

Task 2 : Remote Access & SSH Hardening

Setup: Enabling SSH & Weak Configuration :

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

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

(kali@kali)-[~]
$ sudo systemctl start ssh

(kali@kali)-[~]
$ sudo systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/usr/lib/systemd/system/ssh.service; enabled; preset: disabled)
   Active: active (running) since Sun 2025-03-23 07:49:42 EDT; 2min 35s ago
 Invocation: dcf8144147a84da2a445600c840989b3
    Docs: man:sshd(8)
          man:sshd_config(5)
   Process: 3460 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
  Main PID: 3461 (sshd)
    Tasks: 1 (limit: 3425)
  Memory: 1.9M (peak: 2.3M)
     CPU: 76ms
   CGroup: /system.slice/ssh.service
           └─3461 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

Mar 23 07:49:42 kali systemd[1]: Starting ssh.service - OpenBSD Secure Shell server...
Mar 23 07:49:42 kali sshd[3461]: Server listening on 0.0.0.0 port 22.
Mar 23 07:49:42 kali sshd[3461]: Server listening on :: port 22.
Mar 23 07:49:42 kali systemd[1]: Started ssh.service - OpenBSD Secure Shell server.
```

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

```
(kali@kali)-[~]
$ sudo nano /etc/ssh/sshd_config

(kali@kali)-[~]
$ sudo systemctl restart ssh
```

3. Find PermitRootLogin and PasswordAuthentication and give yes to this

```
#LoginGraceTime 2m
PermitRootLogin yes
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

#PubkeyAuthentication yes

# Expect .ssh/authorized_keys2 to be disregarded by default in future.
#AuthorizedKeysFile .ssh/authorized_keys .ssh/authorized_keys2

#AuthorizedPrincipalsFile none

#AuthorizedKeysCommand none
#AuthorizedKeysCommandUser nobody

# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

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

Exploitation: Brute-Forcing SSH

1. Create a wordlist and run hydra to Brute-force ssh

```
[kali@kali:~]$ hydra -L wordlist.txt -P wordlist.txt ssh://182.74.154.218
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these ** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-03-23 08:26:42
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[INFO] max 16 tasks per 1 server, overall 16 tasks, 25 login tries (1:5/n+5), ~2 tries per task
```

2. 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.

Mitigation:

1. Edit SSH Configuration:

Open the SSH configuration file and modify the following lines:

Disable Root Login and PasswordAuthentication

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

#PubkeyAuthentication yes

# Expect .ssh/authorized_keys2 to be disregarded by default in future.
#AuthorizedKeysFile .ssh/authorized_keys .ssh/authorized_keys2

#AuthorizedPrincipalsFile none
#AuthorizedKeysCommand none
#AuthorizedKeysCommandUser nobody

# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
```

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.

```
(kali@kali)-[~]
$ ssh-keygen -t rsa -b 4096

Generating public/private rsa key pair.
Enter file in which to save the key (/home/kali/.ssh/id_rsa): wordlist.txt
wordlist.txt already exists.
Overwrite (y/n)? y
Enter passphrase for "wordlist.txt" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in wordlist.txt
Your public key has been saved in wordlist.txt.pub
The key fingerprint is:
SHA256:0hBG6Q6Sy2CVUlt+S02RXuKpNeaPpSIWZ6JKhAmxtdw kali@kali
The key's randomart image is:
+--[RSA 4096]--+
| . 0 .. += ..
| * =.0* .
| o =oE+=. =
| +oo . o@
| ++ o o* S
| . oo +.o .
| .. = =
| .. o . o .
| o . . .
+--[SHA256]--+
```

Configure Fail2Ban to Prevent Brute-Force Attacks

1. 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 apt install fail2ban -y
Installing:
fail2ban

Installing dependencies:
python3-systemd

Suggested packages:
mailx system-log-daemon monit

Summary:
Upgrading: 0, Installing: 2, Removing: 0, Not Upgrading: 1175
Download size: 509 kB
Space needed: 2,601 kB / 62.9 GB available

Get:1 http://http.kali.org/kali kali-rolling/main amd64 python3-systemd amd64 235-1+b5 [42.7 kB]
Get:2 http://kali.download/kali kali-rolling/main amd64 fail2ban all 1.1.0-7 [466 kB]
Fetched 509 kB in 22s (22.9 kB/s)
Selecting previously unselected package python3-systemd.
(Reading database ... 401033 files and directories currently installed.)
Preparing to unpack .../python3-systemd_235-1+b5_amd64.deb ...
Unpacking python3-systemd (235-1+b5) ...
Selecting previously unselected package fail2ban.
Preparing to unpack .../fail2ban_1.1.0-7_all.deb ...
Unpacking fail2ban (1.1.0-7) ...
Setting up python3-systemd (235-1+b5) ...
Setting up fail2ban (1.1.0-7) ...
update-rc.d: We have no instructions for the fail2ban init script.
update-rc.d: It looks like a network service, we disable it.
fail2ban.service is a disabled or a static unit, not starting it.
Processing triggers for kali-menu (2024.4.0) ...
Processing triggers for man-db (2.13.0-1) ...
```

2. 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
```

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

3.Finally restart fail2ban to avoid ssh attacks.

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

7. Conclusion

Successfully set up SSH and performed a brute-force attack.

Implemented security measures including disabling root login, key-based authentication, and Fail2Ban.

Ensured SSH is hardened against unauthorized access attempts.