SECURITY · POSTGRES

Sharing a database is sharing an attack surface

A row-level-security lesson from a build where my app shared one Postgres database with a third-party AI service.

Duarte Clemente · Power Platform & Azure consultant · Simple Smooth Safe

CONTEXT

The setup

My app and a self-hosted, third-party AI document service lived in the same Postgres (Supabase) project.

1. **Shared database**
   Both services read and write the same Postgres instance.

2. **Shared public key**
   In a Supabase project the anon API key is the same for everyone.

3. **Shared exposure**
   Which means the other service's tables are now part of my attack surface — whether I like it or not.

EXPOSURE

What the audit found

15/23

of the AI service's tables had row-level security switched off

The door was open

With RLS off and a shared public key, any authenticated browser could read those tables over the auto-generated REST API.

Privileged data

Those tables held document content — exactly what you don't want readable by default.

Not yet exploited

No breach — but “nobody has walked through it” is not the same as “the door is shut.”

REMEDIATION

The fix: deny by default

WHAT I DIDN'T DO

Fork the third-party service.

Patch its code (its license makes that painful).

Write bespoke policies for each table.

WHAT I DID

Enabled RLS on every exposed table.

Added zero policies — which means “nobody unless allowed.”

Matched what the service already did on its other tables.

TAKEAWAY

Shared database, shared responsibility.

If you integrate a tool that lives in your data layer, audit its RLS as if it were your own — because it is. How do you vet what touches your database?