STIG Postgresql 12

Micah Halter

March 23, 2020

Set Up Environment Variables

Find the `postgresql` data folder and version

```
$~ sudo su - postgres
$~ psql -c "SHOW data_directory"
$~ psql -c "SHOW server_version"
```

As a DB administrator, add the following lines to `~/.bashrc` (Updated for your `postgresql` installation

```
export PATH="/usr/pgsql-12/bin:$PATH"
export PGDATA='/var/lib/pgsql/12/data'
export PGVER=12
```

Set up pgaudit

Install `pgaudit` extension with

```
$~ sudo yum install pgaudit14_12
```

Change `shared_preload_libraries` line in `${PGDATA}/postgresql.conf` to

```
shared_preload_libraries = 'pgaudit'
```

Add `pgaudit` configuration options to the bottom of `${PGDATA}/postgresql.conf`

```
#------------------------------------------------------------------------------
# PGAUDIT OPTIONS
#------------------------------------------------------------------------------

# Enable catalog logging - default is 'on'
pgaudit.log_catalog='on'

# Specify the verbosity of log information (INFO, NOTICE, LOG, WARNING, DEBUG)
pgaudit.log_level='log'
```

```
# Log the parameters being passed
pgaudit.log_parameter='on'

# Log each relation (TABLE, VIEW, etc.) mentioned in a SELECT or DML statement
pgaudit.log_relation='off'

# For every statement and substatement, log the statement and parameters
pgaudit.log_statement_once='off'

# Define the master role to use for object logging
# pgaudit.role=''
# Choose the statements to log:
# READ - SELECT, COPY
# WRITE - INSERT, UPDATE, DELETE, TRUNCATE, COPY
# FUNCTION - Function Calls and DO Blocks
# ROLE - GRANT, REVOKE, CREATE/ALTER/DROP ROLE
# DDL - All DDL not included in ROLE
# MISC - DISCARD, FETCH, CHECKPOINT, VACUUM
pgaudit.log='ddl,role,write,read'
```

Set up Logging

Verify the following logging settings in `${PGDATA}/postgresql.conf`

```
log_destination = 'syslog'
logging_collector = on
log_directory = 'log'
log_filename = 'postgresql-%a.log'
log_file_mode = 0600
log_truncate_on_rotation = on
log_rotation_age = 1d
log_rotation_size = 0

syslog_facility = 'LOCAL0'
syslog_ident = 'postgres'

log_checkpoints = on
log_connections = on
log_disconnections = on
log_duration = off
log_error_verbosity = default
log_hostname = on
log_line_prefix = '%m %u %d: '
log_lock_waits = on
log_statement = 'none'
```

```
log_timezone = 'America/New_York'

client_min_messages = notice
log_min_messages = warning
log_min_error_statement = error
log_min_duration_statement = -1
```

Add the following rule to `/etc/rsyslog.conf`

```
local0.*    /var/log/postgresql
```

Install pgcrypto

Run

```
$~ sudo su - postgres
$~ psql -c "CREATE EXTENSION pgcrypto"
```

Setup SSL

Create SSL Certificates

```
# Create Self-Signed certificate
$~ openssl genrsa -aes256 -out ca.key 4096
$~ openssl req -new -x509 -sha256 -days 1825 -key ca.key -out ca.crt \
   -subj "/C=US/ST=GA/L=Atlanta/O=GTRI/CN=root-ca"

# Create Server Intermediate Certificate
$~ openssl genrsa -aes256 -out server-intermediate.key 4096
$~ openssl req -new -sha256 -days 1825 -key server-intermediate.key \
   -out server-intermediate.csr \
   -subj "/C=US/ST=GA/L=Atlanta/O=GTRI/CN=server-im-ca"
$~ openssl x509 -extfile /etc/pki/tls/openssl.cnf -extensions v3_ca \
   -req -days 1825 -CA ca.crt -CAkey ca.key -CAcreateserial \
   -in server-intermediate.csr -out server-intermediate.crt

# Create Client Intermediate Certificate
$~ 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=GA/L=Atlanta/O=GTRI/CN=client-im-ca"
$~ openssl x509 -extfile /etc/pki/tls/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 Certificate
```

```
# replace dbase01 with hostname:
$~ openssl req -nodes -new -newkey rsa:4096 -sha256 -keyout server.key \
   -out server.csr -subj "/C=US/ST=GA/L=Atlanta/O=GTRI/CN=dbase01"
$~ openssl x509 -extfile /etc/pki/tls/openssl.cnf -extensions usr_cert \
   -req -days 1825 -CA server-intermediate.crt -CAkey server-intermediate.key \
   -CAcreateserial -in server.csr -out server.crt

# Create Client Certificate
# client cert must be mapped to a postgres role
$~ openssl req -nodes -new -newkey rsa:4096 -sha256 -keyout client.key \
   -out client.csr -subj "/C=US/ST=GA/L=Atlanta/O=GTRI/CN=ident_map"
$~ openssl x509 -extfile /etc/pki/tls/openssl.cnf -extensions usr_cert \
   -req -days 1825 -CA client-intermediate.crt -CAkey client-intermediate.key \
   -CAcreateserial -in client.csr -out client.crt

$~ cat server.crt server-intermediate.crt ca.crt > ./server-full.crt
# place ca.crt, server.key, server-full.crt in the $PGDATA directory
# chown them postgres:postgres
# chmod them 600
```

Configure Postgresql to use SSL

```
# Create role named same as client certificate
$~ psql -c "CREATE ROLE ident_map LOGIN"
```

Configure `ssl` certificates in `$PGDATA/postgresql.conf` (update file paths as needed):

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

Configure `$PGDATA/pg_ident.conf`:

```
ssl-test    ident_map    identmap
```

Configure `$PGDATA/pg_hba.conf` (replace `dbase01` with host name):

```
hostssl     all     all     dbase01/32    cert clientcert=1 map=ssl-test
hostssl     all     all     ::1/128       cert clientcert=1 map=ssl-test
```

Configure Clients

Copy the certificates on the client (update paths as needed)

```
$~ mkdir $CLIENT_HOME/.postgresql
$~ cp ca.crt $CLIENT_HOME/.postgresql/root.crt
$~ cp client.key $CLIENT_HOME/.postgresql/postgresql.key
```

```
$~ cat client.crt client-intermediate.crt ca.crt \
   > $CLIENT_HOME/.postgresql/postgresql.crt
$~ chmod 600 $CLIENT_HOME/.postgresql/*
```

Test the client (replace `dbase01` with host name):

```
$~ psql "postgresql://dbase01:5432/postgres?sslmode=verify-full" -U ident_map
```

Make sure all of the client roles inherit the `ident_map` role, and the ssl certificates should work.

Run the ansible

Set the defaults in `roles/disa-v1r6/defaults/main.yml`

```
$~ sudo ansible-playbook playbook.yml
```

TODO

- `PGS9-00-002700`: real-time alerts on audit failures. If we set this up for `rsyslog`, then it should automatically work
- `PGS9-00-008000`: FIPS enabled (other rules need this as well)
- `PGS9-00-009900`: 75% audit capacity warning, still going to `rsyslog`. If we set that up with 75% audit capacity warning, we should be good.
- `PGS9-00-011600`: cron script to terminate sessions after defined trigger events