SSL Certificates For PostgreSQL

>

On this page

SSL Certificates For PostgreSQL

SSL Certificates For PostgreSQL

This describes how to set up ssl certificates to enable encrypted connections from PgAdmin on some client machine to postgresql on a server machine. The assumption is that postgresql (compiled with ssl support) and openssl are already installed and functional on the server (Linux). PgAdmin is already installed on the client (either Windows or Linux).

ON SERVER

--------=

On the server, three certificates are required in the data directory. CentOS default is /dbdata/pgsql/data/:

root.crt (trusted root certificate)

server.crt (server certificate)

server.key (private key)

Issue commands as root.

------------------------------

sudo -

cd /dbdata/pgsql/data

Generate a private key (you must provide a passphrase).

------------------------------

openssl genrsa -des3 -out server.key 2048

Remove the passphrase.

------------------------------

openssl rsa -in server.key -out server.key

Create the server certificate.

------------------------------

--subj is a shortcut to avoid prompting for the info.

--x509 produces a self signed certificate rather than a certificate request.

openssl req -new -key server.key -days 3650 -out server.crt -x509 -subj '/C=CA/ST=British Columbia/L=Comox/O=TheBrain.ca/CN=thebrain.ca/emailAddress=info@thebrain.ca'

--Since we are self-signing, we use the server certificate as the trusted root certificate.

cp server.crt root.crt

Set appropriate permission and owner on the private key file.

------------------------------

ls -lhrt

root.crt

server.crt

server.key

chmod 400 server.key

chmod 400 root.crt

chmod 400 server.crt

chown postgres.postgres server.key

chown postgres.postgres root.crt

chown postgres.postgres server.crt

You'll need to edit pg\_hba.conf. For example:

/var/lib/pgsql/14/data/pg\_hba.conf

# TYPE DATABASE USER CIDR-ADDRESS METHOD

# "local" is for Unix domain socket connections only

local all all trust

# IPv4 local connections:

host all all 127.0.0.1/32 trust

--FOR Old Postgesql version

hostssl all www-data 0.0.0.0/0 md5 clientcert=1

hostssl all postgres 0.0.0.0/0 md5 clientcert=1

--FOR New Postgesql versions

hostssl all all 0.0.0.0/0 scram-sha-256 clientcert=verify-ca

hostssl replication replication 174.1.0.7/32 scram-sha-256

You need to edit postgresql.conf to actually activate ssl

/var/lib/pgsql/14/data/postgresql.conf

ssl = ON

ssl\_ca\_file = 'root.crt'

service postgresql-14 start

If the server fails to (re)start, look in the postgresql startup log, /var/lib/pgsql/pgstartup.log default for CentOS, for the reason.

For Windows, these files must be in %appdata%\postgresql\ directory. For Linux ~/.postgresql/ directory.

CREATE new dir

root.crt (trusted root certificate)

postgresql.crt (client certificate)

postgresql.key (private key)

Generate the the needed files on the server machine, and then copy them to the client.

We'll generate the needed files in the /tmp/ directory.

First create the private key postgresql.key for the client machine, and remove the passphrase.

openssl genrsa -des3 -out /home/m2p331/postgresql.key 2048

openssl rsa -in /home/m2p331/postgresql.key -out /home/m2p331/postgresql.key

Then create the certificate postgresql.crt.

It must be signed by our trusted root (which is using the private key file on the server machine).

Also, the certificate common name (CN) must be set to the database user name we'll connect as.

openssl req -new -key /home/m2p331/postgresql.key -out /home/m2p331/postgresql.csr -subj '/C=CA/ST=British Columbia/L=Comox/O=TheBrain.ca/CN=www-data'

openssl x509 -req -in /home/m2p331/postgresql.csr -CA root.crt -CAkey server.key -out /home/m2p331/postgresql.crt -CAcreateserial

ON CLIENT

----------

Copy the three files we created from the server /tmp/ directory to the client machine.

cd /tmp/ind2\_client

chmod 400 postgresql.crt

chmod 400 postgresql.csr

chmod 400 postgresql.key

chown postgres.postgres postgresql.crt

chown postgres.postgres postgresql.key

chown postgres.postgres postgresql.csr

=================================

Comments

By: Gurjeet SinghReply

Using /tmp directory to generate certificates is simply asking for trouble. I am not sure what permissions OpenSSL uses when creating files, but anything in /tmp is world-readable by default, hence insecure since anybody can copy it from there before you get a chance to delete the files.

Regards,

By: Jason SmithReply

I think the same process can be used if you are using other certificates. I use a Quick SSL and used the above process and it was successful. What do you think, will i face any problems later?

By: Vince HerriedReply

in the last steps u say copy the the three files generated on

/tmp on the server to the client.

Where on the client?

u say copy root.crt into pgadmin ~/.posgresql/

I have no directory by that name. Huh... user pgadmin?

or user 'vince' ( me )

I'm running Fedora F14 postgresql-8.4.7-1

By:Reply

As user "vince". That is, as the user (on the client machine) you normally work as.

You need to create the directory on the client machine:mkdir ~/.postgresql

After copying the three files from the server (/tmp/{postgresql.key,postgresql.crt,root.crt}) to the client machine (into directory ~/.postgresql/), you'll need to set the permission of the key to not world readable: chmod 0400 ~/.postgresql/postgresql.key

On windows, permissions in the are handled automatically for you.

By: Jean-Yves F. BarbierReply

Hi,

I followed this HOWTO but found the last command is wrong; ORG code is:

openssl x509 -req -in /tmp/postgresql.csr -CA root.crt -CAkey server.key -out /tmp/postgresql.crt -CAcreateserial

but should be:

openssl x509 -req -in server.req -out /tmp/postgresql.csr -CA root.crt -CAkey server.key -out /tmp/postgresql.crt -CAcreateserial

Anyway, thanks for this excellent HOWTO as it is only using the user name and thus permits a real easy use when you don't have an official IP address nor domain :)

JYFB

By: gwynReply

You must add "clientcert=1" to hostssl options for checking the client certificates, otherwise everyone will be granted access in your setup:

hostssl all postgres 0.0.0.0/0 trust clientcert=1

See: PostgreSQL documentation, 17.9.1. Using Client Certificates

By: Jeff E MandelReply

If you want to use this with certificates from CACert.org, you are limited in what you can place in the CN of the certificate. The workaround is to create server certificates for users in your domain and use a map. Thus, I create a csr on my client machine:

cd ~/.postgresql

openssl req -out postgresql.csr -new -newkey rsa:2048 -nodes -keyout postgresql.key

Specifying CN=myusername.mydomain.org. Paste the CSR into cacert.org, and save the resulting certificate as ~/.postgresql/postgresql.crt (you might also need place the cacert.org root cert in that directory).

On the server, in pg\_hba:

hostnossl all all 0.0.0.0/0 reject

hostssl all all 0.0.0.0/0 cert map=ssl clientcert=1

In pg\_ident:

ssl /^(.\*).mydomain\.org$ \1

Now you can connect from the client:

psql -h server.mydomain.org

psql (9.4.6, server 9.4.1)

SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256, compression: off)

Next project - figure out how to make this work with JDBC

By: Slobodan VesovicReply

Tnx! Very helpful

By: francisReply

I just come back from a client demo. Yesterday evening everything was ok, but this morning my servers were down.

What happend is that my client certificate expired during the night. I'm so pissed off.

Please, update the last line of yout tuto :

openssl x509 -req -in /tmp/postgresql.csr -CA root.crt -CAkey server.key -out /tmp/postgresql.crt -CAcreateserial

should be

openssl x509 -days 3650 -req -in /tmp/postgresql.csr -CA root.crt -CAkey server.key -out /tmp/postgresql.crt -CAcreateserial

if you don't want your client cert to be short-lived.

Thanks

By: davidReply

Recent versions of postgres require this in postgresql.conf:

ssl\_ca\_file = 'root.crt'

By: ChetanReply

Can we enable ssl connection for IP based postgresql, means without domain name?

By: midipixReply

With modern pgAdmin4, you'd be better off installing pgAdmin4 on the same server where postgresql is running in server mode (`SERVER\_MODE = True` in `config\_local.py`), then require valid client certificates as part of your http server configuration. With nginx, that'd be, for instance:

`ssl\_client\_certificate /etc/easy-rsa/pki/ca.crt;`

`ssl\_verify\_client on;`

By: MiPhamReply

If I use PostgreSQL 10.16 on Windows 10.

What should "Set appropriate permission and the owner of the private key file" is?

icacls server.key /reset icals server.key /inheritance:r /grant:r "CREATOR OWNER:F"

Home

SSL Certificates For PostgreSQL

>

==============

-- Unit postgresql-14.service has begun starting up.

Nov 14 06:20:28 az-sin-m2p-preprod-db01.m2pfintech.com postmaster[370307]: 2021-11-14 06:20:28.625 UTC [370307] FATAL: could not load server certificate file "server.crt": ee key too small

Nov 14 06:20:28 az-sin-m2p-preprod-db01.m2pfintech.com postmaster[370307]: 2021-11-14 06:20:28.625 UTC [370307] LOG: database system is shut down

Nov 14 06:20:28 az-sin-m2p-preprod-db01.m2pfintech.com systemd[1]: postgresql-14.service: Main process exited, code=exited, status=1/FAILURE

Nov 14 06:20:28 az-sin-m2p-preprod-db01.m2pfintech.com systemd[1]: postgresql-14.service: Failed with result 'exit-code'.

-- Subject: Unit failed

-- Defined-By: systemd

-- Support: https://access.redhat.com/support

--

-- The unit postgresql-14.service has entered the 'failed' state with result 'exit-code'.

Nov 14 06:20:28 az-sin-m2p-preprod-db01.m2pfintech.com systemd[1]: Failed to start PostgreSQL 14 database server.

-- Subject: Unit postgresql-14.service has failed

-- Defined-By: systemd

-- Support: https://access.redhat.com/support

--

-- Unit postgresql-14.service has failed.

--

-- The result is failed.

~

openssl x509 -in server.crt -noout -text | grep BIT

=========

show ssl\_cert\_file;

show ssl\_key\_file;

show ssl\_ca\_file;

show ssl\_crl\_file;

This will show you how many bytes your slave is beh

clientcert=verify-full

clientcert=verify-ca

postgres=# \d pg\_stat\_ssl

View "pg\_catalog.pg\_stat\_ssl"

Column | Type | Collation | Nullable | Default

----------------+---------+-----------+----------+---------

pid | integer | | |

ssl | boolean | | |

version | text | | |

cipher | text | | |

bits | integer | | |

compression | boolean | | |

client\_dn | text | | |

client\_serial | numeric | | |

issuer\_dn | text | | |

Let us query the system view and see what it contains:

test=# \x

Expanded display is on.

test=# SELECT \* FROM pg\_stat\_ssl;

-[ RECORD 1 ]-+-----------------------

pid | 16378

ssl | t

version | TLSv1.3

cipher | TLS\_AES\_256\_GCM\_SHA384

bits | 256

compression | f

client\_dn |

client\_serial |

issuer\_dn |

The connection has been successfully encrypted. If “ssl = true”, then we have succeeded.

SHOW ssl;

SELECT \* FROM pg\_stat\_ssl;

psql "host=174.1.0.6 port=5432 user=readme dbname=postgres sslmode=verify-full sslcert=client.crt sslkey=client.key sslrootcert=ca.crt"

psql "host=174.1.0.6 port=5432 user=readme dbname=postgres sslmode=verify-ca sslrootcert=root.crt"

psql "host=174.1.0.6 port=5432 user=readme dbname=postgres sslmode=verify-full sslrootcert=/dbdata/pgsql/data/root.crt sslcert=/tmp/postgresql.crt sslkey=/tmp/postgresql.key "

psql -h 174.1.0.6 -p 5432 -U readme postgres -sslrootcert=root.crt

/dbdata/pgsql/data/root.crt /tmp/postgresql.crt /tmp/postgresql.KEY

-----------------

FINAL

--IND2 TO IND1

psql "host=174.1.0.6 port=5432 user=readme dbname=postgres sslmode=verify-ca sslrootcert=/tmp/root.crt sslcert=/tmp/postgresql.crt sslkey=/tmp/postgresql.key "

--IND1 to IND2

psql "host=174.1.0.7 port=5432 user=readme dbname=postgres sslmode=verify-ca sslrootcert=/tmp/ind2\_client/root.crt sslcert=/tmp/ind2\_client/postgresql.crt sslkey=/tmp/ind2\_client/postgresql.key "

create user m2pappdb2 WITH login password 'M@PS0luti0ns' ;

alter user m2pappdb2 with superuser;

alter user m2pappdb2 with password 'M@PS0luti0ns';

select \* from guestbook;

INSERT INTO guestbook (visitor\_email, date, message) VALUES ('notherdummy.com', current\_date, 'Now we are ssl10.');

create user m2p331 WITH login password 'M@PS0luti0ns' ;

alter user m2p331 with superuser;

GRANT SELECT ON pg\_catalog.pg\_file\_settings TO pg\_read\_all\_settings;

GRANT EXECUTE ON FUNCTION pg\_catalog.pg\_show\_all\_file\_settings() TO pg\_read\_all\_settings;

grant select,insert,update,delete on all tables in schema recon to m2pappdb2;

grant select,usage,update on all sequences in schema recon to m2pappdb2;

grant execute on all functions in schema recon to m2pappdb2;

grant references, trigger on all tables in schema recon to m2pappdb2;

grant create on schema recon to m2pappdb2;

grant usage on schema recon to m2pappdb2;

ables:

select tablename from pg\_tables where schemaname = 'public';" YOUR\_DB` ; do psql -c "alter table \"$tbl\" owner to NEW\_OWNER" YOUR\_DB

for tbl in `psql -qAt -c "select sequence\_name from information\_schema.sequences where sequence\_schema = 'public';" YOUR\_DB` ; do psql -c "alter sequence \"$tbl\" owner to NEW\_OWNER" YOUR\_DB ; done

Views:

for tbl in `psql -qAt -c "select table\_name from information\_schema.views where table\_schema = 'public';" YOUR\_DB` ; do psql -c "alter view \"$tbl\" owner to NEW\_OWNER" YOUR\_DB ; done

You could probably DRY that up a bit since the alter statements are identical for all three.

SELECT format(

'ALTER TABLE %I.%I.%I OWNER TO %I;',

table\_catalog,

table\_schema,

table\_name,

postgres

)

FROM information\_schema.tables

WHERE table\_schema = 'recon';

select tablename, tableowner from pg\_catalog.pg\_tables where schemaname = 'public' ;

postgres=> \dn

List of schemas

Name | Owner

---------------+---------------

cron | rds\_superuser

public | postgres

recon | postgres

statusmanager | postgres

(4 rows)

postgres=> ALTER SCHEMA recon OWNER TO m2pappdb2;

ALTER SCHEMA

postgres=> \dn

List of schemas

Name | Owner

---------------+---------------

cron | rds\_superuser

public | postgres

recon | m2pappdb2

statusmanager | postgres

(4 rows)

postgres=> ALTER SCHEMA statusmanager OWNER TO m2pappdb2;

ALTER SCHEMA

postgres=> \dn

List of schemas

Name | Owner

---------------+---------------

cron | rds\_superuser

public | postgres

recon | m2pappdb2

statusmanager | m2pappdb2

(4 rows)