## Lab 14: How to SSH into a Server from a Linux Machine

### Scenario

As part of your penetration testing and system administration tasks, you often need to remotely connect to Linux servers. Secure Shell (SSH) is the most common method for encrypted remote login and management. While working on your Parrot Security OS, you are required to connect to a client’s Linux server, ensure proper key handling, and configure key-based authentication for secure, password-less access.

### Solution

The solution begins by ensuring the Ubuntu virtual machine is up and running, then updating the system packages with the command sudo apt update. After this, the OpenSSH server is installed and enabled to allow remote connections. Using the Parrot Security machine, a secure SSH connection is established to the Ubuntu server. If any host key mismatch warnings occur, the server’s SSH keys are regenerated, and old entries on the client are cleared. Finally, SSH key-based authentication is set up, allowing secure, password-less access for efficient and encrypted remote administration.

|  |  |
| --- | --- |
| **// Install OpenSSH Server**  1. Make sure your Ubuntu VM is up and running. Open **Terminal** and use command **sudo apt update** to update.    2. Use the command **sudo apt install -y openssh-server** to install ssh server.    3. To enable the ssh service use command **sudo systemctl enable --now ssh.**    4. To start the ssh service use the command **sudo systemctl status ssh.** The only thing that will change after a reboot is your **IP address**, so you will probably need to run the command **hostname -I** again before connecting from Parrot.    **// Connect SSH from Parrot**  1. On Parrot, type the command **ssh <ubuntu\_username>@<ubuntu\_192\_IP>**, replacing **username** and **IP** with your Ubuntu VM’s values.    2. The first time SSH connection will say something like:   |  | | --- | | The authenticity of host '192.168.1.25 (192.168.1.25)' can't be established.  ECDSA key fingerprint is SHA256:xxxxxxxxxxxxxxxx.  Are you sure you want to continue connecting (yes/no)? |   Type **Yes** to continue. It will then ask for your Ubuntu user’s password. If login succeeds, your prompt will change from **parrot@parrot:~$** to this **ubunto@ubunto:~$.** That means you are now controlling Ubuntu remotely from Parrot.    3. When you are done work use the command **exit** to exit back to parrot OS.    **// Regenerate the Server’s SSH Host Keys and Handle the Warning**  1. When you connect via SSH for the first time, Parrot stores the Ubuntu server’s fingerprint in **~/.ssh/known\_hosts.**  2. To connect to the remote server again with SSH, run the command **sudo rm /etc/ssh/ssh\_host\_\***    3. Regenerate remote server’s SSH keys with command **sudo dpkg-reconfigure openssh-server**. This will delete and regenerate all SSH host keys.    4. Restart the service to make sure the remote server is working with newly created SSH keys using command **systemctl restart ssh.** Disconnect from the remote server to re-establish the SSH connection. Each remote server operates with a unique SSH certificate. When trying to connect to another server with the same IP address as the remote server, the local SSH client will warn you about the situation in order to prevent user information from being stolen. Now, from parrot OS use command **ssh ubunto@192.168.191.143.** You should now see the **host key has changed** warning.    5. If it is definitely known that the operating system on the remote server has been reinstalled or if the SSH server keys are regenerated like in this example, it will be necessary to update the SSH key records on the local user who established the connection. In the local terminal, reconnect to the remote server via SSH using command **ssh-keygen -R 192.168.191.143.**    6. Then connect again using command **ssh ubunto@192.168.191.143.** Type **yes** when asked to trust the new key, enter your password, and you are back in.    **// SSH Key-Based Authentication**  1.If you make frequent SSH connections to the same server during the day, it is possible to gain password-free connection capability by copying your local SSH public key to the remote server. To manually add the content of the **parrot** user’s **.ssh/id\_rsa.pub** file to the **.ssh/authorized\_keys** file located in the remote user’s home directory. While on the terminal of the **Parrot** VM machine, as **parrot** user and use command **cd .ssh/** and then, **ssh-keygen.**    2. Now to copy your public key to ubuntu use command **ssh-copy-id username@<ubuntu\_ip>**. It will ask for your Ubuntu password just this one time.   3. To test passwordless login ssh ubunto@192.168.191.143. You should log in without being prompted for a password. As can be seen, no password is required for SSH logins. Even if the remote user changes their password, this will not prevent us from logging in without a password. If you were able to log into your account using SSH without a password, you have successfully configured SSH-key-based authentication to your account.   4. However, your password-based authentication mechanism is still active, meaning that your server is still exposed to brute-force attacks. In order to permanently block SSH access to the remote server with a password, it is necessary to edit the **sshd\_config** file and make **PasswordAuthentication** as **no**. The changes will take effect after the SSH service is restarted using command **systemctl restart ssh.** |