# Lab 4 - SSH

Kathara SSH.pdf

SSH connection
- Connect to a server
- Creating an user on the server
- Mnemonica assignments
- Genere a key for ssh connection

Port Forwarding
- Local Port Forwarding
- Remove Port Forwarding

## SSH connection

### Connect to a server

- **ssh username@<host_ip_address> -p 443**: to connect with ssh to <host_ip_address> device. The flag -p is not necessary, normally it run of port 22 ( with TCP ) and the flag have 22 as default

### Creating an user on the server

In home folder we shoudl see all users of the device

```
/etc/init.d/networking restart
/etc/init.d/ssh restart # for when no wan set psw ?
```

```
useradd <ssh_user> -m
echo -e 'ilovessh\nilovessh\n' | passwd ssh_user
```

- **useradd <ssh_user> -m:** create and user named ssh_user, the flag -m create all necessary folders
- **echo -e 'ilovessh\nilovessh\n' | passwd ssh_user:** set password **ilovessh** in startup file ( essentially it run the command passwd <ssh_user> and later write twice the password and you are doing on the terminal )
- **passwd <ssh_user>:** command to change in live the password for log-in ssh_user

## Mnemonica assignments

if we don't want write the whole if when connecting to server, we can write on startup file of the HOST echo command

```
/etc/init.d/networking restart

echo "10.0.1.100 s1" > /etc/hosts
```

in this case we can connect in two ways

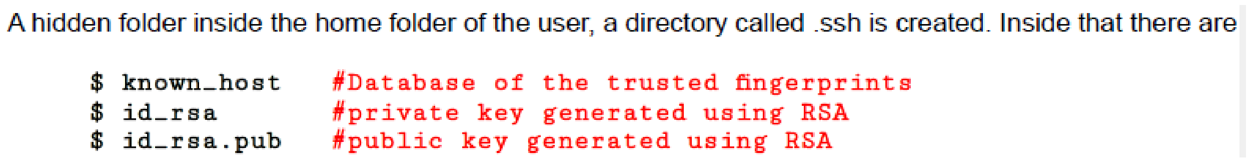
- **ssh username@10.0.1.100**
- **ssh username@s1**

WATCH OUT: if you put before the networking restart, it will remove the echo command

## Genere a key for ssh connection

Given client **pc1** and server **s1**

1. **key generation:**
    a. **ssh-keygen:** to be done on CLIENT to generate a key pair
    b. it will create and hidden folder inside the home folder
2. **copying public key from client to server**
    a. **scp .ssh/id_rsa.pub ssh_user@s1:/home/ssh_user/client_key.pub** ( this is the needed one )



A hidden folder inside the home folder of the user, a directory called .ssh is created. Inside that there are

```
$ known_host     #Database of the trusted fingerprints
$ id_rsa         #private key generated using RSA
$ id_rsa.pub     #public key generated using RSA
```

## Port Forwarding

It is used to redirect packet on a ssh connection

### Local Port Forwarding

1. **Ensure the connection is not possible:** Use Netcat to test the connection to the target IP and port: `nc <ip_dest> <port>` Example: `nc 10.0.3.3 88`
2. **Establish an SSH tunnel for local port forwarding:** Use the following command to set up port forwarding: `ssh -NL local_port:dest_ip_addr:dest_port user@<tunnel_ip>` Example: `ssh -NL -v 5000:10.0.3.5:8080 user@10.0.2.100`
    a. Explanation:
        i. **local_port**: Port on your local machine that will listen for incoming traffic (e.g., 5000).

ii. **dest_ip_addr**: Target server's IP address where the traffic will be forwarded (e.g., 10.0.3.5).

iii. **dest_port**: Target server's port where the traffic will be redirected (e.g., 8080).

iv. **user@tunnel_ip**: Credentials and IP of the intermediate machine (e.g., user@10.0.2.100).

v. **Flags:**

1. `N`: Do not execute remote commands, only establish the tunnel.
2. `L`: Specify local port forwarding.
3. `v`: Enable verbose mode for debugging.

b. Example Explanation:

a. Traffic coming to **port 5000** on your local machine will be sent to **user@10.0.2.100**.

b. **user@10.0.2.100** will forward the traffic to **10.0.3.5** at port **8080**.

3. **Connect to the target server via the forwarded port**

- Open another terminal and use Netcat to connect to the local port: `nc localhost 5000` This connects to the target server (10.0.3.5:8080) through the established SSH tunnel.

💡 This method is to avoid blocks on certain region, bypass firewalls, etc...

## Remove Port Forwarding

- **Command for Remote Port Forwarding**:
  - `ssh -NR remote_port:dest_ip_addr:dest_port user@<tunnel_ip>`

- **Example:** `ssh -v -NR 5247:localhost:6000 188.217.70.88`
    - **Explanation**:
        - **remote_port**: Port on the remote server that will listen for incoming traffic (e.g., 5247).
        - **dest_ip_addr**: IP address of the destination machine where the traffic will be forwarded (e.g., localhost, representing your local computer).
        - **dest_port**: Port on the destination machine where the traffic will be redirected (e.g., 6000).
        - **user@<tunnel_ip>**: Credentials and IP of the remote server (e.g., `188.217.70.88`).
        - **Flags**:
            - `N`: Do not execute remote commands, only establish the tunnel.
            - `R`: Specify remote port forwarding.
            - `v`: Enable verbose mode for debugging.
- **Example Behavior**:
    - The **remote port 5247** is opened on the remote server **188.217.70.88**.
    - Any traffic arriving at **188.217.70.88:5247** will be tunneled back to the destination **localhost (your computer)** on port **6000**.

This setup allows external devices to connect to a service running locally on your computer through the remote server's open port.