The Three Musketeers (Authentication, Authorization, & Accounting)

Sarah Conway

credativ USA
info@credativ.com

September 18, 2014

Agenda

AAA Model

- AAA Model - Framework that can identify users, authorize what they can access, and create audit trails
  - Authentication - Server verifies the user is who they claim to be
  - Authorization - Determines what authenticated user can access and modify
  - Accounting - Records what user accesses, what actions are performed, and date/time of access

postgresql.conf overview

- Located by default on Debian in /etc/postgresql/version/main/
- or whatever directory $PGDATA is for you
- Locate in postgres session as superuser
    - SHOW data_directory;
    - SHOW config_file;
- Comment = #
- `www.postgresql.org/docs/9.3/static/config-setting.html`

postgresql.conf Security and Authentication

```
#authentication_timeout = 1min
ssl = true
#ssl_ciphers = 'DEFAULT:!LOW:!EXP:!MD5:@STRENGTH'
#ssl_renegotiation_limit = 512MB
ssl_cert_file = '/etc/ssl/certs/ssl-cert-snakeoil.pem'
ssl_key_file = '/etc/ssl/private/ssl-cert-snakeoil.key'
#ssl_ca_file = ''
#ssl_crl_file = ''
#password_encryption = on
#db_user_namespace = off
```

postgresql.conf Security and Authentication

```
# Kerberos and GSSAPI
#krb_server_keyfile = ''
#krb_srvname = 'postgres'
#krb_caseins_users = off
```

pg_settings

- Alternate way to view postgres server settings
- Primarily same options as are available in postgresql.conf
- Context column
  - internal - All internal server values, cannot be changed directly
  - postmaster - If changed, requires restart
  - sighup - If changed, requires reload
  - superuser - Can only be changed by superusers in a session
  - user - Can be changed by any user in a session
- `www.postgresql.org/docs/9.3/static/view-pg-settings.html`

pg_settings

```
test=# select * from pg_settings where name in ('authentication_timeout');
-[ RECORD 1 ]----------------------------------------------------------------
name        | authentication_timeout
setting     | 60
unit        | s
category    | Connections and Authentication / Security and Authentication
short_desc  | Sets the maximum allowed time to complete client authentication.
extra_desc  |
context     | sighup
vartype     | integer
source      | default
min_val     | 1
max_val     | 600
enumvals    |
boot_val    | 60
reset_val   | 60
sourcefile  |
sourceline  |
```

pg_settings

```
test=# select name, setting, context, source from pg_settings where name in
('authentication_timeout');
          name          | setting | context | source
------------------------+---------+---------+---------
 authentication_timeout | 60      | sighup  | default
(1 row)

test =# \x
Expanded display now on.

test=# select name, setting, context, source from pg_settings where name in
('authentication_timeout');
-[ RECORD 1 ]-------------------
name    | authentication_timeout
setting | 60
context | sighup
source  | default
```

CREATE ROLE with LOGIN

- Same as CREATE USER
- Creates username/password pair
- Authentication-based parameters
  - username, password, password expiration/encryption settings
- Create user with password valid until October 10th, 2014:

```
CREATE ROLE sauron LOGIN PASSWORD 'nazgul' VALID UNTIL '2014-10-01';
```

- Drop user:

```
DROP ROLE sauron;
```

pg_hba.conf overview

- pg_hba.conf - configuration file that controls client authentication/authorization
- Located by default on Debian in /etc/postgresql/version/main/ or wherever $PGDATA is
- Ask postgres in superuser session 'SHOW hba_file;' to locate
- Specifies connection type, client IP address range, database name, user name, and authentication method used for matching connections
- `www.postgresql.org/docs/9.3/static/auth-pg-hba-conf.html`

auth-method parameters

- *auth-method* - Specifies authentication method for use when match is found
    - *trust* - Allows full access to user; can login as any existing user
    - *reject* - Rejects all access to specific connections/hosts
    - *md5* - Requires user to provide password
        - Password is md5-salted-and-hashed by client
    - *password* - Requires user to provide password
        - Password stored/sent in clear-text
    - *gss* - Uses GSSAPI
        - TCP/IP connections only
    - *sspi* - Uses SSPI
        - Windows OS only
    - *krb5* - Uses Kerberos V5
        - TCP/IP connections only

auth-method parameters, cont.

- *ident* - Contacts ident server on client, checks if client username matches database user name
    - TCP/IP connections only
- *peer* - Checks for match between client username and database user name
    - Local connections only
- *ldap* - Uses LDAP server
- *radius* - Uses RADIUS server
- *cert* - Uses SSL client certificates
- *pam* - Uses Pluggable Authentication Modules (PAM) service
- *auth-options* - Fields of the form name=value specify options for selected authentication method

SSL - Overview

- Normally used as a standard security technology for encrypting network connections
- Also used for authenticating users with certificates
- Certificate issued by CA who authenticates user using a cryptographic public key
- Verifier cannot impersonate user
- Separates user from authentication method; not vulnerable to phishing
- Two-factor authentication recommended
- `www.postgresql.org/docs/9.3/static/ssl-tcp.html`

SSL

- Requires setting 'ssl' to 'on' in postgresql.conf
- Requires installation of SSL certificates on client/server
- Files containing server certificate and private key must exist
  - Named server.crt and server.key by default
  - Located in server's data directory
  - Can rename or relocate by modifying ssl_cert_file and ssl_key_file

SSL - Server File Usage

- The following files (named by default) are relevant to SSL setup on server -
  - ssl_cert_file - $PGDATA/server.crt
    - Contents - server certificate
    - Sent to client to identify server
  - ssl_key_file - $PGDATA/server.key
    - Contents - server private key
    - Proves server certificate was sent by owner without showing if certificate owner is trusted
  - ssl_ca_file - $PGDATA/root.crt
    - Contents - trusted certificate authorities
    - Checks that client certificate is signed by trusted CA
  - ssl_crl_file - $PGDATA/root.crl
    - Contents - certificates revoked by certificate authorities.
    - Lists blocked certificates

SSL - Publicly Signed Certificates

- Verifies existence of the business, domain ownership, and user's authority
  - Generate a cert signing request
  - Submit CSR to the CA using their process, pay
  - Wait for them to sign
  - Download signed cert, install CA chain/signed cert with previously generated private key
- Domain Validated certificates
  - Entry-level
  - Issued quickly
  - Verifies only that the applicate owns domain name

Creating Self Signed Certificates

```
sudo su -
cd /your/data/directory
openssl genrsa -des3 -out server.key 1024
ssl rsa -in server.key -out server.key
chmod 400 server.key
chown postgres.postgres server.key
```

Creating Self Signed Certificates

```
openssl req -new -text -out server.req
openssl req -x509 -in server.req -text -key server.key -out server.crt
cp server.crt root.crt

#use text editor (vim, vi, etc) to edit pg_hba.conf
#add following lines
hostssl all www-data 0.0.0.0/0
hostssl all postgres 0.0.0.0/0

#use text editor (vim, vi, etc) to edit postgresql.conf
ssl = on

#restart postgres
restart service postgresql
```

pg_hba.conf

- Default Debian pg_hba.conf:

```
# Database administrative login by UNIX sockets
local   all         postgres                          peer

# TYPE  DATABASE    USER        CIDR-ADDRESS          METHOD

# "local" is for Unix domain socket connections only
local   all         all                               peer
# IPv4 local connections:
host    all         all         127.0.0.1/32          peer
# IPv6 local connections:
host    all         all         ::1/128               peer
```

pg_hba.conf

```
#Example pg_hba entries:
#Single host allowed
host all all  192.168.1.10/32 trust

#Single host rejection
host all  all  192.168.1.10/32 reject

#Single host connection to single database
host foo all  192.168.1.10/32 md5

#Small network connection
host all  all  192.168.1.0/28 trust

#Larger network connection
host foo all  192.168.1.0/24  trust
```

CREATE ROLE with NOLOGIN

- Same as CREATE GROUP
- Creates group with particular privileges that users can be assigned to
- Authorization-based parameters (also applies to CREATE ROLE with LOGIN)
  - replication, createdb, createrole, superuser
- Create user that is a superuser:

```
CREATE ROLE saruman LOGIN SUPERUSER;
```

- Create administrative group and assign saruman to it:

```
CREATE ROLE admin NOLOGIN SUPERUSER;
GRANT admin TO saruman;
ALTER ROLE saruman INHERIT;
\c - saruman
set role admin;
```

GRANT/REVOKE

- Define/remove access privileges to database objects
  - Can grant privileges on tables, columns, views, databases, sequences, domains, foreign data wrappers, foreign servers, functions, procedural languages, large objects, schemas, tablespaces, types
  - Schema level privileges disabled by default
- Grant/revoke role membership
- `www.postgresql.org/docs/9.3/static/sql-grant.html`
- `www.tutorialspoint.com/postgresql/postgresql_privileges.htm`

GRANT - Example

- Grant all privileges on schema mordor to group role admin:

```
CREATE SCHEMA mordor;
CREATE TABLE mordor.ring(id int);
GRANT ALL PRIVILEGES ON SCHEMA mordor TO admin;
```

REVOKE - Example

```
REVOKE ALL PRIVILEGES ON SCHEMA PUBLIC FROM saruman;

REVOKE ALL ON FUNCTION foo() FROM GROUP PUBLIC;

REVOKE ALL PRIVILEGES ON SCHEMA PUBLIC FROM PUBLIC;
```

ALTER DEFAULT PRIVILEGES

- Define your own default privileges
- DROP OWNED BY to drop default privilege entry for role
  - Required to drop role with changed default settings
- Grant SELECT to public for all tables created under schema mordor:

```
ALTER DEFAULT PRIVILEGES IN SCHEMA mordor
GRANT SELECT ON TABLES TO PUBLIC;
```

Access Privilege Inquiry Functions

- pg_has_role
- has_any_column_privilege
- has_database_privilege
- has_column_privilege
- has_schema_privilege
- etc. for function, foreign_data_wrapper, sequence, table, tablespace
- If user argument omitted, current_user is assumed
- `www.postgresql.org/docs/9.3/static/functions-info.html`

Access Privilege Inquiry Functions

```
test=#SELECT has_table_privilege('frodo','mordor.ring','select');

 has_table_privilege
---------------------
 t
(1 row)
```

psql

- \dp - Obtains information about current privileges for existing database objects
- \ddp - Obtains information about default privilege assignments
- \du - Obtains information about the list of existing roles
- All are only available in psql
- `www.postgresql.org/docs/9.3/static/app-psql.html`

postgresql.conf

- log_destination, log_directory, log_filename
  - Locate logs
- log_connections, log_pid, log_statement, log_duration, log_timestamp
  - Logs respective items
- debug_print_parse, debug_print_rewritten, debug_print_plan
  - Enables various debugging output to be sent to server log
- debug_pretty_print
  - Sends debugging output in an longer, indented, more readable format
- hostname_lookup
  - Shows hostname in logs

csvlog

- Displays log lines in files, with abilty to import into table
- Efficient way to view important logs at once
- Displays concise list of information with options to add or remove specified files
    - Time stamp, username, databse name, PID, SID, client host:port, per-session line number, command tag, session start, virtual/regular transaction IDs, error severity, etc...

Importing csvlog

- COPY postgres_log FROM '/full/path/to/logfile.csv' WITH csv;
- Set log_filename and log_rotation age to predict what filename will be, and when files are ready for import
- Set log_truncate_on_rotation to avoid mixing old data with new

Event Triggers

- Newly introduced in 9.3, still being expanded
- Capable of capturing DDL events
- Global to a specified database
- Can be written in any procedural language with event trigger support

pgaudit

- `https://github.com/2ndQuadrant/pgaudit`
- Based on event triggers
- Collects audit events and logs in CSV log format
- Supports DDL, DML, and utility commands

audit-trigger

- `https://github.com/2ndQuadrant/audit-trigger`
- Attached to a single table
- Captures DML events only
- Script generates an audit trigger for each table in database
- Easily modifiable

pgbadger

- `https://github.com/dalibo/pgbadger`
- Add-on that analyzes logs and compiles results into csvlog, syslog, or stderr
- Built to be quick
- Written in Perl
- Mostly performance reports

Questions?

Thank You!