# SSH CCDC Recipe

## Strategy for SSH in CCDC

New competitors in their first CCDC competition often struggle to properly secure their SSH servers, including the author of this guide. Nothing is more of a let down than asking the red team how they hacked you and learning that they had an ssh key in your ~/.ssh/authorized_keys file the whole time. Another big danger is that improper SSH setup can lock you out of a server  for good.

- All SSH Users should require both a key and a password for access
- All keys should be encrypted
- All keys should have copies on multiple other machines
- If the SSH service is scored, add an exception for only the users that are scored using the “Match User” directive in sshd_config. Make sure to add “Match all” at the end of the match block.
- If SSH is for service, like Git, make sure that it is a different SSH server
- Scored SSH users should not have access to `sudo` unless they need it
- When changing the config, always run “`sudo sshd -t`” before you reload
- If it’s a remote server, always maintain an open SSH session when you change the config, and test login before you exit that session.
- Getting logging data from SSH in Kibana is a priority
- Check “`.ssh/authorized_keys`” for unknown keys frequently. The red team loves adding them to the root account especially.

`> ssh-keygen -f <name of key>`
`> ssh-copy-id -i <public key> <user>@<ssh server>`
    This puts the public key in the `~/.ssh/authorized_keys`
Edit `/etc/ssh/sshd_config` with the options in the table (page 2):
`> sudo sshd -t` to test the configs
`> w` to check for malicious users
`> less /home/*/.ssh/*` to check the home directories of all users for SSH keys/config
- use :n to move through the files

`> systemctl reload sshd` (stopping won’t kick you out, but it will not allow any new sessions - be careful)
`> cat /etc/passwd` to check which users have login shells

| Option | Recommend Value | Description |
|---|---|---|
| **`PermitRootLogin`** | `no` | Can the root user log in with SSH |
| **`PasswordAuthentication`** | `yes` | Can passwords be used as one of the authentication methods? |
| **`PubkeyAuthentication`** | `yes` | Can keys be used as one of the authentication methods? |
| **`X11Forwarding`** | `no` | Allows some X11 GUI apps to operate remotely. This is usually pointless and increases attack surface. |
| **`AuthenticationMethods`** | `publickey,password` | The important one. Requires that both a key and a password are used for login, instead of one or the other. |
| `AllowUsers` | | A whitelist of permitted users. It's fine to leave these blank on their own line. Users can be comma separated. |
| `DenyUsers` | | A blacklist of users who cannot log in. |
| `AllowGroups` | | A whitelist of permitted user groups. |
| `DenyGroups` | | A blacklist of groups whose members should not log in. |
| `ForceCommand` | | Override the user's login shell from `/etc/passwd` |

**Prevent anyone new from logging in**

- As root, create the empty file `/etc/nologin`.
- As long as this file exists, no one new can log in except for root, but users with existing sessions are not affected.
- Delete the file to allow logins once more
- This may be useful to implement at the start of the competition, when you are still configuring SSH

**View active SSH sessions**

- Use the `w` command
- Use the `last` command

**Kill a specific SSH session**

- When a users opens a new SSH session, a specific sshd process starts, running as that user, that is the parent to other processes a user starts, like `bash`. To kill a specific session, you need to find the PID of this sshd process and kill it (`kill -9 <PID>`).
- Try running `pstree -up | less` and looking around for sshd's running as specific users.
- Try looks at the PTS. Run `ps aux | grep pts`, then look for any sshd processes. You can get your own PTS number with `tty`, so you can avoid killing your own session. Note that programs like TMUX open up new pts as well.
- `> sudo ps aux | grep pts | grep ssh | grep <user>`

## Server Configuration File

- The configuration file for an SSH server will typically be located at `/etc/ssh/sshd_config`
- When you make a change, on most distros you can run `systemctl reload ssh`, instead of `systemctl restart ssh`, applying the config change without totally restarting the server. This will preserve active SSH sessions. It is recommended to edit ssh settings with an active session, than try to log in with a new session while keeping the first open, for safer testing of your new settings.

**Basics**

- To go straight through a jumpbox: `> ssh -J <jumpbox user>@<jumpbox> <other user>@<other server>`
- To run a command remotely and then disconnect: `ssh <user>@<host> "<command>".` This is often how the NECCDC scoring engine works.

**SSH_CONFIG example-** just use ssh <nickname> to automatically ssh

```
Host <nickname for user>
    HostName <ip of host>
    User <username>
    Port <port>
    IdentityFile <private key location>
```