HOW-TOS, LINUX

# BUILDING A ROOT CA AND AN INTERMEDIATE CA USING OPENSSL AND DEBIAN STRETCH

DECEMBER 27, 2017 | D H

## A bit of background

A Root Certificate Authority is used to issue digital certificates to servers, clients or users. It generates digital certificates that certify the ownership of a public key, allowing others to trust the certificate.

An Intermediate Certificate is a subordinate certificate issued by a Root certificate authority for the purpose of issuing certificates. This creates a certificate chain that begins in the Root CA, through the intermediate and ending in the issued certificate. This establishes a chain of trust that can verify the validity of a certificate.

In this post, we will step through the process of creating a Root CA, then an Intermediate CA and finally sign digital certificates for a server. A bit of warning, this setup should be sufficient for a homelab or a small local setup; you should not use this as a production service.

## Prepare to build

Install Debian Stretch, the minimum should suffice. There is no need any GUI. Install SSH for ease of administration and to transfer you certificates securely out.

Make sure that the Fully Qualified Domain Name of the computer is set correctly.

Make sure your time zone is correctly set.

Make sure the time and date are correctly set.

If you wish you can install ntp to ensure time is always correct.

# apt install ntp

Download the following configuration files:

openssl_root.cnf

openssl_intermediate.cnf

openssl_csr_san.cnf

After download rename all of them by dropping the ".txt" extension.

## Creating the Root CA

Create the directory structure for the Root CA:

```
# mkdir /root/ca
# cd /root/ca
# mkdir newcerts certs crl private requests
```

While at /root/ca we should also create "index.txt" file for OpenSSL to keep track of all signed certificates and the "serial" file to give the start point for each signed certificate's serial number. This can be accomplished by doing the following:

```
# cd /root/ca
# touch index.txt
# touch index.txt.attr
# echo '1000' > serial
```

Copy openssl_root.cnf to /root/ca, edit it and look for the following entries:

```
# The root key and root certificate.
private_key = $dir/private/ca.DOMAINNAME.key.pem
certificate = $dir/certs/ca.DOMAINNAME.cert.pem
```

```
# For certificate revocation lists.
crlnumber = $dir/crlnumber
crl = $dir/crl/ca.DOMAINNAME.crl.pem
crl_extensions = crl_ext
default_crl_days = 30
```

Change DOMAINNAME to something that matches the domain of your network, this isn't strictly necessary but it makes for a more customized naming convention.

### Generate the Root private key

(change DOMAINNAME to match what you used in the openssl_root.cnf):

```
# cd /root/ca
# openssl genrsa -aes256 -out private/ca.DOMAINNAME.key.pem 4096
```

### Signing the Root Certificate

Use the root private key to sign the root certificate.
(change DOMAINNAME to match what you used in the openssl_root.cnf):

```
# openssl req -config openssl_root.cnf -new -x509 -sha512 -extensions v3_ca -key /root/ca/private/ca.DOMAINNAME.key.pem -out
/root/ca/certs/ca.DOMAINNAME.crt.pem -days 3650 -set_serial 0
```

Ensure that when filling out the "Common Name" variable that you use the CA server + Domain name of the network

## Creating an Intermediate Certificate Authority

Create a directory to separate the intermediary files from our root configuration

```
# mkdir /root/ca/intermediate
```

Also all the directories and files needed to support (similar to the ones we created for the Root CA):

```
# cd /root/ca/intermediate
# mkdir certs newcerts crl csr private
# touch index.txt
# touch index.txt.attr
# echo 1000 > /root/ca/intermediate/crlnumber
# echo '1234' > serial
```

Copy  openssl_intermediate.cnf to /root/ca/intermediate, edit it and look for the following entries:

```
# The root key and root certificate.
private_key = $dir/private/int.DOMAINNAME.key.pem
certificate = $dir/certs/int.DOMAINNAME.crt.pem

# For certificate revocation lists.
crlnumber = $dir/crlnumber
crl = $dir/crl/int.DOMAINNAME.crl.pem
crl_extensions = crl_ext
default_crl_days = 30
```

Change DOMAINNAME to the same thing that you used in openssl_root.cnf

### Creating the private key and certificate signing request for the Intermediate CA

(change DOMAINNAME to the value you've been using so far)

```
# cd /root/ca
# openssl req -config /root/ca/intermediate/openssl_intermediate.cnf -new -newkey rsa:4096 -keyout
```

```
/root/ca/intermediate/private/int.DOMAINNAME.key.pem -out /root/ca/intermediate/csr/int.DOMAINNAME.csr
```

### Creating the intermediate certificate

(change DOMAINNAME to the value you've been using so far)

```
# openssl ca -config /root/ca/openssl_root.cnf -extensions v3_intermediate_ca -days 3650 -notext -md sha512 -in
/root/ca/intermediate/csr/int.DOMAINNAME.csr -out /root/ca/intermediate/certs/int.DOMAINNAME.crt.pem
```

** Notice that the root CA configurtion (openssl_root.cnf) is used.

## Creating the certificate chain

```
# cd /root/ca
# cat intermediate/certs/int.DOMAINNAME.crt.pem certs/ca.DOMAINNAME.crt.pem > intermediate/certs/chain.DOMAINNAME.crt.pem
```

## What are all these files for?

So now that you have created all these files, which ones are the ones you need?

In /root/ca/certs, ca.DOMAINNAME.crt.pem is the Root CA certificate.
In /root/ca/intermediate/certs, int.DOMAINNAME.crt.pem is the Intermediate CA certificate.
In /root/ca/intermediate/certs, chain.DOMAINNAME.crt.pem is the concatenation of the Root CA certificate and the Intermediate CA certificate.

## Creating server certificates

Copy openssl_csr_san.cnf to /root/ca/intermediate, edit it and change the entries under [alt_names] so that the DNS.* entries match the Fully Qualified Domain Name of the server you wish to create a certificate for. This will create a certificate with embedded Subject Alternative Name (SANs), so no more warnings from Chrome about NET::ERR_CERT_AUTHORITY_INVALID.

### Creating the key and certificate signing request

(change "www.example.com" to your server's FQDN)

```
# cd /root/ca
# openssl req -out intermediate/csr/www.example.com.csr.pem -newkey rsa:2048 -nodes -keyout
intermediate/private/www.example.com.key.pem -config intermediate/openssl_csr_san.cnf
```

### Creating the certificate by signing the signing request with the intermediate CA

(change "www.example.com" to your server's FQDN)

```
# cd /root/ca
# openssl ca -config intermediate/openssl_intermediate.cnf -extensions server_cert -days 3750 -notext -md sha512 -in
intermediate/csr/www.example.com.csr.pem -out intermediate/certs/www.example.com.crt.pem
```

In /root/ca/intermediate/certs you should now have a certificate for use in the server (www.example.com in the case of the example).

## Creating a combined certificate for use with Apache server

(change "www.example.com" to your server's FQDN)

```
# openssl pkcs12 -inkey www.example.com.key.pem -in www.example.com.crt.pem -export -out www.example.com.combined.pfx
# openssl pkcs12 -in www.example.com.combined.pfx -nodes -out www.example.com.combined.crt
```