How to create your own root CA for Solid *

Jens Lechtenbörger

VM Neuland im Internet 2021

This text explains how to create and use a certificate authority (CA) with self-signed root certificate for use on a local Solid server with domain names that end in `solid.localhost`. Certificates are created with OpenSSL, which is free software. The following is known to work under GNU/Linux and with Docker.

Note! The following installs a new root certificate, which should only be done on test systems. With such settings, one can create **valid** certificates for **any** domain.

As explained in Solid's readme a multi-user installation requires the use of wildcard certificates (as a new hostname is created for each user). A wildcard certificate is a certificate for a domain name such as `*.example.org` and is acceptable for all servers under the domain `example.org` (e.g., `www.example.org`, `mail.example.org`).

Initially, I tried to use domain names such as `alice.localhost` and `bob.localhost` with a wildcard certificate for `*.localhost`. This did not work as `localhost` is a top-level domain, for which wildcard certificates are *not* accepted by browsers (without clear error message). Thus, I decided to go for `solid.localhost` as domain name for use with Solid on my machine.

1 Create CA with OpenSSL

The `openssl` program can use default values from a configuration file to reduce typing efforts. Mine are available in this directory, which also contains the files (keys, certificates, ...) created during the following steps. Copy the files `openssl.cnf` and `openssl-wildcard.cnf` to some directory in which you want to create your CA and its certificates.

1. Files for the CA will be located in the new sub-directory `CA`.

   ```
   mkdir CA
   ```

2. Create certificate authority.

   - Create directory structure and necessary files (`"unique_subject = no"` in `index.attr` allows to create multiple certificates per name; the initial serial number is arbitrary).

*This PDF document is an inferior version of an OER HTML page; free/libre Org mode source repository.

```
cd CA; mkdir certs crl newcerts private; touch index.txt; echo "unique_subject =
```

- Invoke `openssl` to create self-signed root certificate. (When asked, use and write down a short pass phrase; remember that all this is only for testing. Hit return to accept default values from configuration file. Note the output options passed to `openssl`.)

  ```
  openssl req -config ../openssl.cnf -new -x509 -days 3650 -newkey rsa:4096 -sha256
  ```

- A real CA would publish its self-signed certificate and have it embedded in operating systems and browsers by default. Do this manually:
  - Make new root certificate known to operating system (afterwards, you can find the new certificate under `/etc/ssl/certs/` and in `/etc/ssl/certs/ca-certificates.crt`)

    ```
    sudo cp certs/my_cacert.crt /usr/local/share/ca-certificates/
    sudo update-ca-certificates
    ```

    * Note that some software uses that certificate store, while other does not. Browsers usually come with their own certificate store (see next step). For node.js you may need to point to the certificate in an environment variable, as is done in entrypoints.sh for a Solid server with Docker.
  - Import new root certificate into browser (e.g., with Firefox: Preferences → Privacy & Security → Certificates → View Certificates → Authorities → Import). You may want to use a separate browser profile for such experiments.

3. Create server key pair and certificate signing request (CSR). Such commands would really be executed by the organization owning the server, not by the CA; the resulting request (containing the public key) is then turned into a signed certificate by the CA in the next step; the CA must never learn the private key.

   Here, the config file `openssl-wildcard.cnf` is used, which contains a section `subjectAltName = @alt_names` with hard-coded names to generate a wildcard certificate for `*.solid.localhost`. When asked, the challenge password can be empty.

   ```
   cd ..
   openssl genrsa -out solid_key.crt 4096
   openssl req -config openssl-wildcard.cnf -sha256 -new -key solid_key.crt -out solid_c
   ```

4. Sign CSR, again with wildcard information. Our new CA does this. Use the pass phrase written down in step 2.

   ```
   openssl ca -config openssl-wildcard.cnf -extensions v3_req -notext -md sha256 -in soli
   ```

5. Create certificate chain. The organization owning the server would do this. The server needs to be told where to find it (and the corresponding private key).

   ```
   cat solid.crt CA/certs/my_cacert.crt > solid.chain.crt
   ```

For Solid with Docker below, copy the CA certificate to the current directory:

```
cp CA/certs/my_cacert.crt .
```

2 Use CA with Solid (manual approach)

Use key and certificate when initializing your Solid server.

Make sure that host names under `solid.localhost` are resolved to your local machine. The IPv4 address of `localhost` is `127.0.0.1`. Add additional lines like this to `/etc/hosts`:

```
127.0.0.1 *.localhost
127.0.0.1 *.solid.localhost
```

Some operating systems seem to ignore such wildcard entries in the `hosts` file, others accept them. Try `ping solid.localhost` with the above settings. If that works, everything is fine. If not, either add entries with full names such as the following ones, or install `dnsmasq` (as in the Docker image mentioned in the next section).

```
127.0.0.1 solid.localhost
127.0.0.1 alice.solid.localhost
127.0.0.1 bob.solid.localhost
127.0.0.1 <more names as necessary>
```

3 Use CA with Solid in Docker

The above setup is bundled in this Docker image for the node Solid server.

Run as follows (maybe replace `$PWD` with a directory of your choice, where you want to collect Solid data):

```
docker run -it --cap-add=NET_ADMIN --dns=127.0.0.1 -p 8443:8443 --name my-solid -v $PWD:/o
```

Finally, explore your own POD: `https://solid.localhost:8443`

4 Clean up

If you imported the above CA certificate into your browser, make sure to delete it at the end of your experiments. View certificates (as for import above), scroll down to "University of Muenster", delete "Solid operator".

License Information