Using OpenSSL to generate and format certificates

Last Updated: 2024-12-16

Use OpenSSL to create your own self-signed certificates, or convert PEM certificate files to P12 files.

OpenSSL is an open source toolkit that can be used for generating and validating TLS certificates.

This topic covers how to generate a self-signed TLS certificate by using the OpenSSL toolkit, and how to convert certificates in PEM format to P12 format.

Creating a self-signed certificate with OpenSSL

Use the following command to create a private key and public certificate with OpenSSL:

```
openssl req -newkey rsa:2048 -nodes -keyout private_key.pem -x509 -days 365 -out public_certificate.pem
```

The OpenSSL command asks for the following information to create your private key and public certificate (example values shown):

- Country Name (2 letter code) []: US
- State or Province Name (full name) []: California
- Locality Name (eg, city) []: Sacramento
- Organization Name (eg, company) []: exampleorganization
- Organizational Unit Name (eg, section) []:
- Common Name (eg, fully qualified host name) []: gateway.exampleorganization.com
- Email Address []:

Common Name must be set to the fully qualified domain name of the system that uses the certificate. For static DNS, use the hostname or IP address set in your Gateway Cluster (for example. `192.16.183.131` or `gateway.exampleorganization.com`).

To review the generated public certificate run:

```
openssl x509 -text -noout -in public_certificate.pem
```

To combine the generated private_key and public_certificate PEM files into a password protected P12 file, run:

```
openssl pkcs12 -inkey private_key.pem -in public_certificate.pem -export -out certificate.p12
```

To validate a P12 file, run:

```
openssl pkcs12 -in certificate.p12 -noout -info
```

Creating PKCS#12 (P12) certificate from PEM files

PKCS#12 (P12) defines an archive file format for storing cryptographic objects as a single file. Use OpenSSL to create a password protected P12 file from your PEM files.

Prerequisites:

- Private key file. For example: `private_key.pem`.
- CA signed certificate. For example: `public_certificate.pem`.
- If you have intermediate certificates from your CA, concatenate them into a single `.pem` file to create a `ca_chain.pem` file:

```
cat ca1.pem ca2.pem ca3.pem > ca_chain.pem
```

the `ca_chain.pem` file should look like this:

```
-----BEGIN CERTIFICATE-----
MIIEpjCCA46gAwIBAgIQEOd26KZabjd+BQMG1Dwl6jANBgkqhkiG9w0BAQUFADCB
...
lQX7CkTJn6lAJUsyEa8H/gjVQnHp4VOLFR/dKgeVcCRvZF7Tt5AuiyHY
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIEPDCCAySgAwIBAgIQSEus8arH1xND0aJ0NUmXJTANBgkqhkiG9w0BAQUFADBv
...
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIENjCCAx6gAwIBAgIBATANBgkqhkiG9w0BAQUFADBvMQswCQYDVQQGEwJTRTEU
```

```
...
-----END CERTIFICATE-----
```

– If you have multiple end-entity (leaf) certificates in one file, then split them into separate files and create separate P12 files for each certificate.

Create the P12 file with the following OpenSSL command (if you don't have CA certificates to include, then omit `-CAfile <filename> -chain`):

```
openssl pkcs12 -inkey private_key.pem -in public_certificate.pem -export -out certificate.p12 -CAfile caChain.pem -chain
```

Parent topic:

→ Creating a TLS client profile

Related tasks

→ Creating a TLS client profile
→ Creating a keystore
→ Creating a truststore