

Appendix G

PostgreSQL SSL Configuration

This appendix documents the process of configuring PostgreSQL to use SSL for secure connections. This includes placing an intermediate CA in the chain of trust. While this is not strictly necessary, it allows users to keep their root CA safe (i.e., offline) once the intermediate certificates have been created. In the case of a security breach, only the intermediate certificate needs to be revoked.

This appendix details how to create a self-signed root CA for the purposes of demonstration, but a user's own root CA can be substituted.

Configuring `openssl.cnf`

These instructions assume the `openssl` configuration file is located at `/etc/ssl/openssl.cnf`. From a default configuration, ensure the following line in the `[v3_ca]` section has been uncommented:

```
keyUsage = cRLSign, keyCertSign
```

Create a self-signed CA (optional)

There will likely be a certificate signed by a trusted CA, but for some installations, a user may want to create a self-signed certificate.

```
Create a private key:  
openssl genrsa -aes256 -out ca.key 4096
```

Enter a passphrase. This should be long and well-guarded.

```
Create a self-signed certificate:  
openssl req -new -x509 -sha256 -days 1825 -key ca.key -out ca.crt \  
-subj "/C=US/ST=VA/L=Arlington/O=Crunchy Data Solutions/CN=root-ca"
```

Now, create the intermediate CAs that will be used to sign server and client certificates.

```
Create the server intermediate private key;  
openssl genrsa -aes256 -out server-intermediate.key 4096
```

Enter a passphrase. Do not reuse the passphrase from the root key.

```
Create the server intermediate certificate signing request (CSR):
openssl req -new -sha256 -days 1825 -key server-intermediate.key \
  -out server-intermediate.csr \
  -subj "/C=US/ST=VA/L=Arlington/O=Crunchy Data Solutions/CN=server-im-ca"
```

```
Create the server intermediate certificate by signing with the CA certificate:
openssl x509 -extfile /etc/ssl/openssl.cnf -extensions v3_ca -req \
  -days 1825 -CA ca.crt -CAkey ca.key -CAcreateserial \
  -in server-intermediate.csr -out server-intermediate.crt
```

Now, repeat the process to create the client intermediate CA.

```
openssl genrsa -aes256 -out client-intermediate.key 4096
openssl req -new -sha256 -days 1825 -key client-intermediate.key \
  -out client-intermediate.csr -subj "/C=US/ST=VA/L=Arlington/O=Crunchy Data
Solutions/CN=client-im-ca"
openssl x509 -extfile /etc/ssl/openssl.cnf -extensions v3_ca -req \
  -days 1825 -CA ca.crt -CAkey ca.key -CAcreateserial
  -in client-intermediate.csr -out client-intermediate.crt
```

Create Server/Client Certificate

Server and client certificates are signed by their respective intermediate CAs rather than the root CA. Additionally, the common name on server certificates must match the hostname of the server and the common name of the client certificates must match the client's PostgreSQL user logon (or be mapped in `pg_ident.conf`). The private keys will be created without passphrases to allow automatic startup of the PostgreSQL server and client.

```
Create a server certificate:
openssl req -nodes -new -newkey rsa:4096 -sha256 -keyout server.key \
  -out server.csr -subj "/C=US/ST=VA/L=Arlington/O=Crunchy Data
Solutions/CN=server.crunchydata.com"
openssl x509 -extfile /etc/ssl/openssl.cnf -extensions usr_cert -req \
  -days 1825 -CA server-intermediate.crt -CAkey server-intermediate.key \
  -CAcreateserial -in server.csr -out server.crt
```

Create a client certificate:

```
openssl req -nodes -new -newkey rsa:4096 -sha256 -keyout client.key \  
  -out client.csr -subj "/C=US/ST=VA/L=Arlington/O=Crunchy Data  
Solutions/CN=pgusername"  
openssl x509 -extfile /etc/ssl/openssl.cnf -extensions usr_cert -req \  
  -days 1825 -CA client-intermediate.crt -CAkey client-intermediate.key \  
  -CAcreateserial -in client.csr -out client.crt
```

Configuring PostgreSQL Server Configuration

The examples below will use `${PGDATA?}` from APPENDIX-F as the `data_directory` setting in `postgresql.conf`.

Copy the root CA

```
cp ca.crt ${PGDATA?}/ca.crt
```

Copy the server key

```
cp server.key ${PGDATA?}/server.key
```

Copy the server, server-intermediate, and root ca certificates to PostgreSQL's `server.crt`.

The exact order specified here is required:

```
cat server.crt server-intermediate.crt ca.crt > ${PGDATA?}/server.crt
```

Set permissions (this is required for server start).

```
chown postgres:postgres ${PGDATA?}/ca.crt \  
  ${PGDATA?}/server.crt ${PGDATA?}/server.key  
chmod 600 ${PGDATA?}/ca.crt ${PGDATA?}/server.crt \  
  ${PGDATA?}/server.key
```

Now, configure `${PGDATA?}/postgresql.conf` with the SSL settings.

```
ssl = true  
ssl_cert_file = 'server.crt'  
ssl_key_file = 'server.key'  
ssl_ca_file = 'ca.crt'
```

Ensure `${PGDATA?}/pg_hba.conf` requires certs for the clients if they are not optional:

```
hostssl all all network/mask cert
```

Restart the server for settings to take effect.

Client Configuration

The examples below assume the user is logged on to the OS as the user being configured.

Copy the root CA.

```
cp ca.crt ~/.postgresql/root.crt
```

Copy the client key.

```
cp client.key ~/.postgresql/postgresql.key
```

The client, client-intermediate, and root ca certificates must be copied to the client's `postgresql.crt`. The exact order specified here is required.

```
cat client.crt client-intermediate.crt ca.crt > ~/.postgresql/postgresql.crt
```

Set permissions (this is required for client operation).

```
chmod 600 \  
~/.postgresql/root.crt \  
~/.postgresql/postgresql.key \  
~/.postgresql/postgresql.crt
```

Note: These files can also be configured with environment variables. Refer to <https://www.postgresql.org/docs/current/libpq-ssl.html> for more information.

Running the Client

When running client software it is best to use the verify-full SSL mode. Refer to the link in Client Configuration for a description of what the SSL modes mean and what level of protection they provide.

An example using `psql`:

```
psql "postgresql://server.crunchydata.com/postgres?sslmode=verify-full"
```