Preparation

Start the base image and install the required software. We will require the following packages:

- `openssh-server`
- `openssh-client`
- `wireshark`
- `apache2`
- `curl`

Run `sudo apt update` to update the package list from repositories. Then install the required software with:

```
sudo apt install openssh-server openssh-client wireshark apache2 curl
```

Note: While installing Wireshark, when asked **"Should non-superusers be able to capture packets?"**, select **Yes**. (If you select **No** by mistake, run the following command to change your selection: `sudo dpkg-reconfigure wireshark-common`.) Then add your user to the `wireshark` group with:

```
sudo usermod -a -G wireshark $USER
```

Shutdown the virtual machine.

Clone the Base Image

1. Clone the base image twice using **linked clone** and regenerate the MAC address.
2. Name the new machines **ssh-server** and **ssh-client**.

Configure Network Settings

1. Configure both machines to use a single **Network Interface Card (NIC)**:
    - Disable (if not already disabled) **Adapter 2**, **Adapter 3**, and **Adapter 4** in `Machine > Settings > Network`.
    - Ensure **Adapter 1** is enabled.
    - Use either **Bridged** mode or place both machines in the same **NAT network**.
2. Assert that the machines can **ping** each other.

Machine: ssh-server

Change the Hostname

1. Open the file `/etc/hosts` and add the following line:

    ```
    127.0.1.1 ssh-server
    ```

2. Save the file.
3. Run:

    ```
    sudo hostnamectl set-hostname ssh-server
    ```

4. Restart the terminal and observe that each line starts with `isp@ssh-server`.

Regenerate SSH Server Keys

Regenerate the SSH server keys for this machine. Use the following commands and provide an **empty passphrase** when asked. Name the keys according to the `HostKey` directive in `/etc/ssh/sshd_config`:

```
sudo ssh-keygen -t ecdsa -f /etc/ssh/ssh_host_ecdsa_key
sudo ssh-keygen -t rsa -f /etc/ssh/ssh_host_rsa_key
sudo ssh-keygen -t dsa -f /etc/ssh/ssh_host_dsa_key
sudo ssh-keygen -t ed25519 -f /etc/ssh/ssh_host_ed25519_key
```

Assignments

Assume:

- IP of **ssh-client**: `$CLIENT`
- IP of **ssh-server**: `$SERVER`

When running commands below, replace `$CLIENT` and `$SERVER` with actual IP addresses.

1. Username/Password Client Authentication, Server Authentication

1. On **ssh-client**, connect to the server:

   ```
   ssh isp@$SERVER
   ```

   Authenticate the server's public key:

   - Switch to the **ssh-server** machine.
   - Find the server's public key fingerprint:

     ```
     # ECDSA key
     ssh-keygen -lf /etc/ssh/ssh_host_ecdsa_key.pub
     ```

```
# ED25519 key
ssh-keygen -lf /etc/ssh/ssh_host_ed25519_key.pub

# RSA key
ssh-keygen -lf /etc/ssh/ssh_host_rsa_key.pub

# DSA key
ssh-keygen -lf /etc/ssh/ssh_host_dsa_key.pub
```

2. Switch back to **ssh-client**:
    - Verify that the displayed fingerprint matches the server's actual fingerprint.
    - If fingerprints mismatch, an **attack** (e.g., a man-in-the-middle attack) could be occurring.
3. Input `yes` if the fingerprint matches and provide the password. Observe that the terminal prompt changes from `isp@isp` to `isp@ssh-server`.
4. Log out with `exit`, `logout`, or `Ctrl+D`.

2. Change SSH Keypairs on ssh-server

1. Regenerate SSH keys on the server:

```
sudo ssh-keygen -t ecdsa -f /etc/ssh/ssh_host_ecdsa_key
sudo ssh-keygen -t rsa -f /etc/ssh/ssh_host_rsa_key
sudo ssh-keygen -t dsa -f /etc/ssh/ssh_host_dsa_key
sudo ssh-keygen -t ed25519 -f /etc/ssh/ssh_host_ed25519_key
```

2. On **ssh-client**, reconnect to the server. A warning will appear indicating the server's fingerprint has changed. Remove the saved fingerprints from `~/.ssh/known_hosts` and reconnect.
3. Authenticate the server's fingerprint or input `yes`.

3. Authenticating the Client with its Public Key

1. On **ssh-client**, regenerate user's SSH keys (do not use `sudo`):

```
ssh-keygen -t rsa
ssh-keygen -t dsa
```

```
ssh-keygen -t ecdsa
```

2. Connect to **ssh-server** and authenticate using the public key:

```
ssh -i ~/.ssh/id_rsa isp@$SERVER
```

3. If the server requests a password, public-key authentication failed. Debug using verbose mode (`-v` switch).

4. Enable public-key authentication:

```
ssh-copy-id isp@$SERVER
```

5. Reconnect to the server. You should log in without a password.

6. Disable password-based login on the server:

   - Open `/etc/ssh/sshd_config` and set:

     ```
     PasswordAuthentication no
     ```

   - Restart the SSH server:

     ```
     sudo service ssh restart
     ```

7. Test with:

```
ssh -o PreferredAuthentications=password -o PubkeyAuthentication=no
$SERVER
```

   The connection should be rejected.

4. Tunneling with SSH

1. Configure the Apache webserver on **ssh-server** to allow localhost connections only:

```
<Directory /var/www/html>
    Require ip 127.0.0.1/8
</Directory>
```

Reload Apache:

```
sudo service apache2 reload
```

2. On **ssh-client**, set up a tunnel:

```
ssh -L 127.0.0.1:8080:127.0.0.1:80 -N $SERVER
```

3. Open a new terminal and run:

```
curl localhost:8080
```

4. Check Apache access logs on **ssh-server**:

```
tail -f /var/log/apache2/access.log
```

Question: What is the IP address of the client issuing the HTTP requests? Why?

5. Reverse SSH Tunneling

1. Disable IPv6 on **ssh-server**:
   - Add the following to `/etc/sysctl.conf`:

```
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1
```

   - Apply changes:

```
sudo sysctl -p
```

2. Configure the firewall on **ssh-server**:

```
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -p tcp --dport 22 -m state --state NEW -j ACCEPT
```

3. Remove Apache access control on **ssh-server** and reload configuration:

```
sudo service apache2 reload
```

4. On **ssh-server**, set up a reverse tunnel:

```
ssh -R 127.0.0.1:8080:127.0.0.1:80 -N isp@$CLIENT
```

5. On **ssh-client**, access Apache pages:

```
curl localhost:8080
```

6. Use Wireshark to observe network messages during communication setup.

6. Explore scp and rsync

Learn and use the following commands to transfer files between **ssh-client** and **ssh-server**:

- `scp`: Secure copy (remote file copy program)
- `rsync`: Fast, versatile remote (and local) file-copying tool