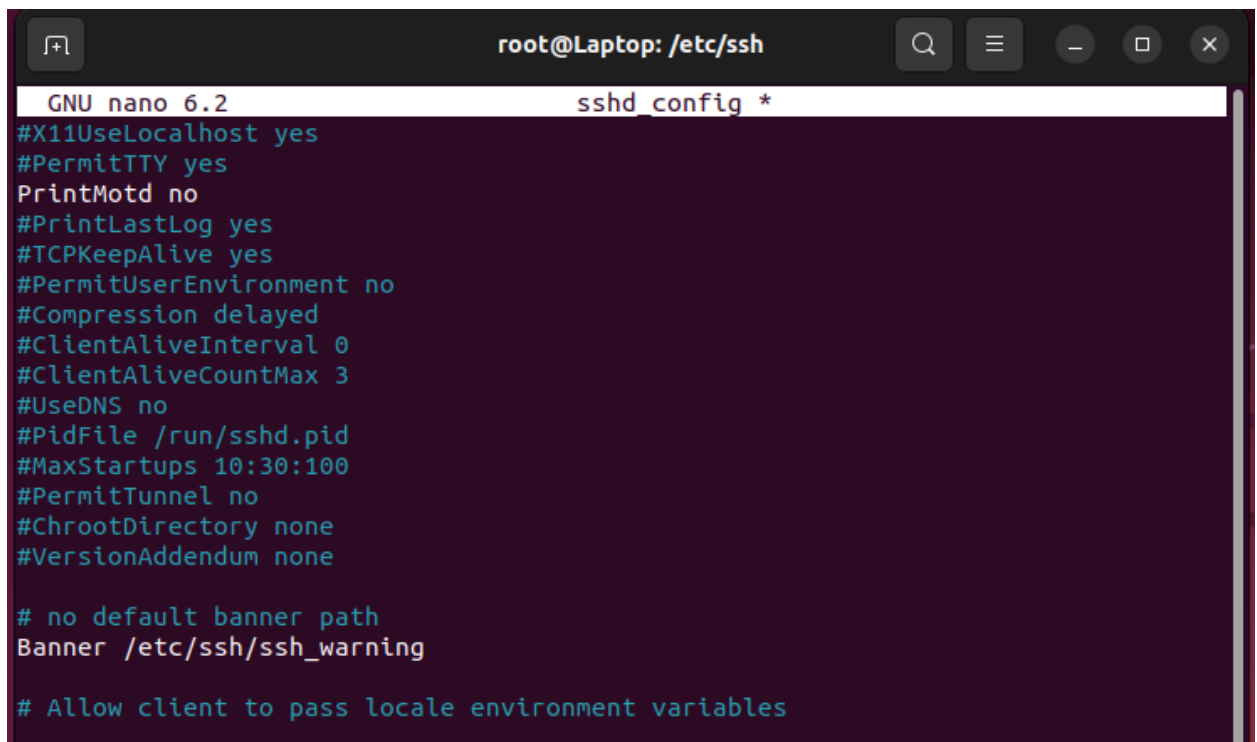


```
root@Laptop: /etc/ssh
GNU nano 6.2 ssh_warning *
WARNING! AUTHORIZED USERS ONLY! ALL OTHERS WILL BE PROSECUTED!
*****ALL YOUR ACTIONS WILL BE MONITORED!*****

^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^_ Go To Line
```

2. Add the path of the banner file onto the /etc/ssh/sshd\_config file.  
Screenshot:



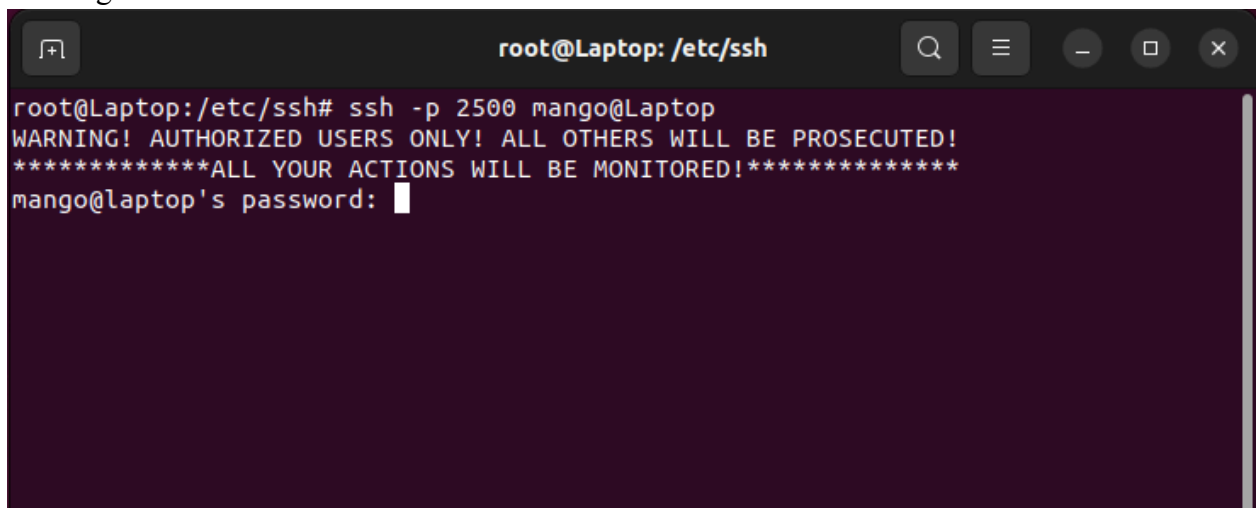


```
root@Laptop: /etc/ssh
GNU nano 6.2 sshd_config *
#X11UseLocalhost yes
#PermitTTY yes
PrintMotd no
#PrintLastLog yes
#TCPKeepAlive yes
#PermitUserEnvironment no
#Compression delayed
#ClientAliveInterval 0
#ClientAliveCountMax 3
#UseDNS no
#PidFile /run/sshd.pid
#MaxStartups 10:30:100
#PermitTunnel no
#ChrootDirectory none
#VersionAddendum none

# no default banner path
Banner /etc/ssh/ssh_warning

# Allow client to pass locale environment variables
```

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



```
root@Laptop: /etc/ssh# ssh -p 2500 mango@Laptop
WARNING! AUTHORIZED USERS ONLY! ALL OTHERS WILL BE PROSECUTED!
*****ALL YOUR ACTIONS WILL BE MONITORED!*****
mango@laptop's password: 
```

## 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:

```
root@Laptop: ~  
root@Laptop:~# apt install fail2ban  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
The following additional packages will be installed:  
  python3-pyinotify whois  
Suggested packages:  
  mailx monit sqlite3 python-pyinotify-doc  
The following NEW packages will be installed:  
  fail2ban python3-pyinotify whois  
0 upgraded, 3 newly installed, 0 to remove and 29 not upgraded.  
Need to get 473 kB of archives.  
After this operation, 2,486 kB of additional disk space will be used.  
Do you want to continue? [Y/n] Y  
0% [Connecting to banjo.canonical.com (91.189.91.38)]
```

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.conf is not created at first and must be created as a copy as jail.conf.

Screenshot:

```
root@Laptop: /etc/fail2ban  
root@Laptop:~# cd ..  
root@Laptop:/# cd etc  
root@Laptop:/etc# cd fail2ban  
root@Laptop:/etc/fail2ban# ls  
action.d      fail2ban.d    jail.conf     paths-arch.conf  paths-debian.conf  
fail2ban.conf filter.d      jail.d        paths-common.conf paths-opensuse.conf  
root@Laptop:/etc/fail2ban# cp jail.conf jail.local  
root@Laptop:/etc/fail2ban# ls  
action.d      fail2ban.d    jail.conf     jail.local        paths-common.conf  paths-opensuse.conf  
fail2ban.conf filter.d      jail.d        paths-arch.conf   paths-debian.conf  
root@Laptop:/etc/fail2ban# S
```

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.

```
root@Laptop: /etc/fail2ban
GNU nano 6.2 jail.local *
#
#
# SSH servers
#
[sshd]
enabled = true
port = ssh
backend = systemd
maxretry = 5
findtime = 300
bantime = 3600
logpath = /var/log/auth.log
filter = sshd
```

Setting	Value	Reasoning
enabled	true	I want fail2ban to monitor SSH or the sshd service.
port	ssh	I want fail2ban to 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.

```

spy2@Laptop: ~
spy2@Laptop:~$ sudo systemctl enable fail2ban.service
Synchronizing state of fail2ban.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable fail2ban
Created symlink /etc/systemd/system/multi-user.target.wants/fail2ban.service → /lib/systemd/system/fail2ban.service.
spy2@Laptop:~$ sudo systemctl status fail2ban.service
○ fail2ban.service - Fail2Ban Service
   Loaded: loaded (/lib/systemd/system/fail2ban.service; enabled; vendor preset: enabled)
   Active: inactive (dead)
     Docs: man:fail2ban(1)
spy2@Laptop:~$ sudo systemctl start fail2ban.service
spy2@Laptop:~$ sudo systemctl status fail2ban.service
● fail2ban.service - Fail2Ban Service
   Loaded: loaded (/lib/systemd/system/fail2ban.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2025-02-01 14:08:50 PST; 13s ago
     Docs: man:fail2ban(1)
  Main PID: 3653 (fail2ban-server)
    Tasks: 5 (limit: 4892)
   Memory: 35.1M
      CPU: 633ms
   CGroup: /system.slice/fail2ban.service
           └─3653 /usr/bin/python3 /usr/bin/fail2ban-server -xf start

Feb 01 14:08:50 Laptop systemd[1]: Started Fail2Ban Service.
Feb 01 14:08:50 Laptop fail2ban-server[3653]: Server ready
lines 1-13/13 (END)...skipping...
● fail2ban.service - Fail2Ban Service

```

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 military
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 qlen
    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:~$
```