Membuat sertifikat SSL untuk lingkungan pengembangan aplikasi berbasis web

Rizky Januar Akbar

Referensi:
https://www.davidpashley.com/articles/becoming-a-x-509-certificate-authority/
https://github.com/midasplatform/infrastructure/wiki/Become-your-own-SSL-Certificate-Authority
https://www.namecheap.com/support/knowledgebase/article.aspx/9821/38/apache-redirect-to-https
https://www.thesslstore.com/knowledgebase/ssl-generate/csr-generation-guide-for-nginx-openssl/
https://blog.zencoffee.org/2013/04/creating-and-signing-an-ssl-cert-with-alternative-names/
https://www.sslshopper.com/article-most-common-openssl-commands.html

Latar belakang

Pada lingkungan development pada platform web ada beberapa Javascript API yang membutuhkan koneksi SSL atau menggunakan localhost. Dalam kasus kita menggunakan domain sendiri misalnya dev.local, example.local, kita membutuhkan sertifikat SSL yang digunakan untuk domain-domain tersebut. Solusi yang bisa dilakukan adalah membuat sertifikat SSL dari penyedia SSL berbayar seperti VeriSign, DigiCert dan lainnya. Akan tetapi solusi itu berbayar dan tentunya kita membutuhkan banyak sertifikat. Sehingga tidak ekonomis untuk dilakukan. Solusi lainnya adalah dengan membuat sertifikat self-signed untuk setiap domain atau project yang digunakan. Akan tetapi solusi tersebut mengharuskan kita melakukan bypass warning di browser. Solusi lain adalah dengan membuat Root CA sendiri dan melakukan signing untuk setiap self-signed SSL untuk setiap domain. Kita hanya perlu menambahkan satu Root CA saja ke browser dan self-signed SSL lainnya otomatis terpercaya.

Membuat self-signed Root Certificate Authority (Root CA)

1. Buat direktori git dengan nama local-certs.

```
mkdir local-certs
cd local-certs
git init
```

2. Di dalam folder local-certs buat direktori dengan nama root-ca dan di dalam direktori root-ca buat direktori dengan nama conf, private, dan public.

```
cd local-certs
mkdir -p root-ca/{conf,private,public}
cd root-ca
```

3. Buat file `conf/openssl.cnf`. Ganti nama developer/perusahaan dengan nama Anda atau nama perusahaan Anda.

```
[ req ]
default_bits            = 4096
default_keyfile         = ./private/root.pem
default_md              = sha256
prompt                  = no
distinguished_name      = root_ca_distinguished_name
x509_extensions         = v3_ca

[ root_ca_distinguished_name ]
countryName             = ID
stateOrProvinceName     = Jawa Timur
localityName            = Surabaya
0.organizationName      = Nama Developer/Perusahaan
commonName              = Nama Developer/Perusahaan Root CA
emailAddress            = email@example.com

[ v3_ca ]
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer:always
basicConstraints = CA:true
```

4. Buat private dan public key

```
openssl req -nodes -config conf/openssl.cnf -days 1825
   -x509 -newkey rsa:4096 -out public/root.pem -outform PEM
```

5. Membuat public key dalam format DER (jika diperlukan).

```
openssl x509 -in public/root.pem -outform DER -out public/root.der
```

6. Menambahkan konfigurasi di `conf/openssl.cnf` untuk melakukan signing.

```
[ ca ]
default_ca = CA_default
[ CA_default ]
dir                 =.
new_certs_dir       = ./signed-keys/
database            = ./conf/index
certificate         = ./public/root.pem
serial              = ./conf/serial
private_key         = ./private/root.pem
x509_extensions     = usr_cert
name_opt            = ca_default
cert_opt            = ca_default
default_crl_days    = 30
default_days        = 365
default_md          = sha256
preserve            = no
policy              = policy_match
[ policy_match ]
countryName             = match
stateOrProvinceName     = optional
```

```
organizationName        = optional
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional
[ usr_cert ]
basicConstraints=CA:FALSE
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer:always
nsCaRevocationUrl     = https://ca.local/ca-crl.pem
```

7. Buat direktori untuk menyimpan daftar key yang telah di-signed

```
mkdir signed-keys
echo "01" > conf/serial
touch conf/index
```

Menambahkan Root CA ke OS atau browser

<tergantung OS dan browser>

Untuk browser Firefox cukup diimpor di browser
Untuk browser Chrome diimpor di OS

Membuat self-signed SSL certificate

Misalkan kita akan membuat SSL certificate untuk domain **dev.local**

1. Buat direktori dev.local di dalam folder local-certs

   ```
   cd local-certs
   mkdir dev.local
   ```

2. Buat file `req.conf` dan masukkan detil CSR

   ```
   [req]
   distinguished_name = req_distinguished_name
   req_extensions = v3_req
   prompt = no

   [req_distinguished_name]
   C = ID
   ST = Jawa Timur
   L = Surabaya
   O = Nama developer/perusahaan
   OU = Nama developer/perusahaan
   CN = dev.local


   [v3_req]
   basicConstraints = CA:FALSE
   ```

```
keyUsage = nonRepudiation, digitalSignature,
keyEncipherment
extendedKeyUsage = serverAuth
subjectAltName = @alt_names
[alt_names]
DNS.1 = dev.local
DNS.2 = www.dev.local
```

Keterangan:

a. Common Name: FQDN (fully-qualified domain name) misalnya dev.local atau example.org
b. Organization: Nama lengkap perusahaan
c. Organization Unit (OU): Nama departemen atau divisi misal IT Division.
d. City or Locality: Nama kota tempat perusahaan.
e. State or province: nama provinsi
f. Country: nama negara menggunakan 2-karakter kode negara.

3. Buat private key dan CSR.

```
openssl req -new -out dev.local.csr -newkey rsa:2048 -nodes -sha256 -keyout
dev.local.key -config req.conf
```

4. Memeriksa CSR sesuai dengan yang request.

```
openssl req -text -noout -verify -in dev.local.csr
```

Pastikan Entri Subject Alternative Names (SAN) sudah masuk.

```
X509v3 Subject Alternative Name:
                DNS:dev.local, DNS:www.dev.local
```

Melakukan signing menggunakan Root CA

1. Pindah ke root-ca

```
cd local-certs/root-ca
```

2. Edit conf/`openssl.cnf` untuk menambahkan SAN. Langkah ini harus dilakukan untuk setiap signing dengan domain/CN yang berbeda.

```
[ usr_cert ]
basicConstraints=CA:FALSE
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer:always
nsCaRevocationUrl     = https://ca.local/ca-crl.pem
subjectAltName = @alt_names
[alt_names]
DNS.1 = dev.local
```

```
DNS.2 = www.dev.local
```

3. Lakukan signing

```
openssl ca -batch -config conf/openssl.cnf -in
../dev.local/dev.local.csr -out ../dev.local/dev.local.crt
```

4. Periksa SSL certificate hasil signing

```
openssl x509 -in dev.local.crt -text -noout
```

Pastikan Entri SAN sudah masuk.

```
X509v3 Subject Alternative Name:
                DNS:dev.local, DNS:www.dev.local
```

Memasang self-CA-signed SSL ke aplikasi web (Apache)

```
<VirtualHost *:80>
        DocumentRoot "/Users/rizky/Sites"
        ServerName dev.local
        ErrorLog "/var/log/apache2/dev.local-error_log"
        CustomLog "/var/log/apache2/dev.local-access_log" common

        <Directory "/Users/rizky/Sites">
                AllowOverride All
                Require all granted
        </Directory>

        Redirect permanent / https://dev.local/
</VirtualHost>

<VirtualHost *:443>
    DocumentRoot "/Users/rizky/Sites"
    ServerName dev.local
    SSLEngine on
    SSLCertificateFile "/Users/rizky/git/local-ca/dev.local/dev.local.crt"
    SSLCertificateKeyFile "/Users/rizky/git/local-
ca/dev.local/dev.local.key"
</VirtualHost>
```