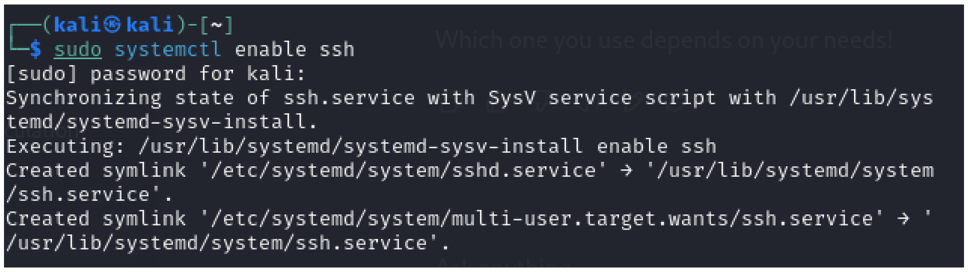
Task 2 :

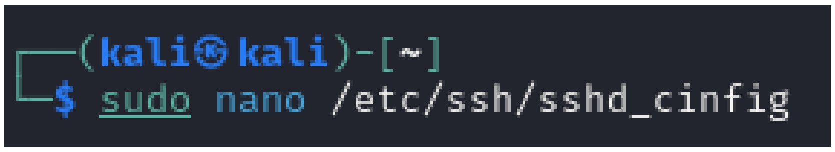
Remote Access & SSH Hardening

Setup: Enabling SSH & Weak Configuration 🔑 :



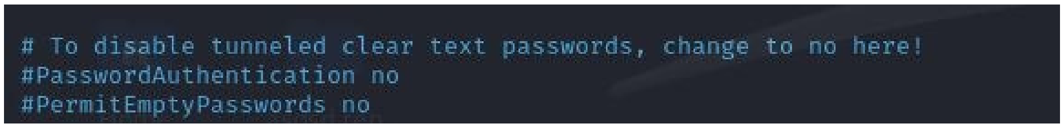
```
┌──(kali㉿kali)-[~]
└─$ sudo systemctl enable ssh
[sudo] password for kali:
Synchronizing state of ssh.service with SysV service script with /usr/lib/sys
temd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable ssh
Created symlink '/etc/systemd/system/sshd.service' → '/usr/lib/systemd/system
/ssh.service'.
Created symlink '/etc/systemd/system/multi-user.target.wants/ssh.service' → '
/usr/lib/systemd/system/ssh.service'.
```

To initiate the SSH service, we first enable it using `sudo systemctl enable ssh`,followed by `sudo systemctl start ssh` to ensure it is running and ready for remote access.

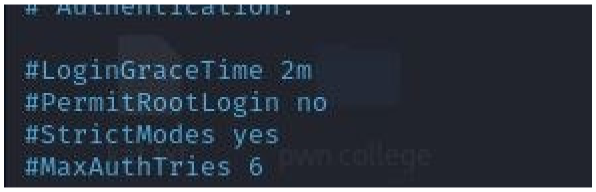


```
┌──(kali㉿kali)-[~]
└─$ sudo nano /etc/ssh/sshd_cinfig
```

Next, we modify the SSH configuration to permit root login and enable password authentication by editing the `/etc/ssh/sshd_config` file.



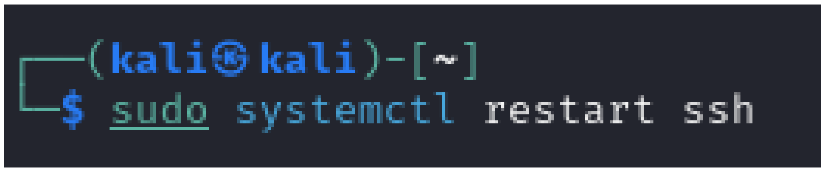
```
# To disable tunneled clear text passwords, change to no here!
#PasswordAuthentication no
#PermitEmptyPasswords no
```



```
# Authentication:

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

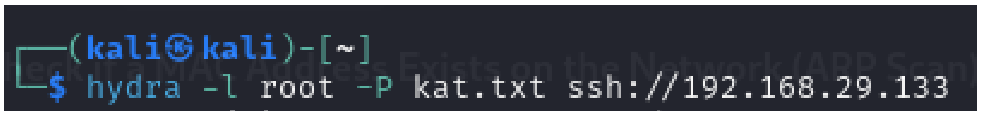
3. Update the `Per mitRootLogin` and `PasswordAuthentication` parameters to `yes.`



```
┌──(kali㉿kali)-[~]
└─$ sudo systemctl restart ssh
```

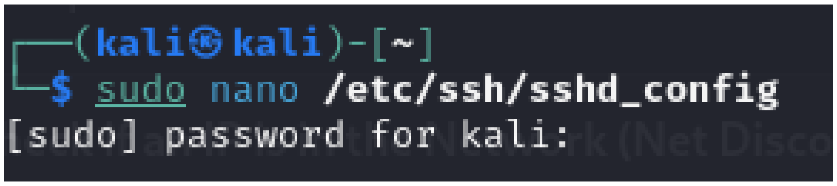
Then we restart the ssh service.

Exploitation: Brute-Forcing SSH🛠️:



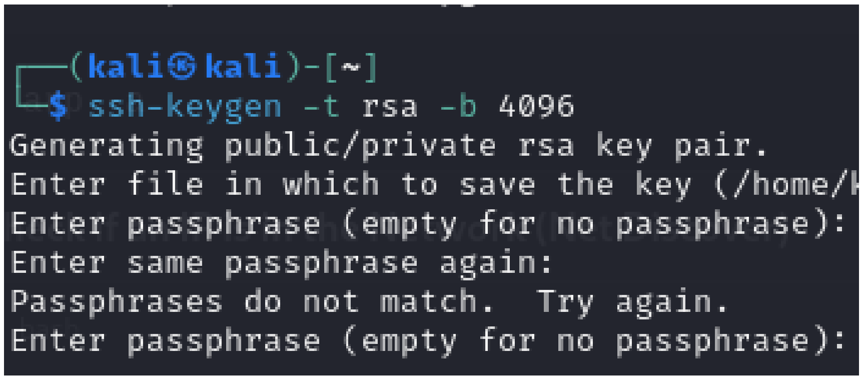
```
┌──(kali㉿kali)-[~]
└─$ hydra -l root -P kat.txt ssh://192.168.29.133
```

We use **Hydra** to perform a brute-force SSH root login using a custom-generated wordlist, targeting our own machine's IP address. This allows us to test authentication security and assess password strength.

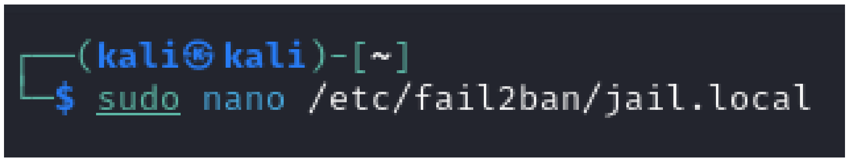


```
┌──(kali㉿kali)-[~]
└─$ sudo nano /etc/ssh/sshd_config
[sudo] password for kali:
```

To enhance security, root login and password authentication are disabled by setting PermitRootLogin no and PasswordAuthentication no in the SSH configuration file, followed by restarting the SSH service to apply the changes.



```
┌──(kali㉿kali)-[~]
└─$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/k
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Passphrases do not match.  Try again.
Enter passphrase (empty for no passphrase):
```

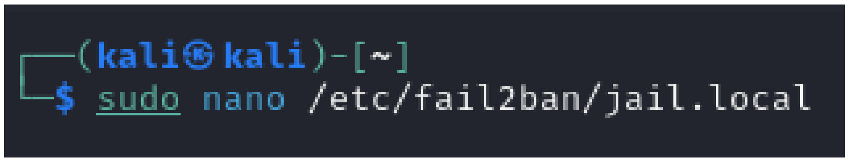


```
┌──(kali㉿kali)-[~]
└─$ sudo nano /etc/fail2ban/jail.local
```

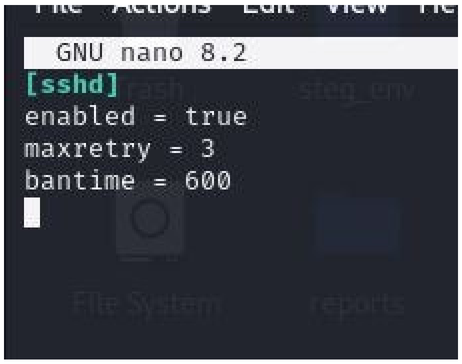
4 . To enhance authentication security, generate an SSH key pair on the client machine using ssh-keygen -t rsa -b 4096. Next, copy the public key to the server with ssh-copy-id user@<server-IP䷀, and finally, restart the SSH service using sudo systemctl restart ssh to apply the changes.

Configure Fail2Ban to Prevent Brute-Force Attacks

To enhance system security, install **Fail2Ban** by running sudo apt install fail2ban -y, which helps protect against brute-force attacks by monitoring and blocking suspicious login attempts.

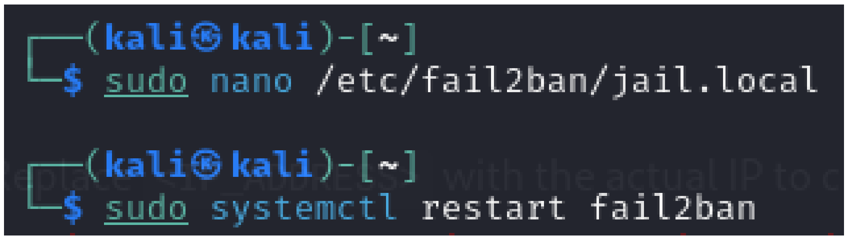


```
┌──(kali㉿kali)-[~]
└─$ sudo nano /etc/fail2ban/jail.local
```



```
  GNU nano 8.2
[sshd]
enabled = true
maxretry = 3
bantime = 600
```

To configure **Fail2Ban**, edit the jail configuration file using sudo nano /etc/fail2ban/jail.local, then add the following settings under [sshd]: enabled = true, maxretry ≣ 3, and bantime ≣ 600, ensuring protection against repeated failed SSH login attempts.



```
┌──(kali㉿kali)-[~]
└─$ sudo nano /etc/fail2ban/jail.local

┌──(kali㉿kali)-[~]
└─$ sudo systemctl restart fail2ban
```

Finally restart fail2ban to avoid ssh attacks.