

PostgreSQL Security Hardening

End-to-End Security Hardening Guide for Production Databases

PostgreSQL 15 / 16 / 17 compatible

Updated: October 2026 · Production-grade



01 Authentication pg_hba.conf MOST CRITICAL

METHOD	STATUS	DESCRIPTION
scram-sha-256	RECOMMENDED	Challenge-response, salted hash. Default (PG14+).
cert	RECOMMENDED	Client certificate authentication (hostssl).
peer	Local	Unix socket — OS user name mapping.
md5	LEGACY	Weak hash, replay risk. Deprecated in PG18.
password	FORBIDDEN	Password is sent in clear text.
trust	FORBIDDEN	Passwordless login! Isolated localhost only.

SECURE PG_HBA.CONF

#	TYPE	DATABASE	USER	ADDRESS	METHOD
	local	all	postgres		peer
	local	all	all		scram-sha-256
	hostssl	app_db	app_user	10.0.1.0/24	scram-sha-256
	hostssl	all	dba_team	10.0.9.0/28	cert
	hostssl	replication	repl	10.0.2.11/32	scram-sha-256
	hostnossl	all	all	0.0.0.0/0	reject
# REMOVE ↓					
	host	all	all	0.0.0.0/0	trust

APPLY & VERIFY

```
-- Apply without restart
SELECT pg_reload_conf();

-- Check rules and errors
SELECT line_number, type, database,
       user_name, address, auth_method, error
FROM   pg_hba_file_rules
ORDER BY line_number;

-- Find users with md5 hashes
SELECT rolname FROM pg_authid
WHERE  rolpassword LIKE 'md5%';
```

⚠️ ⚠️ **Warning:** pg_hba.conf applies the **first matching line** — put restrictive rules on top. The **error** column must be empty before reload; otherwise the rules won't load.

02 Password Policy

PARAMETER / FEATURE	RECOMMENDED
password_encryption	scram-sha-256
passwordcheck	Enabled
VALID UNTIL	90 days
CREATEROLE	Restrict
LOGIN + NOINHERIT	Service acct.
auth_delay.milliseconds	500

POSTGRESQL.CONF

```
password_encryption      = 'scram-sha-256'
shared_preload_libraries = 'passwordcheck,auth_delay'
auth_delay.milliseconds = 500
authentication_timeout   = 30s
```

ROLE & PASSWORD MANAGEMENT

```
-- Human account: expiring password
CREATE ROLE ali LOGIN PASSWORD '0%7xQ!r2...'
      VALID UNTIL '2027-01-10';

-- Service account: no inheritance, connection limit
CREATE ROLE svc_api LOGIN NOINHERIT
      CONNECTION LIMIT 50 PASSWORD '...';

-- Revoke CREATEROLE privilege
ALTER ROLE app_admin NOCREATEROLE;

-- Expired / soon-to-expire passwords
SELECT rolname, rolvaliduntil
FROM   pg_roles
WHERE  rolcanlogin
      AND rolvaliduntil < now() + interval '14 days';
```

⚠️ ⚠️ **Warning:** passwordcheck can only validate passwords sent in plain text. psql \password hashes the password client-side — the check is skipped in that case.

✅ **Best Practice:** Since PG16, a CREATEROLE holder can only manage roles it has ADMIN OPTION on — audit the privilege chain regularly.

03 Network Security postgresql.conf

PARAMETER	DEFAULT	RECOMMENDED
listen_addresses	localhost	'localhost,10.0.1.5'
port	5432	5432 / custom
ssl	off	on
ssl_min_protocol_version	TLSv1.2	TLSv1.2+
ssl_ciphers	HIGH:MEDIUM...	HIGH:!aNULL:!MD5:!3DES
ssl_prefer_server_ciphers	on	on
max_connections	100	as needed

SSL/TLS CONFIGURATION

```
listen_addresses = 'localhost,10.0.1.5'
ssl              = on
ssl_cert_file    = 'server.crt'
ssl_key_file     = 'server.key' # chmod 600
ssl_ca_file      = 'root.crt'
ssl_min_protocol_version = 'TLSv1.2'
ssl_ciphers      = 'HIGH:!aNULL:!MD5:!3DES:!RC4'
```

FIREWALL (DEFENSE IN DEPTH)

```
# App subnet only → 5432
ufw allow from 10.0.1.0/24 to any port 5432 proto tcp
ufw deny 5432/tcp

# firewallld alternative
firewall-cmd --permanent --add-rich-rule='rule family=ipv4
source address=10.0.1.0/24 port port=5432 protocol=tcp accept'
```

SSL CHECK ON ACTIVE CONNECTIONS

```
SELECT a.username, a.client_addr, s.ssl, s.version, s.cipher
FROM   pg_stat_ssl s JOIN pg_stat_activity a USING (pid)
WHERE  a.backend_type = 'client backend';
```

✅ **Best Practice:** Firewall + listen_addresses + pg_hba.conf form a **three-layer defense**. If one is misconfigured, the others still protect you. Never expose the database directly to the internet.

04

Role & Privilege Management

Built-in Role (Predefined)	Usage
pg_read_all_data	SELECT on all tables (PG14+) — reporting
pg_write_all_data	INSERT/UPDATE/DELETE on all tables (PG14+)
pg_monitor	Monitoring tools — no superuser needed
pg_signal_backend	Cancel queries / terminate sessions
pg_checkpoint	Run CHECKPOINT (PG15+)
pg_maintain	VACUUM, ANALYZE, REINDEX (PG17+)

LEAST PRIVILEGE + PUBLIC SCHEMA

```
-- Lock down public schema (PG15+ CREATE off by default)
REVOKE ALL ON SCHEMA public FROM PUBLIC;
REVOKE ALL ON DATABASE app_db FROM PUBLIC;

-- Group role → grant to user
CREATE ROLE app_ro NOLOGIN;
GRANT CONNECT ON DATABASE app_db TO app_ro;
GRANT USAGE ON SCHEMA app TO app_ro;
GRANT SELECT ON ALL TABLES IN SCHEMA app TO app_ro;
ALTER DEFAULT PRIVILEGES IN SCHEMA app
    GRANT SELECT ON TABLES TO app_ro;
GRANT app_ro TO report_user;
```

SUPERUSER AUDIT & SET ROLE

```
SELECT rolname, rolsuper, rolcreatorole,
       rolcreatedb, rolbypassrls, rolcanlogin
FROM   pg_roles
WHERE  rolsuper OR rolcreatorole OR rolbypassrls;

-- Temporary privilege escalation (NOINHERIT user)
SET ROLE dba_admin;  -- do the work
RESET ROLE;
```

ROW-LEVEL SECURITY (RLS)

```
ALTER TABLE orders ENABLE ROW LEVEL SECURITY;
ALTER TABLE orders FORCE ROW LEVEL SECURITY;

CREATE POLICY tenant_isolation ON orders
    USING (tenant_id = current_setting('app.tenant_id')::int);
```

Warning: Table owners and BYPASSRLS roles bypass RLS. Use FORCE ROW LEVEL SECURITY for owners; the application should never connect as the table owner.

05

Auditing & Logging

RECOMMENDED LOG_LINE_PREFIX

```
log_line_prefix =
 '%t [%p]: [%l-1] user=%u,db=%d,app=%a,client=%h '
# %t time · %p PID · %l line no · %u user
# %d database · %a application · %h client IP
```

PGAUDIT — DETAILED AUDIT TRAIL

```
# postgresql.conf (restart required)
shared_preload_libraries = 'pgaudit'
pgaudit.log              = 'ddl, role, write'
pgaudit.log_catalog      = off
pgaudit.log_relation     = on
pgaudit.log_parameter    = off  # avoid leaking sensitive data
pgaudit.role             = 'auditor'
```

```
CREATE EXTENSION pgaudit;

-- Object-level audit: sensitive table
CREATE ROLE auditor NOLOGIN;
GRANT SELECT, UPDATE ON customers_pii TO auditor;

-- Extra audit for a specific user only
ALTER ROLE contractor SET pgaudit.log = 'read, write';
```

SAMPLE AUDIT LOG LINE

```
AUDIT: SESSION,1,1,DDL,DROP TABLE,TABLE,public.orders,
      DROP TABLE orders;,<not logged>
```

Warning: log_statement = 'all' and pgaudit.log = 'all' cause disk/I/O explosions in production and may write passwords to logs. Ship logs to a central SIEM.

06

Encryption

PGCRYPTO — COLUMN ENCRYPTION

```
CREATE EXTENSION IF NOT EXISTS pgcrypto;

-- Symmetric encryption (bytea column)
INSERT INTO customers (name, tckn_enc)
VALUES ('Ayşe', pgp_sym_encrypt('12345678901', :'key'));

SELECT pgp_sym_decrypt(tckn_enc, :'key') FROM customers;

-- Application password: bcrypt hash
SELECT crypt('s3cret', gen_salt('bf', 12));
```

PG_TDE — TRANSPARENT DATA ENCRYPTION

```
shared_preload_libraries = 'pg_tde'
CREATE EXTENSION pg_tde;
-- define key provider + principal key, then:
CREATE TABLE cards (...) USING tde_heap;
```

ENCRYPTED BACKUP

```
pg_dump -Fc app_db | gpg --encrypt \
  --recipient dba@corp.com > app_db.dump.gpg

gpg --decrypt app_db.dump.gpg | pg_restore -d app_db

# pgBackRest: repo1-cipher-type=aes-256-cbc
```

CERTIFICATE & CLIENT VERIFICATION

```
ALTER SYSTEM SET ssl_cert_file = 'server.crt';
ALTER SYSTEM SET ssl_key_file  = 'server.key';
SELECT pg_reload_conf();

# Client: full verification against MITM
psql "host=db.corp.com dbname=app_db user=app_user
     sslmode=verify-full sslrootcert=root.crt"
```

Warning: Don't embed the pgcrypto key in SQL — it can end up in logs via log_statement. sslmode=require does not verify the certificate; use verify-full. Community PostgreSQL has no built-in TDE.

07

Extension Security

Extension	Purpose	Risk
pgaudit	Audit trail	SAFE
pgcrypto	Cryptographic functions	SAFE
pg_stat_statements	Query statistics	SAFE
postgres_fdw / dblink	Remote connections — credential leakage	CAUTION
adminpack / file_fdw	Server file system access	HIGH
plpython3u / plperlU	Untrusted language — runs OS commands	HIGH

SHARED_PRELOAD_LIBRARIES WHITELIST

Approved libraries only — restart required

shared_preload_libraries =

'pg_stat_statements,pgaudit,passwordcheck'

Current value: SHOW shared_preload_libraries;

INSTALLATION & FUNCTION ACCESS

-- Privilege control: install extension into a dedicated schema

CREATE SCHEMA ext AUTHORIZATION postgres;

CREATE EXTENSION IF NOT EXISTS pgcrypto SCHEMA ext;

-- Functions are granted to PUBLIC by default!

REVOKE EXECUTE ON ALL FUNCTIONS IN SCHEMA ext FROM PUBLIC;

GRANT EXECUTE ON FUNCTION ext.pgp_sym_decrypt(bytea, text)

TO app_secure;

ALTER DEFAULT PRIVILEGES

REVOKE EXECUTE ON FUNCTIONS FROM PUBLIC;

-- Installed extension inventory

SELECT e.extname, e.extversion, n.nspname AS schema,

r.relname AS owner

FROM pg_extension e

JOIN pg_namespace n ON n.oid = e.extnamespace

JOIN pg_roles r ON r.oid = e.extowner;

SECURITY DEFINER FUNCTIONS

CREATE FUNCTION app.rotate_key() RETURNS void

LANGUAGE plpgsql SECURITY DEFINER

SET search_path = pg_catalog, pg_temp -- mandatory!

AS \$\$... \$\$;

Trusted extensions (PG13+): Extensions marked trusted = true can be installed by non-superusers with CREATE privilege on the database. For tighter control, enforce a whitelist with pgextwlist.

Warning: Untrusted languages (*u) and COPY ... PROGRAM let a superuser run OS-level commands — which is why superuser = the OS user postgres.

08

Security Checklist

GO-LIVE READINESS

Check Item	Priority
pg_hba.conf — trust removed pg_hba_file_rules WHERE auth_method='trust'	CRITICAL
scram-sha-256 active, no md5 hashes left SHOW password_encryption;	CRITICAL
SSL/TLS enabled, TLSv1.2+ enforced SHOW ssl; · pg_stat_ssl	CRITICAL
Superuser count ≤ 2 SELECT count(*) FROM pg_roles WHERE rolsuper;	CRITICAL
public schema restricted \dn+ public	IMPORTANT
pgaudit installed and active SHOW pgaudit.log;	IMPORTANT
Password policy enforced (passwordcheck + VALID UNTIL) SHOW shared_preload_libraries;	IMPORTANT
Backup encryption active gpg / repo1-cipher-type	IMPORTANT
RLS enabled on critical tables pg_class WHERE relrowsecurity	RECOMMENDED
log_connections on SHOW log_connections;	RECOMMENDED

CRITICAL

fix now

IMPORTANT

this sprint

RECOMMENDED

plan

ONE-QUERY SECURITY AUDIT

-- dbaguide.com · security quick audit

SELECT name, setting,

CASE

WHEN name = 'ssl' AND setting = 'on' THEN '✓'

WHEN name = 'password_encryption'

AND setting = 'scram-sha-256' THEN '✓'

WHEN name IN ('log_connections',

'log_disconnections')

AND setting = 'on' THEN '✓'

WHEN name = 'shared_preload_libraries'

AND setting LIKE '%pgaudit%' THEN '✓'

WHEN name = 'listen_addresses'

AND setting <> '*' THEN '✓'

ELSE 'x CHECK'

END AS status

FROM pg_settings

WHERE name IN ('ssl', 'password_encryption',

'log_connections', 'log_disconnections',

'shared_preload_libraries', 'listen_addresses');

-- Critical tables without RLS

SELECT relname FROM pg_class

WHERE relkind = 'r' AND NOT relrowsecurity

AND relnamespace = 'app'::regnamespace;

Best Practice: Run these checks weekly via CI/CD or cron and ship results to your monitoring system. Re-run the full list after every major upgrade.