SCRAM authentication

Michael Paquier – VMware

2017/12/06, PGConf Asia 2017

Authentication methods

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

Plain text

- Password sent in clear text

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

- Can be used with SSL!
  - sslmode = verify-full
  - sslmode = prefer, the default, is an abomination
- 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

- Can still be used with SSL!
- Issues
  - User rename
  - MD5 is said to be ***bad*** (see community lists).

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?

Deprecated features in v10

- Removal
  - Password_encryption = off/plain is removed
  - Plain text entries in pg_authid
  - Createdb --unencrypted
- libpq
  - PQencryptPassword() is deprecated.
  - Use PQencryptPasswordConn().
  - psql’s \password uses it and sets password hash to password_encryption.

About SCRAM

- **S**alted **C**hallenge **R**esponse **A**uthentication **M**echanism
- Implementation of SCRAM-SHA-256
- RFCs
  - 5802: https://tools.ietf.org/html/rfc5802
  - 7677: https://tools.ietf.org/html/rfc7677
- New in PostgreSQL 10
- RFC compliant
  - Note that username is sent empty

SCRAM protocol

- Challenge-based exchange

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/s2hlD23sMqUIIlsxk=

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

SCRAM security

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

SCRAM-SHA-256

- SCRAM originally uses SHA-1.
- SHA-256,384 and 512 functions available in src/common/sha2*.c.
- Only uses SHA-256 as hash function.

Verifiers

- Still one verifier per role.
- No dependency on role name (no removal on rename).
- Complicates upgrade scenarios.
- Controlled by password_encryption
  - Default to ‘md5’

```
=# SET password_encryption TO 'md5';
SET
=# CREATE ROLE role_md5 PASSWORD 'hoge';
CREATE ROLE
=# SET password_encryption TO 'scram-sha-256';
SET
=# CREATE ROLE role_scram PASSWORD 'hoge';
CREATE ROLE
```

Format

- Usable with LDAP
- SCRAM-SHA-256$<iteration count>:<salt>$<StoredKey>:<ServerKey>

```
=# SELECT rolname, rolpassword FROM pg_authid
     WHERE rolname ~ 'role';
  rolname   |                 rolpassword
------------+---------------------------------------------
 role_md5   | md5927f6dffb8b758965daa42fb9a868958
 role_scram | SCRAM-SHA-256$4096:nEYxO97qeT9i89Zrkegox
w==$aSEM7ph+TydluWFRXFLSJ9Aqen2qtGw/lkH1rePRYBk=:w2pW4
8qqsaMvPfzWZmUAbeNoOctBnK2myx35XM7XJLo=
(2 rows)
```

pg_hba.conf

```
# Local connections for Unix domain sockets
# TYPE  DATABASE     USER     ADDRESS          METHOD
local   all          all                       trust

# md5 authentication with SSL from dev machines
hostssl all          all      dev.example.com  md5

# SCRAM for the rest, still with SSL
hostssl all          all      all              scram-sha-256
```

- https://www.postgresql.org/docs/current/static/auth-pg-hba-conf.html

Support server-client

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

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

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

[2]: SCRAM is used.

Extra work with v10

- Middleware support
  - Pgbouncer (patch available on github)
  - Pgpool
- Drivers: https://wiki.postgresql.org/wiki/List_of_drivers
  - ODBC, stuff using libpq => OK
  - JDBC supports protocol

Future improvements

- LDAP extension, storage of SCRAM verifier
- Channel binding
- Iteration count configurable
- Not possible to enforce scram in libpq, rogue servers can force downgrades :(

Upgrade to SCRAM

- Update pg_hba.conf (not mandatory)
  - Password, md5 to scram-sha-256
- Client updates, only libpq >= v10 supports SCRAM
- password_encryption = ‘scram-sha-256’ in postgresql.conf.
- Change user passwords.
- Note: still only one password per role :(

SCRAM and encryption

- SCRAM = authentication, **not** encryption.
- Please still use with sslmode=verify-full in v10.
- With channel binding, setup gets easier.

About SASL

- **S**imple **A**uthentication and **S**ecurity **L**ayer
- RFC 4422: https://tools.ietf.org/html/rfc4422
- “Framework for authentication and data security in Internet protocols. It decouples authentication mechanisms from application protocols, in theory allowing any authentication mechanism supported by SASL to be used in any application protocol that uses SASL.”
- SCRAM is a SASL mechanism.
- Client drivers could use generic SASL and SCRAM implementation as Postgres is RFC-compliant.
- DIGEST-MD5 or others could be added.

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

- 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 has support (fancy research).
- Macos, gnuTLS (?) and Windows not directly.
- Connection parameters
  - Channel binding name.
  - Allow client to choose with or without.
- Patch for v11 in review.

Questions?