Placement Empowerment Program

Cloud Computing and DevOps Centre

Up SSH Key-Based Authentication Locally: Generate an SSH key pair and configure it for passwordless login between two local machines or VMs

Name: Jayadasan S Department: CSE

Introduction

Secure Shell (SSH) is a powerful protocol for securely accessing remote systems over a network. Typically, SSH uses a username and password for authentication, but SSH key-based authentication offers a more secure and convenient alternative. It involves generating a pair of cryptographic keys—private and public—where the private key stays on the client machine, and the public key is placed on the target machine. This allows users to log in without entering a password, streamlining automation and access control. This Proof of Concept (PoC) focuses on setting up SSH key-based authentication between two local machines or VMs, enabling passwordless login for streamlined and secure remote access.

Overview

In this PoC, you'll go through the following steps:

1. **Generate an SSH Key Pair**: The process starts with creating an RSA key pair (private and public keys) using the ssh-keygen command. The private key is stored securely on the source machine, while the public key is transferred to the target machine.

2. **Copy the Public Key to the Target Machine**: The public key will be copied to the target machine's ~/.ssh/authorized_keys file. This step can be done using ssh-copy-id or manually by copying the contents of the public key file.

3. **Set Correct File Permissions**: The target machine's SSH service requires that the ~/.ssh directory and authorized_keys file have specific file permissions to function securely. Permissions will be adjusted to ensure SSH works correctly.

4. **Test Passwordless SSH Login**: After completing the configuration, you will test the setup by attempting an SSH login from the source machine to the target machine without entering a password.

Objectives

1. **Generate an SSH Key Pair**: Learn how to generate a 2048-bit RSA key pair for SSH key-based authentication.

2. **Transfer the Public Key**: Gain hands-on experience with transferring the public key to the target machine and adding it to the authorized keys for passwordless login.

3. **Configure Permissions**: Understand the importance of setting correct file permissions for the ~/.ssh directory and authorized_keys file to secure SSH access.

4. **Test SSH Connectivity**: Verify that SSH key-based authentication works by testing passwordless login between the two machines.

Importance

1. **Security**: SSH key-based authentication is more secure than traditional password-based methods because it's resistant to bruteforce attacks. The private key is never transmitted over the network, reducing the risk of interception.

2. **Automation**: For environments that require frequent or automated logins (such as scripts or DevOps pipelines), passwordless SSH login is essential for smooth operations without manual intervention.

3. **Streamlined Access**: Passwordless login simplifies the process of accessing remote machines, reducing the need to manually input passwords, and enabling seamless automation for system administration tasks.

4. **Best Practice**: Using SSH keys for authentication is widely considered a best practice in the industry, as it provides both security and convenience.

Step-by-Step Overview

Step 1:

Generate SSH Key Pair

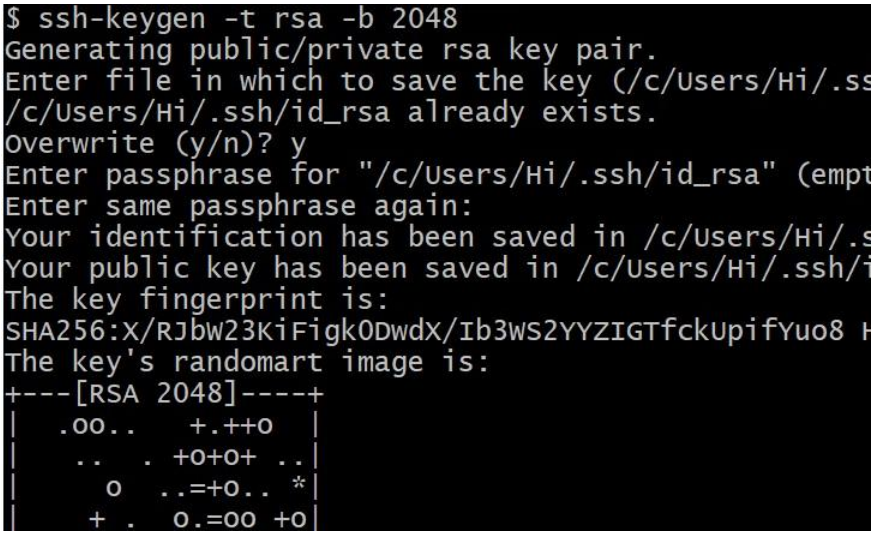
1. Open Git Bash or Terminal on the local machine from which you'll be accessing the target machine (VM or another local machine).

2. Run the following command to generate an SSH key pair:

   ssh-keygen -t rsa -b 2048

   This will create a 2048-bit RSA key pair. You will be prompted for a file to save the key. Press **Enter** to use the default path (~/.ssh/id_rsa).

   Optionally, you can set a passphrase for extra security, but if you want passwordless login, leave it empty and press **Enter**.



Step 2:

Copy Public Key to Target Machine

1. Now, you need to copy the public key (id_rsa.pub) to the target machine (the one you're trying to log into).

2. Use the ssh-copy-id command to copy the public key:

   ssh-copy-id username@target-machine-ip

   Replace username with the actual user on the target machine, and target-machine-ip with the target machine's IP address.

   If ssh-copy-id isn't available, you can manually copy the public key:

   - On the source machine, display the public key:

     cat ~/.ssh/id_rsa.pub

   - On the target machine, open the ~/.ssh/authorized_keys file (create it if it doesn't exist):

     nano ~/.ssh/authorized_keys

- Paste the public key into the authorized_keys file and save it.

Step 3:

Set Correct File Permissions

Ensure the correct file permissions are set for SSH to function properly:

On the target machine, set the correct permissions for the ~/.ssh directory and authorized_keys file:

chmod 700 ~/.ssh chmod 600 ~/.ssh/authorized_keys

Step 4:

Test Passwordless SSH Login

Now that the key is copied and permissions are set, you can test passwordless SSH login from the source machine:

ssh username@target-machine-ip

You should be able to log in without entering a password.

This sets up passwordless SSH login between the two machines.

Outcomes

By completing this PoC on setting up SSH key-based authentication locally, you will:

1. **Generate and Manage SSH Keys**: Gain hands-on experience in generating and managing SSH key pairs using ssh-keygen, enhancing your understanding of public-key cryptography and its role in secure authentication.

2. **Configure Passwordless Login**: Successfully configure passwordless SSH login between two local machines or VMs, improving security and convenience by eliminating the need for password-based authentication.

3. **Understand SSH Permissions and Security**: Learn to set correct file permissions for the ~/.ssh directory and authorized_keys file, reinforcing security best practices for SSH access.

4. **Improve Efficiency in Remote Access**: Streamline remote access operations by enabling seamless, automated logins, which are essential for automation scripts, administrative tasks, and DevOps pipelines.

5. **Enhance Security Skills**: Strengthen your foundational security skills by implementing a secure authentication mechanism that reduces exposure to brute-force attacks and credential theft.