Quick Cookbook on using Certificates for Testing with TLS

Generating TLS Certificate Files for Use in Lab Environment

TLS uses X.509 certificates during the handshake to verify identity and establish secure communication. There are two levels of authentication that are typical.

1) One way authentication: Each client authenticates the server's certificate during the handshake. The server does
not authenticate the client's identity. This is the most common method in web applications.

2) Mutual authentication: Each client still authenticates the server. In addition, the server authenticates every client's certificate.

For this cookbook, there are three components.

1) The CA (Certificate Authority)
2) The Server
3) The Client

The CA needs 2 files:
ca.key : the CA's private key
ca.crt : the CA's certificate, also known as the root certificate

For one way authentication, the server needs two files:
server.crt : the server's certificate
server.key : the server's private key

For mutual authentication, the server needs three files:
server.crt : the server's certificate
server.key : the server's private key
ca.crt : the root certificate

For one way authentication, the client needs one file
ca.crt : the root certificate

For mutual authentication, the client needs three files
client.crt: the client's certificate
client.key the client's private key
ca.crt : the root certificate

To generate the CA files:
openssl genrsa -out ca.key 4096
openssl req -new -x509 -days 1000 -key ca.key -out ca.crt
(The second command will prompt for input; none of the answers are important in this step.)

To generate the server files:
openssl genrsa -out server.key -out server.csr
openssl x509 -req -days 1000 -in server.csr -CA ca.crt -CAkey ca.key -set_serial 01 -out server.crt
(The second command will prompt for input. When prompted for Common Name, this must be the DNS name or IP address of the server.)

To generate the client files.
openssl genrsa -out server.key -out client.csr
openssl x509 -req -days 1000 -in client.csr -CA ca.crt -CAkey ca.key -set_serial 01 -out client.crt
(The second command will prompt for input. When prompted for Common Name, this must be the DNS name or IP address of the client.)

Next, copy these files to appropriate devices and configure applications with references to these files.

Some examples:

To configure kube-apiserver for one-way authentication:

kube-apiserver ... --tls_cert_file=server.crt --tls_private_key_file=server.key

To configure kubecctl for one-way authentication

kubectl ... --certificate-authority=ca.crt

To configure kube-apiserver for mutual authentication:

kube-apiserver ... --tls_cert_file=server.crt --tls_private_key_file=server.key --certificate-authority=ca.crt

To configure kubecctl for mutual authentication:

kubectl ... --client-certificate=client.crt --client-key=client.key --certificate-authority=ca.crt