

SSH Client and Host Onboarding Guide

Quickest

ul

```
sudo -iu akadmin vw unlock
```

n00b

```
cat /etc/ssh/ssh_host_ed25519_key
```

ul

```
vim ~/.ssh/authorized_keys  
# type-y type-y
```

n00b

```
ssh -i /etc/ssh/ssh_host_ed25519_key ul  
# test ^  
cat /etc/ssh/ssh_host_ed25519_key.pub | ssh ul "sudo sign-host-key n00b -l n00b0x" | sudo tee  
/etc/ssh/ssh_host_ed25519_key-cert.pub  
sudo chmod 644 /etc/ssh/ssh_host_ed25519_key-cert.pub  
printf 'HostCertificate /etc/ssh/ssh_host_ed25519_key-cert.pub\n' | sudo tee  
/etc/ssh/sshd_config.d/10-cert.conf  
sudo sshd -T -f /etc/ssh/sshd_config | grep -q HostCertificate || echo fail  
sudo systemctl reload sshd
```

UNLOCK VAULTWARDEN

The CA signing keys reside in Vaultwarden and are kept in volatile memory for 1 hour once unlocked. Before onboarding, log into **ul** and unlock the vault:

```
sudo -iu akadmin vw unlock
```

WORKFLOW 1: ONBOARDING A NEW HOST (Server Host Certificate)

Host certificates allow clients to verify the authenticity of a server automatically without manual fingerprint checks ("The authenticity of host ... can't be established").

1. Generate Host Keypair on New Machine

On the machine being onboarded "n00b", generate its standard Ed25519 host key (if it does not already exist):

```
sudo ssh-keygen -t ed25519 -f /etc/ssh/ssh_host_ed25519_key -N ""
```

2. Out-of-Band Bootstrap & Verification

Verify it using `authorized_keys` first:

1. Copy the n00b's public key text (`/etc/ssh/ssh_host_ed25519_key.pub`) out-of-band
2. Add it to `~/.ssh/authorized_keys` on ul.
3. Test SSH connectivity from the new host to **ul**:

```
ssh -i /etc/ssh/ssh_host_ed25519_key ul
```

Verified!

3. Mint Signed Host Certificate

From the new host, stream the public key over that verified SSH connection directly into `sign-host-key`:

```
cat /etc/ssh/ssh_host_ed25519_key.pub | ssh ul "sudo sign-host-key bmo95 -l studio-pc" | sudo tee /etc/ssh/ssh_host_ed25519_key-cert.pub
```

- **Custom Principals:** To specify custom DNS names or IP addresses:

```
cat /etc/ssh/ssh_host_ed25519_key.pub | ssh -t ul "sudo sign-host-key danknasty
danknasty,danknasty.drheck.dev,10.0.0.1,fd77:7777:5010::1" | sudo tee
/etc/ssh/ssh_host_ed25519_key-cert.pub
```

- Set file permissions on the new host:

```
sudo chmod 644 /etc/ssh/ssh_host_ed25519_key-cert.pub
```

4. Configure sshd on the New Host

Add the host certificate to the SSH daemon configuration on the new host (`/etc/ssh/sshd_config` or `/etc/ssh/sshd_config.d/10-cert.conf`):

```
HostCertificate /etc/ssh/ssh_host_ed25519_key-cert.pub
```

Reload the SSH daemon:

```
sudo systemctl reload sshd
```

5. De-provision Bootstrap Key

Remove the temporary bootstrap entry from `~/.ssh/authorized_keys` on `ul`.

WORKFLOW 2: ONBOARDING A NEW USER (Client User Certificate)

User certificates allow clients to log into servers without putting per-user public keys in `~/.ssh/authorized_keys`.

1. Generate User Keypair on Client Workstation

On your laptop or workstation:

```
ssh-keygen -t ed25519 -f ~/.ssh/id_ed25519 -C "workstation-label"
```

2. Out-of-Band Bootstrap & Verification

1. Copy `~/.ssh/id_ed25519.pub` out-of-band and temporarily append it to `~/.ssh/authorized_keys` on `ul`.

2. Confirm login works:

```
ssh ul
```

3. Mint Signed User Certificate

From your client workstation, pipe your public key into `sign-user-key`:

```
cat ~/.ssh/id_ed25519.pub | ssh -t ul "sudo sign-user-key <ldap_username>" >
~/.ssh/id_ed25519-cert.pub
chmod 644 ~/.ssh/id_ed25519-cert.pub
```

Note: The principal `<ldap_username>` must exactly match the Authentik/LDAP username on **ul**.

4. De-provision Bootstrap Key

Remove the raw public key from `~/.ssh/authorized_keys` on **ul**.

5. Verify Certificate Authentication

Test login with verbose output:

```
ssh -v ul
```

Confirm the server accepts the certificate:

```
Server accepts key: pkalg ssh-ed25519-cert-v01@openssh.com
```

CLIENT CONFIGURATION: TRUSTING THE HOST CA

To ensure client workstations trust all servers signed by the DrHeck Host CA without fingerprint prompts, add this line to `/etc/ssh/ssh_known_hosts` (system-wide) or `~/.ssh/known_hosts`:

```
@cert-authority * ssh-ed25519
AAAAC3NzaC1lZDI1NTE5AAAAIC0m1wzW2e50529d+E3iA7x8iM3Q8qS5q7iVjGvF8e7d
```

(Replace with the actual public key from `/etc/ssh/drheck_ca_host_key.pub`).

QUICK REVOCATION REFERENCE

Every signed key is automatically archived in `/etc/ssh/ca_inventory/`.

- **List all signed keys:**

```
revoke-key --list
```

- **Revoke a key immediately:**

```
sudo revoke-key <label>
```

WORKFLOW 3: ANDROID / TERMUX ONE-TOUCH QR ONBOARDING

For Android devices using Termux, an entire SSH CA environment (device Ed25519 keypair, CA-signed user certificate, DrHeck Host CA trust, and SSH configuration) can be minted and imported using a single QR code.

1. Unlock Vaultwarden on `ul`

```
sudo -iu akadmin vw unlock
```

2. Mint & Render QR Code on `ul`

Run `/home/luke/bin/ssh-addclient`:

```
# Syntax: ssh-addclient <device_name> [ldap_username]
ssh-addclient pixel9 luke
```

This utility:

- Generates a private Ed25519 keypair in RAM (`/dev/shm`).
- Signs the user certificate via `sign-user-key` with the DrHeck User CA.
- Embeds the Host CA `@cert-authority` trust and pre-configured hosts (`ul`, `danknasty`).
- Compresses and encodes the payload into a self-extracting Termux bootstrap one-liner.
- Renders an ASCII QR code directly in the terminal.

3. Ingest in Termux (Android)

In Termux on the phone: In Termux with WireGuard connected:

```
pkg install openssh  
curl -s http://10.0.0.3:7700 | sh
```

(The one-shot listener runs strictly on the internal WireGuard interface `10.0.0.3:7700` and shuts down immediately upon retrieval).

4. Connect

```
ssh ul  
# or  
sftp ul
```

Authentication succeeds immediately using the CA user certificate with zero host fingerprint prompts.

Revision #5

Created 2026-09-03 12:54:32 UTC by Guest

Updated 2026-09-26 17:19:42 UTC by luKe