Table of Contents

Introduction

Smallstep's article on SSH certificates convinced me that they're a good idea, but their documentation left me somewhat unclear on how to make them work. After some discussion with them, and some figuring things out on my own, I have this much, at least, worked out.

These instructions assume that you already have a Smallstep certificate authority set up, and that it's configured to issue both x.509 certificates and SSH certificates. If you don't, consult the Smallstep docs and blog posts.

SSH certificates can certify a host or a user. These instructions will cover each, separately.

Install the step CLI Utility

You'll need to install the Step CLI utility on any computer (client or server) you want to use with these certificates, and use that utility to “log in”, as it were, to your CA. To install the utility, consult the Smallstep docs.

In order to configure systems to use your CA, you'll need the hostname or IP address of your CA, and the fingerprint of the root signing cert. If you don't have the latter, you can get it by logging into the CA, finding that certificate (ordinarily it will be at `/etc/step-ca/certs/root_ca.crt`), and running `step certificate fingerprint root_ca.crt`.

Once installed, configure the system to use your CA by running `step ca bootstrap --install --ca-url ca.hostname --fingerprint root_certificate_fingerprint`. The output should look like this:

```
[root@neth-lemon ~]# step ca bootstrap --install --ca-url ca.familybrown.org --fingerprint 2f477e4bd5cddf3908521e57f4884247388123be6c1faae80caf883c1b2a3153
The root certificate has been saved in /root/.step/certs/root_ca.crt.
Your configuration has been saved in /root/.step/config/defaults.json.
Installing the root certificate in the system truststore... done.
```

Repeat this on every client and server you want to use with your SSH certificate authority

SSH Host Certificates

Using SSH host certificates will require you to configure any client systems to trust the SSH CA, and then obtaining the certificate on the host, and configuring the host to use that certificate.

Client configuration

On your client system, run `step ssh config --host --roots` to retrieve the host signing certificate. The output will look like this:

```
 dan@Dan-Hack-Mini  ~  step ssh config --host --roots
ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBLP6daZnfOGuHpcjFI2N6VI3dr
W9eH0lqCJP9S+xRfvovpHK134EFj7w5b/yHsfssrZJ4/bkNf8JIcKt9qa34vQ=
```

Then edit your `~/.ssh/known_hosts` file, adding this line:

```
@cert-authority *.yourdomain ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBLP6daZnfOGuHpcjFI2N6VI3dr
W9eH0lqCJP9S+xRfvovpHK134EFj7w5b/yHsfssrZJ4/bkNf8JIcKt9qa34vQ=
```

Naturally, you'll replace the key information above with the information you retrieved using `step ssh config --host --roots`.

Repeat this on any client system/user you want to trust your SSH CA

Obtaining the cert

On a host system, change to the `/etc/ssh` directory and run `step ssh certificate --insecure --no-password --host (your hostname) ssh_host_ecdsa_key2`. If you want this certificate to be valid for multiple names (for example, the host's bare hostname and also its FQDN), add both with `-n` flags to the command like this: `step ssh certificate --insecure --no-password -n $hostname -n $fqdn --host (your hostname) ssh_host_ecdsa_key2`.

You'll be asked to choose a “provisioner”; choose the first one on the list (it should be identified with `(JWK)`). You'll then need to enter the password for that provisioner; this would be the password you chose when you initially set up that provisioner (when you set up the CA). This command will create three files: `ssh_host_ecdsa_key2`, `ssh_host_ecdsa_key2.pub`, and `ssh_host_ecdsa_key2-cert.pub`.

Renewing the cert

The certificate you've just obtained is short-lived; it will need to be renewed frequently. Therefore, you'll want to automate this task using either a cron job or a systemd unit.

Cron job

Create a script at `/etc/cron.daily/ssh-cert-renew`. Its contents should be:

```
#!/bin/sh
step ssh renew --force /etc/ssh/ssh_host_ecdsa_key2-cert.pub
/etc/ssh/ssh_host_ecdsa_key2
service sshd restart
```

Make it executable with `chmod +x /etc/cron.daily/ssh-cert-renew`.

systemd unit

If your system runs systemd, as most modern Linux distributions do, you can instead set up the daily certificate renewal using a systemd timer. This will require creating both a service and a timer file, and then enabling the latter.

Service file

Create `/etc/systemd/system/ssh-host-cert.service` with the following contents:

```
# Renew SSH host certificate
#

[Unit]
Description=Renew SSH host certificate
Wants=ssh-host-cert.timer

[Service]
Type=oneshot
ExecStart=/usr/local/bin/step ssh renew --force /etc/ssh/ssh_host_ecdsa_key2-
cert.pub /etc/ssh/ssh_host_ecdsa_key2
ExecStart=/bin/systemctl restart sshd

[Install]
WantedBy=multi-user.target
```

Timer file

Create `/etc/systemd/system/ssh-host-cert.timer` with the following contents:

```
# Renew SSH host certificate daily
```

```
#

[Unit]
Description=Renew SSH host certificate daily
Requires=ssh-host-cert.service

[Timer]
OnCalendar= *-*-* 0:0:0
AccuracySec=2h

[Install]
WantedBy=timers.target
```

Enable the timer

Run `systemctl daemon-reload` followed by `systemctl enable ssh-host-cert.timer`.

Configure sshd to use the cert

Most Unix-y systems

Now you've obtained the certificate, and you've set up your system to automatically renew it every day. Now all that's left is to configure the SSH daemon to use the new cert and key you've created. On a “normal” Unix-y system, you'll just need to edit `/etc/ssh/sshd_config`, and add these lines at the bottom:

```
# Path to this host's private key and certificate
HostKey /etc/ssh/ssh_host_ecdsa_key2
HostCertificate /etc/ssh/ssh_host_ecdsa_key2-cert.pub
```

Then restart `sshd`

FreeNAS/TrueNAS systems

Nearly everything in the /etc directory is overwritten at boot, so you'll want to keep the cert and key somewhere else–I suggest `/root/.step/`. And instead of editing `sshd_config`, you can enter the lines above (correcting the path as appropriate) in the GUI, under **Services**, **SSH**, **Auxilliary Parameters**. Finally, in the renewal script, replace `service sshd restart` with `midclt call service.restart "ssh"`.

Nethserver systems

The Nethserver system configuration is generated from templates, so rather than directly editing `sshd_config`, you'll need to create two template fragments. Run `mkdir -p /etc/e-smith/templates-custom/etc/ssh/sshd_config` followed by `nano /etc/e-smith/templates-custom/etc/ssh/sshd_config/20Encryption`. Its contents should be:

```
#
# 20Encryption
#
{
my $Encryption = $sshd{'StrongEncryption'} || 'disabled';

  if ($Encryption eq 'enabled') {
$OUT .=q(
# Require strong Encryption
Ciphers aes128-ctr,aes192-ctr,aes256-ctr,aes128-gcm@openssh.com,aes256-gcm@openssh.com,chacha20-poly1305@openssh.com
HostKeyAlgorithms ssh-dss,ecdsa-sha2-nistp256,ssh-ed25519,rsa-sha2-256,rsa-sha2-512
KexAlgorithms curve25519-sha256,curve25519-sha256@libssh.org,diffie-hellman-group14-sha256,diffie-hellman-group16-sha512,diffie-hel$
MACs hmac-sha2-512-etm@openssh.com,hmac-sha2-256-etm@openssh.com,umac-128-etm@openssh.com
);
  }
}
```

Next, `nano /etc/e-smith/templates-custom/etc/ssh/sshd_config/80Certificates` Its contents should be:

```
# Path to this host's private key and certificate
HostKey /etc/ssh/ssh_host_ecdsa_key2
HostCertificate /etc/ssh/ssh_host_ecdsa_key2-cert.pub
```

Regenerate the config file, and restart sshd, by running `signal-event nethserver-openssh-update`.

In the renewal script above, replace the last line with `/sbin/e-smith/signal-event nethserver-openssh-update`.

Test

Edit your `known_hosts` file to remove the entry for the host whose certificate you've just set up, and then try to SSH to that host. You shouldn't get the common message:

```
The authenticity of host 'ec2-54-161-77-102.compute-1.amazonaws.com
(54.161.77.102)' can't be established.
ECDSA key fingerprint is SHA256:2ae53QcOB0W6HO+XtPmMXk7To/MvMuhFxTj8ZD7eSsE.
Are you sure you want to continue connecting (yes/no)?
```

If you don't, and are directly prompted for the password (or directly logged in, if you have public key authentication enabled), congratulations! It's working.

SSH User Certificates

Using SSH User certificates will likewise require configuration on both the client and the server.

Server configuration

First, you'll need to save a copy of the CA's user signing key on your system. To do that, run `step ssh config --roots > /etc/ssh/ssh_user_key.pub`.

Most Unix-y systems

Next, `nano /etc/ssh/sshd_config`. Add these lines to the end:

```
# Path to the CA public key for verifying user certificates
TrustedUserCAKeys /etc/ssh/ssh_user_key.pub
```

Restart `sshd`.

FreeNAS/TrueNAS systems

Save the User CA key retrieved above in `/root/.step`. And instead of editing `sshd_config`, you can enter the lines above (correcting the path as appropriate) in the GUI, under **Services**, **SSH**, **Auxilliary Parameters**.

Nethserver systems

As noted above, the Nethserver system configuration is generated from templates. If you set up your system for host certificates as described above, you'll only need to edit the `80Certificates` template fragment. If not, you'll first need to create it. First run `mkdir -p /etc/e-smith/templates-custom/etc/ssh/sshd_config`, then `nano /etc/e-smith/templates-`

`custom/etc/ssh/sshd_config/80Certificates`. Add the lines above. Then `signal-event nethserver-openssh-update`.

Client configuration

Before you ssh to a host that requires (or accepts) a certificate, you'll need to log in with `step ssh login <username>`. Step will ask you for a provisioner. For now, as above, choose the `(JWK)` provisioner and enter its password. The CA will issue you a certificate, valid for 16 hours.

The “killer app” feature of Step in this regard is its ability to use OpenID Connect as a provisioner, and therefore use any compatible single sign-on service to authenticate you to the CA. I have it running using LemonLDAP::NG as an authentication provider; this article describes that process.

Test

Make sure you don't have your SSH public key in `~/.ssh/authorized_keys` on the server you just set up above, and then SSH to it–you should be logged in with your certificate.