Authentication in PostgreSQL

Michael Paquier – VMware

2018/05/31, PGCon 2018

The man

- Michael Paquier.
- French, based in Tokyo.
- PostgreSQL contributor since 2009
  - Some patches, some reviews and some bug fixes.
  - Blogging.
- Working at VMware on PostgreSQL
  - Packaging.
  - Integration.
  - Support.

Authentication methods

- Password
  - Plain text
  - MD5
  - SCRAM-SHA-256
  - RADIUS, ldap, pam, BSD…
- Certificates
- Kerberos, SSPI (Windows)
- peer
- https://www.postgresql.org/docs/current/static/auth-methods.html

Code location

- Backend, src/backend/libpq
  - auth.c, auth-scram.c for authentication.
  - be-secure*.c for SSL.
  - hba.c for administration.
- Frontend (libpq), src/interfaces/libpq:
  - fe-auth.c, fe-auth-scram.c for authentication.
  - fe-secure*.c for SSL.

pg_hba.conf

- Administration policy with filter sets
  - User
  - Database
  - Host
  - Type
- Controls authentication and connection policies.
- Order-dependent:
  - First match wins.
  - Place the most specific first.
- Also listen_addresses!

pg_ident.conf

- User name mapping
  - Map name
  - OS user
  - Database user
- Useful for GSSAPI, peer.
- regex support
- Additional field map=hoge in pg_hba.conf

pg_service.conf

- Centralize connection parameters for clients.
- PGSERVICEFILE, and *no* connection parameters
- Say a local service connecting to Postgres
- Connection parameter “service=archiver” or PGSERVICE

```
[archiver]
host=$DB_HOST_OR_SOCKET_DIR
port=$DB_PORT
user=$DB_USER
```

- Use with pg_ident.conf!

Trust method

- No security at all.
- Simply allow connections to come any
  - Anybody
  - Anywhere (can filter by IP)
- Use cases
  - Unix domain sockets (local) for debugging.
  - Personal laptop and development.

Plain text

- Password sent in clear text

```
Server: Please send your password
Client: “hoge”
Server: OK, good to go
```

- Use SSL!
- Weak to password sniffing, across network.

MD5

- Password hash sent:

```
Server: Here is a salt (4 random bytes),
           please compute md5(md5(password || username), salt)
Client: “ad22f1df5331cfa7603c67a2092c6159”
Server: OK, good to go
```

- Again use SSL!
- Issues
  - User rename
  - Bad and weak reputation (see community lists).
  - Contents of pg_authid

Attacking MD5 hash

- Guess attack
  - Hash calculation is fast (Millions per second)
- Replay attack
  - Salt is 4 bytes
  - 4-billion possibilities
- Pass-the-hash
  - Connection possible just by knowing the stored hash.
  - Old backups lying around?

SCRAM-SHA-256

- Challenge-based exchange, added in v10.

```
Client: Here is a random nonce (18 bytes)
  r=ReZeIvordKIQsS5/uybHrLKa

Server: Here is my random nonce, salt and iteration count
  r=ReZeIvordKIQsS5/uybHrLKaJ4YZ83N/PitA0fx0eEmj1Gro,
  s=aqgRYGF+L5LUrYpej98rgA==,
  i=4096

Client: Proof that I know the password.
  p=O/BAMj7s/fbE5UvMKfhXRmObj/s2hlD23sMqUIlIsxk=

Server: Proof that I also know the password.
  v=JyGOhjHVCnLjCbJuC/XTICPPQFQ2fGk8+sCbSq2g+5I=
```

SCRAM security

- Replay attacks => longer nonces
- Hash stored in pg_authid cannot be used directly.
- Dictionary attacks
  - Iteration count can be used as parameter
  - Computation of connection proof is costly (cost at connection startup)
- Still use SSL.

Client/server and HBA entries

- With password, md5 and scram-sha-256…

	hba configuration		
Verifier type	**password**	**md5**	**scram-sha-256**
MD5	O [1]	O	X
SCRAM	O [1]	O [2]	O

- [1]: Plain text is used, hash generated server-side.
- [2]: SCRAM is used.

SCRAM Channel binding

- MITM prevention, by “binding” FE/BE
- RFC 5929: https://tools.ietf.org/html/rfc5929
- Ensure that the point where a connection is done is still the same.
- Channel types:
  - unique: a specific connection is sure to be used.
  - endpoint: the endpoints are the same.

Channel binding for Postgres

- Added in Postgres 11.
- Two channel types
  - tls-unique, ensure that using a hash of the TLS end message.
  - tls-server-end-point, using a hash of server certificate (useful for JDBC).
- OpenSSL, gnuTLS have support.
- Macos and Windows not directly.
- Connection parameter scram_channel_binding
  - Default is “tls-unique”
  - Empty value disables channel binding.
  - Choice left to the client, server advertises it.
- Protocol changes needed again in drivers!

Driver support

- Be careful with authentication choice and the client interface used!
- JDBC, npgsql with SCRAM (+ channel binding!)
- Things linking with libpq:
  - ODBC
  - psycopg2, etc.
- Gets complicated with large product integration.
- https://wiki.postgresql.org/wiki/List_of_drivers

Peer

- Unix socket connections (local)
  - No Windows here.
- Relies on kernel call getpeereid()
- Use with pg_ident.conf and static service files.
  - Local WAL archiver.
  - Cron diagnostic tool (or background worker).
  - No need for superuser!

LDAP

- Server-side implementation
- Useful for large organizations
- Cleartext password seen from client
- Format supported
  - prefix+suffix, or simple bind
  - search+bind
- Use SSL: ldaptls=1 and hostssl
- Password policies with ppolicy

LDAP, new as of v11

- Addition of LDAPS
  - LDAP + StartTLS is standard
  - New parameter ldapscheme
- ldapsearchfilter
  - More flexible than ldapsearchattribute
  - ldapsearchfilter="(|(uid=$username)(mail=$username))"
  - $username as magic value

GSS/SSPI

- Uses Kerberos.
  - Active directory available
  - No password prompt.
- User mapping with pg_ident.conf.
- Again use SSL!
- No support for GSSAPI encryption
  - Patch submitted for v10, not merged.
  - Requires low-level surgery for message exchange.
  - Requires equivalent of sslmode.

Certificates

- No password prompt.
- CN field checked for match with database user.
- User mapping in pg_ident.conf.
- Only over SSL.
- Client needs to use trusted certificate.
- Documentation improvements in v11 (see 815f84aa)
- Use v3_ca for intermediate certificates

Superusers

- Never use them, except if you really can’t.
- System function ACLs!
  - Grant execution and access to specific users
  - pg_rewind not requiring superuser
  - System roles at the rescue

Some extras

- PAM
  - password for the client.
  - SSL, again!
  - PAM through LDAP with pam_ldap.
- BSD
  - password for the client.
  - Added in 9.6.
  - OpenBSD only.

SSL negotiation

- Server sends options.
- Client decides.
- Controlled by:
  - sslmode, connection parameter
  - PGSSLMODE, environment variable

Security with sslmode

Modes	Protection		Server-side SSL	
Verifier type	**Eavesdropping**	**MITM**	**Disabled**	**Required**
disable	X	X	O	X
allow	X	X	O	O
prefer (default!)	X	X	O	O
require	O	X	X	O
verify-ca	O	O	X	O
verify-full	O	O	X	O

Authentication tests

- src/test/
  - authentication/, hba and SCRAM (SASLprep)
  - kerberos/
  - ldap/
  - ssl/, certificates and channel binding
- PG_TEST_EXTRA
- PROVE_TESTS

Questions?