Sign server and client certificates

We will be signing certificates using our intermediate CA. You can use these signed certificates in a variety of situations, such as to secure connections to a web server or to authenticate clients connecting to a service.

> **Note**
>
> The steps below are from your perspective as the certificate authority. A third-party, however, can instead create their own private key and certificate signing request (CSR) without revealing their private key to you. They give you their CSR, and you give back a signed certificate. In that scenario, skip the `genrsa` and `req` commands.

Create a key

Our root and intermediate pairs are 4096 bits. Server and client certificates normally expire after one year, so we can safely use 2048 bits instead.

> **Note**
>
> Although 4096 bits is slightly more secure than 2048 bits, it slows down TLS handshakes and significantly increases processor load during handshakes. For this reason, most websites use 2048-bit pairs.

If you're creating a cryptographic pair for use with a web server (eg, Apache), you'll need to enter this password every time you restart the web server. You may want to omit the `-aes256` option to create a key without a password.

```
# cd /root/ca
# openssl genrsa -aes256 \
      -out intermediate/private/www.example.com.key.pem 2048
# chmod 400 intermediate/private/www.example.com.key.pem
```

Create a certificate

Use the private key to create a certificate signing request (CSR). The CSR details don’t need to match the intermediate CA. For server certificates, the **Common Name** must be a fully qualified domain name (eg, `www.example.com`), whereas for client certificates it can be any unique identifier (eg, an e-mail address). Note that the **Common Name** cannot be the same as either your root or intermediate certificate.

```
# cd /root/ca
# openssl req -config intermediate/openssl.cnf \
      -key intermediate/private/www.example.com.key.pem \
      -new -sha256 -out intermediate/csr/www.example.com.csr.pem

Enter pass phrase for www.example.com.key.pem: secretpassword
You are about to be asked to enter information that will be incorporat
into your certificate request.
-----
Country Name (2 letter code) [XX]:US
State or Province Name []:California
Locality Name []:Mountain View
Organization Name []:Alice Ltd
Organizational Unit Name []:Alice Ltd Web Services
Common Name []:www.example.com
Email Address []:
```

To create a certificate, use the intermediate CA to sign the CSR. If the certificate is going to be used on a server, use the `server_cert` extension. If the certificate is going to be used for user authentication, use the `usr_cert` extension. Certificates are usually given a validity of one year, though a CA will typically give a few days extra for convenience.

```
# cd /root/ca
# openssl ca -config intermediate/openssl.cnf \
      -extensions server_cert -days 375 -notext -md sha256 \
      -in intermediate/csr/www.example.com.csr.pem \
      -out intermediate/certs/www.example.com.cert.pem
# chmod 444 intermediate/certs/www.example.com.cert.pem
```

The `intermediate/index.txt` file should contain a line referring to this new certificate.

```
V 160420124233Z 1000 unknown ... /CN=www.example.com
```

Verify the certificate

```
# openssl x509 -noout -text \
      -in intermediate/certs/www.example.com.cert.pem
```

The **Issuer** is the intermediate CA. The **Subject** refers to the certificate itself.

```
Signature Algorithm: sha256WithRSAEncryption
    Issuer: C=GB, ST=England,
            O=Alice Ltd, OU=Alice Ltd Certificate Authority,
            CN=Alice Ltd Intermediate CA
    Validity
        Not Before: Apr 11 12:42:33 2015 GMT
        Not After : Apr 20 12:42:33 2016 GMT
    Subject: C=US, ST=California, L=Mountain View,
             O=Alice Ltd, OU=Alice Ltd Web Services,
             CN=www.example.com
    Subject Public Key Info:
        Public Key Algorithm: rsaEncryption
            Public-Key: (2048 bit)
```

The output will also show the **X509v3 extensions**. When creating the certificate, you used either the `server_cert` or `usr_cert` extension. The options from the corresponding configuration section will be reflected in the output.

```
X509v3 extensions:
    X509v3 Basic Constraints:
        CA:FALSE
    Netscape Cert Type:
        SSL Server
    Netscape Comment:
        OpenSSL Generated Server Certificate
    X509v3 Subject Key Identifier:
        B1:B8:88:48:64:B7:45:52:21:CC:35:37:9E:24:50:EE:AD:58:02:B5
    X509v3 Authority Key Identifier:
        keyid:69:E8:EC:54:7F:25:23:60:E5:B6:E7:72:61:F1:D4:B9:21:D4:45
        DirName:/C=GB/ST=England/O=Alice Ltd/OU=Alice Ltd Certificate
        serial:10:00

    X509v3 Key Usage: critical
        Digital Signature, Key Encipherment
    X509v3 Extended Key Usage:
        TLS Web Server Authentication
```

Use the CA certificate chain file we created earlier (`ca-chain.cert.pem`) to verify that the new certificate has a valid chain of trust.

```
# openssl verify -CAfile intermediate/certs/ca-chain.cert.pem \
      intermediate/certs/www.example.com.cert.pem

www.example.com.cert.pem: OK
```

Deploy the certificate

You can now either deploy your new certificate to a server, or distribute the certificate to a client. When deploying to a server application (eg, Apache), you need to make the following files available:

- `ca-chain.cert.pem`
- `www.example.com.key.pem`
- `www.example.com.cert.pem`

If you’re signing a CSR from a third-party, you don’t have access to their private key so you only need to give them back the chain file (`ca-chain.cert.pem`) and the certificate (`www.example.com.cert.pem`).

Comments

← Previous

Next →

Version 1.0.4 — Last updated on 2015-12-09.