Own SSL CA Authority for Local HTTPS

When you generate a self-signed certificate the browser doesn't trust it. It hasn't been signed by a CA. The way to get around this is to generate our own root certificate and private key. We then add the root certificate to all the devices we own just once, and then all the self-signed certificates we generate will be inherently trusted.

CA Key and Certificate

Step 1: Create private key for local CA Certificate

To generate the private key to become a local CA execute:

```
openssl genrsa -des3 -out Home-CA.key 2048
```

OpenSSL will ask for a passphrase, which we recommend not skipping and keeping safe. The passphrase will prevent anyone who gets your private key from generating a root certificate of their own. The output should look like this:

```
$ openssl genrsa -des3 -out Home-CA.key 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
..........................+++++
......................+++++
e is 65537 (0x010001)
Enter pass phrase for Home-CA.key:
Verifying - Enter pass phrase for Home-CA.key:
```

The following key file is generated:

```
$ ls -al
total 4
drwxr-xr-x  2 oscar oscar   60 Apr  1 21:55 .
drwxrwxrwt 16 root  root   380 Apr  1 21:49 ..
-rw-------  1 oscar oscar 1743 Apr  1 21:52 Home-CA.key
```

Step 2: Generate a root CA certificate

Next, we generate a root certificate:

```
openssl req -x509 -new -nodes -key Home-CA.key -sha256 -days 15000 -out
Home-CA.pem
```

You will be prompted for the passphrase of the private key you just chose and a bunch of questions. The answers to those questions aren't that important. They show up when looking at the certificate, which you will almost never do. I suggest making the Common Name something that you'll recognize as your root certificate in a list of other certificates. That's really the only thing that matters.

```
Enter pass phrase for Home-CA.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:NL
State or Province Name (full name) [Some-State]:Zuid-Holland
Locality Name (eg, city) []:Rijnsburg
Organization Name (eg, company) [Internet Widgits Pty Ltd]:oscardegroot.nl
Organizational Unit Name (eg, section) []:oscardegroot.nl
Common Name (e.g. server FQDN or YOUR name) []:Oscar de Groot
Email Address []:oscar@oscardegroot.nl
```

When you should see the following two files: Home-CA.key (your private key) and Home-CA.pem (your root certificate), you're now a CA.

Installing Your Root Certificate

To become a CA for the devices we own, we need to add the root certificate to any laptops, desktops, tablets, and phones that access your HTTPS sites. This can be a bit of a pain, but the good news is that we only have to do it once. Our root certificate will be good until it expires.

Adding the Root Certificate to Linux

There are so many Linux distributions, but Ubuntu/Debian is by far the most popular. Therefore these instructions will cover Ubuntu. If it isn't already installed, install the **ca-certificates package**.

```
sudo apt-get install -y ca-certificates
```

Copy the Home-CA.pem file to the **/usr/local/share/ca-certificates** directory as a Home-CA.crt file.

```
sudo cp ~/certs/myCA.pem /usr/local/share/ca-certificates/myCA.crt
```

Update the certificate store.

```
sudo update-ca-certificates
```

You can test that the certificate has been installed by running the following command:

```
awk -v cmd='openssl x509 -noout -subject' '/BEGIN/{close(cmd)};{print |
cmd}' < /etc/ssl/certs/ca-certificates.crt | grep Hellfish
```

If it's installed correctly, you'll see the details of the root certificate.

```
subject=C = US, ST = Springfield State, L = Springfield, O = Hellfish Media,
OU = 7G, CN = Hellfish Media, emailAddress = abraham@hellfish.media#
```

Creating CA-Signed Certificates for Your Dev Sites

Now we're a CA on all our devices and we can sign certificates for any new dev sites that need HTTPS.

Step 1: Create a Private Key

First, we create a private key for the dev site. Note that we name the private key using the domain name URL of the dev site. This is not required, but it makes it easier to manage if you have multiple sites.

```
openssl genrsa -out internal.server.key 2048
```

Step 2: Generate the CSR (certificate signing request)

Then we create a CSR:

```
openssl req -new -key internal.server.key -extensions v3_ca -out
internal.server.csr
```

```
You'll get all the same questions as you did above and, again, your answers
don't matter. In fact, they matter even less because you won't be looking at
this certificate in a list next to others.
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:NL
State or Province Name (full name) [Some-State]:Zuid-Holland
Locality Name (eg, city) []:Rijnsburg
Organization Name (eg, company) [Internet Widgits Pty Ltd]:oscardegroot.nl
Organizational Unit Name (eg, section) []:oscardegroot.nl
Common Name (e.g. server FQDN or YOUR name) []:oscardegroot.nl
Email Address []:oscar@oscardegroot.nl

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:xxxxxx
An optional company name []:
```

Step 3: Create extensions file to specify subjectAltName

Finally, we'll create an X509 V3 certificate extension config file, which is used to define the Subject Alternative Name (SAN) for the certificate. In our case, we'll create a configuration file called internal.server.ext containing the following text:

```
basicConstraints=CA:FALSE
subjectAltName=DNS:*.home.lan
extendedKeyUsage=serverAuth
```

Step 4: Generate the Certificate using the CSR

We'll be running openssl x509 because the x509 command allows us to edit certificate trust settings. In this case we're using it to sign the certificate in conjunction with the config file, which allows us to set the Subject Alternative Name. I originally found this answer on Stack Overflow.

Now we run the command to create the certificate: using our CSR, the CA private key, the CA certificate, and the config file:

```
openssl x509 -req -in internal.server.csr -CA Home-CA.pem -CAkey Home-CA.key
-CAcreateserial -out internal.server.crt -days 15000 -sha256 -extfile
internal.server.ext
```

We now have three files: internal.server.key (the private key), internal.server.csr (the certificate signing request, or csr file), and internal.server.crt (the signed certificate). We can configure local web servers to use HTTPS with the private key and the signed certificate.

From:
https://wiki.oscardegroot.nl/ - **HomeWiki**

Permanent link:
https://wiki.oscardegroot.nl/doku.php?id=networking:ssl-own-ca&rev=1680452178

Last update: **2023/04/02 16:16**