### **Lab 14– SSH into a Server from a Linux Machine**

This lab introduces you to using **SSH (Secure Shell)** from a Linux machine to connect to remote servers securely and manage them. It demonstrates key SSH features and practical tasks.

### **1. Purpose of SSH on Linux**

* **SSH in Linux:** Built-in functionality without requiring external applications.
* **Secure Communication:** Enables encrypted access to remote servers for management and file transfer.

### **2. Tools and Setup**

* **Kali Linux VM:** Acts as the client machine.
* **Ubuntu Server:** The SSH-enabled remote server for connections. Prebuilt images were used from [osboxes.org](https://www.osboxes.org/ubuntu-server/).

### **3. Lab Walkthrough**

#### **Task 1: Basic SSH Connection**

1. Open a terminal and run the command:  
   ssh osboxes@192.168.1.25

* Steps:
  1. Accept the server's key fingerprint by typing yes.
  2. Enter the password for the "osboxes" user (osboxes).
  3. A remote session is established.

#### **Task 2: Changing Server SSH Keys**

1. Regenerate server SSH keys:

sudo su -

rm -v /etc/ssh/ssh\_host\_\*

dpkg-reconfigure openssh-server

systemctl restart ssh

1. Update local SSH client records and reconnect:  
   ssh-keygen -f "/home/kali/.ssh/known\_hosts" -R "192.168.1.25"

ssh osboxes@192.168.1.25

This process prevents unauthorized access if the server's keys change.

#### **Task 3: Password-Free SSH with Key-Based Authentication**

1. Generate SSH keys locally:  
   Ssh-keygen
2. Copy the public key to the remote server:  
   ssh-copy-id osboxes@192.168.1.25
3. Disable password-based SSH logins by editing /etc/ssh/sshd\_config on the server:  
   PasswordAuthentication no

systemctl restart ssh

This strengthens security by relying only on key-based authentication.

#### **Task 4: SSH Tunneling for Port Forwarding**

1. Forward a local port to a remote destination:  
   ssh -L8080:192.168.1.1:80 osboxes@192.168.1.25
2. Access the remote service (192.168.1.1:80) locally via 127.0.0.1:8080.

#### **Task 5: Secure File Transfer**

1. Copy a file to the remote server using scp:  
   scp /etc/hosts osboxes@192.168.1.25:/tmp/
2. This command securely transfers files over the SSH tunnel.

#### 

#### **Task 6: Running SSH on a Custom Port**

1. Create a custom SSH configuration:  
   nano ~/.ssh/sshd\_config

Add:  
 Port 2222

UsePAM yes

HostKey ~/.ssh/id\_rsa

1. Start an SSH service listening on port 2222:  
   /usr/sbin/sshd -f ~/.ssh/sshd\_config
2. Connect to the custom port:  
   ssh -p 2222 osboxes@192.168.1.25

### **4. Key Takeaways**

* **SSH in Linux:** Offers built-in, versatile capabilities for remote server management.
* **Key-Based Authentication:** Enhances security and eliminates the need for passwords.
* **Tunneling and Port Forwarding:** Enables access to services securely through SSH.
* **Custom SSH Ports:** Adds flexibility and security by configuring alternative access points.

By completing this lab, you have gained advanced knowledge of managing and securing SSH connections from a Linux machine.