

SSH Checklist

SSH Key Setup Checklist

1. Generate SSH Key Pair

- Ensure that you have generated an SSH key pair on the client machine (`system a`).
 - Run the following command and check for the existence of `~/.ssh/id_rsa` (private key) and `~/.ssh/id_rsa.pub` (public key):

```
ls -l ~/.ssh/id_rsa ~/.ssh/id_rsa.pub
```

- If the keys do not exist, generate them using:

```
ssh-keygen -t rsa -b 4096
```

- Ensure the keys have appropriate permissions (600 for private and 644 for public keys):

```
chmod 600 ~/.ssh/id_rsa  
chmod 644 ~/.ssh/id_rsa.pub
```

2. Copy Public Key to Remote Server

- Ensure that the public key has been copied to `system b` and appended to the `~/.ssh/authorized_keys` file.
 - Use the `ssh-copy-id` command to copy the key:

```
ssh-copy-id username@system_b
```

- Alternatively, you can manually copy the contents of `~/.ssh/id_rsa.pub` to `~/.ssh/authorized_keys` on `system b`.
- Verify that the `authorized_keys` file has the correct permissions:

```
chmod 600 ~/.ssh/authorized_keys
```
- Make sure the `.ssh` directory has the correct permissions:

```
chmod 700 ~/.ssh
```

3. Check SSH Daemon Configuration on Remote Server

- Ensure the SSH daemon is configured to allow key-based authentication. Check `/etc/ssh/sshd_config` on `system b` for the following settings:
 - `PubkeyAuthentication yes`
 - `AuthorizedKeysFile .ssh/authorized_keys`
 - `PasswordAuthentication no` (optional, for enforcing key-based auth only)
- Restart the SSH daemon to apply changes:

```
sudo systemctl restart sshd
```

4. Verify Ownership and Permissions

- Check ownership and permissions of the user's home directory on `system b`:
 - Ensure that the home directory and `.ssh` directory are owned by the user and have appropriate permissions:

```
chown -R username:username /home/username  
chmod 700 /home/username/.ssh
```

5. Ensure Correct SSH Command Usage

- Use the correct username and hostname when attempting to SSH into `system b`:

```
ssh username@system_b
```

- If using a non-standard SSH port, specify it using the `-p` option:

```
ssh -p <port_number> username@system_b
```

6. Check SSH Agent (Optional)

- Ensure that the SSH agent is running and the key is added to it, especially if using `ssh-agent` for key management:

```
eval "$(ssh-agent -s)"  
ssh-add ~/.ssh/id_rsa
```

- Verify the keys added to the agent:

```
ssh-add -l
```

7. Review SSH Debugging Output

- Use the `-v` option with the SSH command to enable verbose mode, which can provide detailed debugging information:

```
bash
```

```
ssh -v username@system_b
```

- Look for lines indicating whether the key is being offered and any errors that may indicate why the key is being rejected.

8. Network Issues

- Ensure there are no network issues preventing the connection. Check that `system_b` is reachable via the network.
- Verify that the firewall on `system_b` is allowing incoming SSH connections on the specified port.

9. SELinux and AppArmor (Linux Specific)

- If `system_b` uses SELinux, ensure it is not blocking SSH connections:

```
sudo setenforce 0 # Temporarily disable for testing
```

- If using AppArmor, check that it allows SSHD to read the `authorized_keys` file.

10. Check for Multiple Keys (Optional)

- If multiple keys are present, specify the exact key to use:

```
ssh -i ~/.ssh/id_rsa username@system_b
```

By following this checklist, you should be able to identify and resolve most issues preventing SSH key-based authentication. If problems persist, examining the verbose output from the SSH command can provide further clues.

Updated 2024-08-04 17:18:41 UTC by Max