BYPASSRLS live audit — bilko_admin, FORCE-RLS migracije (MC #106342, 2026-08-02)

MC #106342 — BYPASSRLS Live Audit — READ-ONLY

Agent: sec-106342-bypassrls (Securion) Date: 2026-08-02 Mode: STRICTLY READ-ONLY. Only `SELECT` statements were run against the live database. No INSERT/UPDATE/DELETE/DDL executed anywhere. Not marked done — see "What this does NOT settle" at the end.

---

0. PREMISE CORRECTION — there are not 3 Bilko databases, there is 1

The task asks to check `bilko-demo-pg`, `stage`, and `prod` as three databases. That framing is stale. Verified live and against the canonical infra doc:

- `~/business/ALAI-Holding-AS/products/Bilko/DEPLOY-MAP.md` (last verified 2026-07-15, MC #105793): "⚠ CANONICAL: Azure is live — GCP is dead (billing exhausted 2026-06-14)." Line 340: **"Environment classification: `bilko-demo` (bilko-web-demo + bilko-api-demo + bilko-demo-pg) is customer-facing PRODUCTION. Stage (`bilko-*-stage`) is CI/E2E."** Line 135 / OCD-3: stage and demo **share the same `bilko-demo-pg` instance** (no isolation), multi-tenant by `organizations.country` only.
- Live tool-verified, this session, `alai-cli-deployer` SP (Contributor @ subscription `5b0b4d9b-e677-464e-abf0-5170cbce3b8e`):

```
az postgres flexible-server list --query "[].{name:name, rg:resourceGroup,
state:state, fqdn:fullyQualifiedDomainName}" -o table
```

```
Name                Rg                  State    Fqdn
------------------  ------------------  -------
-----------------------------------------
lumiscare-demo-db   rg-lumiscare-demo   Stopped  lumiscare-demo-
db.postgres.database.azure.com
plock-staging-db    plock-staging-rg    Ready    plock-staging-
db.postgres.database.azure.com
pg-alai-control     rg-alai-control     Ready    pg-alai-
control.postgres.database.azure.com
bilko-demo-pg       rg-bilko-demo       Ready    bilko-demo-
pg.postgres.database.azure.com
qody-demo-db        rg-qody-demo        Ready    qody-demo-
db.postgres.database.azure.com
qody-prod-db        rg-qody-prod        Ready    qody-prod-
db.postgres.database.azure.com
```

`bilko-demo-pg` **is the only Postgres server anywhere in the subscription with "bilko" in its name.** No separate stage or prod server exists. The historical GCP `ENV-MATRIX.md` (`infrastructure/gcp/ENV-MATRIX.md`, "standardized 2026-05-21") describing separate `bilko-staging-db` / `bilko-demo-db` / unwired-`prod` is **decommissioned** — GCP project `tribal-sign-487920-k0` billing was cut 2026-06-14 (DEPLOY-MAP.md §"HISTORICAL: GCP Cloud Run / Cloud SQL (dead 2026-06-14)"). Do not use that doc for current topology.

Consequence: the DECIDING FACT query only needs to run once — against `bilko-demo-pg`, database `bilko` — because that single instance IS "stage" and IS "prod" (demo). There is no third database to independently check. I did not check a separate prod because none exists.

---

1. DECIDING FACT — bilko_admin BYPASSRLS, live, raw output

Access chain (read-only, all tool-verified this session, no secrets printed beyond what's needed for this record):

- `node ~/system/tools/vault.js get "Bilko Prod DB Credentials"` → GCP-era secret, **not used** (wrong topology, confirmed stale by DEPLOY-MAP.md).
- Current default `az` login (unnamed SP, appid `1a0b3018-...`) had **zero** authorization on `rg-bilko-demo` (`AuthorizationFailed` on `postgres flexible-server show`, KV secret read, and `containerapp secret show`). Not usable.

- `node ~/system/tools/vault.js get "Azure Service Principal — alai-cli-deployer"` → Contributor at subscription scope `5b0b4d9b-e677-464e-abf0-5170cbce3b8e` (exact scope of `rg-bilko-demo`). Logged in via an **isolated** `AZURE_CONFIG_DIR` (scratchpad-local) so this session's Azure CLI state never touched the shared `~/.azure` profile other concurrent agents may depend on. Login succeeded — secret not yet rotated past its 2026-07-26 `rotation_due` note.
- `az containerapp secret show -g rg-bilko-demo -n bilko-api-demo --secret-name db-password --query value -o tsv` → live `bilko_admin` password (per DEPLOY-MAP.md:20/137, this IS the Flyway/DB admin password, ACA secret `db-password` = KV `db-admin-password`).

Deciding-fact query, run against `bilko-demo-pg.postgres.database.azure.com`, db `bilko`, as `bilko_admin` (same identity `FLYWAY_DB_USER` uses per `azure-pipelines.yml:88`):

```
SELECT rolname, rolsuper, rolbypassrls, rolcanlogin, rolinherit FROM pg_roles WHERE
rolname='bilko_admin';
```

```
   rolname   | rolsuper | rolbypassrls | rolcanlogin | rolinherit
-------------+----------+--------------+-------------+------------
 bilko_admin | f        | t            | t           | t
(1 row)
```

VERDICT: bilko_admin DOES have BYPASSRLS = true, live, today, on the only Bilko database that exists. `rolsuper = f` (matches V124/V132's own description: "BYPASSRLS but NOT superuser"). The task's central fear — that BYPASSRLS is missing and every bare backfill silently no-op'd — is **REFUTED for the current live environment.**

Why: it's Azure's admin-login grant, not anything in the repo

```
SELECT r.rolname AS member, m.rolname AS of_role FROM pg_auth_members am
JOIN pg_roles r ON r.oid=am.member JOIN pg_roles m ON m.oid=am.roleid WHERE
r.rolname='bilko_admin';


   member    |       of_role
-------------+----------------------
 bilko_admin | pg_read_all_settings
 bilko_admin | pg_read_all_stats
 bilko_admin | pg_stat_scan_tables
 bilko_admin | azure_pg_admin
 bilko_admin | bilko_app
 bilko_admin | bilko_admin (x2, dup rows from bilko_app + bilko_purge_worker grants)
```

Full role dump (`SELECT rolname, rolsuper, rolbypassrls, rolcreaterole, rolcreatedb FROM pg_roles ORDER BY rolname`) shows only two roles server-wide with `rolbypassrls=t`: `azuresu` (Azure's true PG superuser, `rolsuper=t`, control-plane only) and `bilko_admin` (`rolsuper=f`, member of `azure_pg_admin`). `bilko_admin` is a member of `pg_read_all_stats` / `pg_read_all_settings` / `pg_stat_scan_tables` / `azure_pg_admin` — this is exactly Azure Postgres Flexible Server's **administrator login** role shape, provisioned by the platform at server-create time, independent of any SQL Flyway ever ran.

Cross-checked against the migration source that the task and prior verdicts cite:

- `V30_1__ensure_bilko_admin_role.sql` / `V32__fix_v31_bilko_admin_role.sql`: both do `CREATE ROLE bilko_admin BYPASSRLS NOINHERIT` **inside** `IF NOT EXISTS (SELECT 1 FROM pg_roles WHERE rolname='bilko_admin')` — confirmed this branch never fires on Azure (role pre-exists as the server admin login before Flyway ever runs V30_1).
- `grep -rn "ALTER ROLE bilko_admin" *.sql` across all 149 migrations on `azdo/main`: **zero hits.** No migration ever sets BYPASSRLS on Azure.
- V30_1's own comment: "*BYPASSRLS is required only for SECURITY DEFINER auth lookup functions... NOINHERIT prevents role members from automatically inheriting BYPASSRLS.*" — this describes the **GCP** topology, where the real Flyway/API connecting user was `bilko` (a *member* of the `bilko_admin` role, not `bilko_admin` itself — V32's own comment: "*bilko-demo-db Cloud SQL... only created 'bilko' user; bilko_admin was never provisioned*"). Under `NOINHERIT`, mere membership does **not** auto-grant BYPASSRLS — the connecting role would need an explicit `SET ROLE bilko_admin` to actually get it. **This is why the witness's fear (`witness-verdict.md` CASE A: owner, FORCE RLS, `rolbypassrls=f` → `UPDATE 0`) was a real and correctly-identified risk for the GCP topology.** It does not describe Azure, where the connecting identity **is** `bilko_admin` directly and carries `BYPASSRLS` as its own role attribute, granted by the Azure control plane at server provisioning — not by `NOINHERIT` membership, not by any migration.

So there are, and have been for some time, two different things sharing the name "bilko_admin" across the product's history: a GCP-era non-login `BYPASSRLS NOINHERIT` role created by Flyway (V30_1/V32) for object ownership only, and the Azure-era LOGIN administrator account that Flyway now connects as. They coincidentally share a name; only the second is live today, and it happens to satisfy the assumption the first was never able to guarantee.

---

2. Migrations with DML on FORCE-RLS tables — enumerated, and did they actually write rows?

Live FORCE RLS table count, `public` schema, `bilko-demo-pg`/`bilko`:

```
SELECT count(*) FROM pg_class c JOIN pg_namespace n ON n.oid=c.relnamespace
WHERE c.relforcerowsecurity = true AND n.nspname='public';
--> 47   (matches the task's documented count exactly; all 47 owned by bilko_admin)
```

Enumeration method: grepped every `.sql` file in `apps/api/src/main/resources/db/migration/` for a top-level (any indentation) `INSERT INTO|UPDATE|DELETE FROM` statement naming one of the 47 live FORCE-RLS tables. Ran against the **local working-tree checkout** (146 files — this repo's shared checkout is currently on an unrelated branch, `fix/chatbot-per-user-ratelimit-105463-v2`, left by another agent; I did not `git checkout` it to avoid disrupting concurrent work) **plus** a `git ls-tree`/`git show` diff against the cached `azdo/main` ref (149 files) to catch anything only present on main. Diff found exactly 3 files present on main but not the local checkout: `V147`, `V148`, `V149` — all three individually reviewed below. (`git fetch azdo main` itself failed live — `fatal: could not read Username... Device not configured` — so `azdo/main` here is the last-cached ref, `e81db44a`, 2026-07-29 12:00:34 UTC per `git log -g azdo/main`. This matches the same caveat the 106313 witness hit; re-fetch with an authenticated session before trusting this as "current main" for anything beyond this audit.)

Migration	Target FORCE-RLS table(s)	Guard pattern	Verdict
V39 `fix_hr_demo_seed_rls_privileges`	`users` (`organizations` UPDATE not FORCE-RLS)	`GRANT ... ; SET ROLE bilko_admin; ... ;` (RESET ROLE not shown in grep window but pattern matches V61)	SAFE — explicit role switch
V61 `hr_demo_admin_user`	`users`	`GRANT ...; SET ROLE bilko_admin;` before INSERT	SAFE
V62 `hr_demo_chart_of_accounts`	`accounts`	`GRANT...; SET ROLE bilko_admin;` (line 67) ... `RESET ROLE` (line 168) wraps the INSERT	SAFE
V66 `entra_rls_and_password_nullable`	`entra_external_identities` INSERT is **inside a `CREATE FUNCTION ... SECURITY DEFINER AS $$ ... $$` body**	N/A — not migration-time DML, it's runtime application code (already reviewed for RLS-safety in the 106313 witness review, CLAIM 3)	NOT IN SCOPE for this lint class — false positive from the grep, correctly excluded
V97 `hr_rrif_chart_of_accounts`	`accounts`	`GRANT...; SET ROLE bilko_admin;` (line 57) ... `RESET ROLE` (line 256)	SAFE
V111 `demo_gl_backfill_sales_invoices`	`accounts`, `transactions`, `invoices` (UPDATE), `expenses`	`GRANT...; SET ROLE bilko_admin;` (line 162) ... `RESET ROLE` (line 481)	SAFE

Migration	Target FORCE-RLS table(s)	Guard pattern	Verdict
V144 `seed_ci_tenant_chart_of_accounts`	`accounts`	**NONE.** No `SET ROLE`, no `GRANT`, no capability check, no RLS-awareness comment anywhere in the file. Bare top-level `INSERT INTO public.accounts (...)`.	**UNGUARDED — succeeded only because bilko_admin happens to carry BYPASSRLS today. Would have silently no-op'd exactly like BUG-005/V65 on any environment where it didn't.**
V147 `seed_e2e_admin_user`	`users` (UPDATE, existing-user branch, line 59-63), `entra_external_identities` (INSERT), `organizations` (not FORCE-RLS)	Comment only: "*Bypass RLS via direct insert (migration runs as the migration superuser role, not as bilko_app — RLS does not apply here, same as V71)*" — an **assumption stated in a comment, not a runtime check.** The `users` INSERT (line 104) is preceded by `SET LOCAL app.current_org_id` (line 102) which would satisfy the org_isolation policy independent of BYPASSRLS, but the `users` UPDATE at line 59 (promote-existing-user branch) has **no such fallback** and depends entirely on the unverified BYPASSRLS assumption.	**PARTIALLY UNGUARDED — the UPDATE branch has zero runtime verification of its own premise.**
V148 `jit_org_creator_owner` (revised, merged 2026-07-26, MC #106313)	`users` (UPDATE, via `bilko_auth.backfill_jit_orphan_owners()` SECURITY DEFINER function)	**Runtime capability check**: `SELECT rolsuper OR rolbypassrls INTO v_may_see_all_rows FROM pg_roles WHERE rolname = current_user` → `RAISE EXCEPTION` if false, **plus** a post-condition assertion (`v_remaining > 0` after the UPDATE → `RAISE EXCEPTION`) so a partial/silent failure can't hide behind a merely-nonzero update count either	**SAFEST pattern in the repo — this is the model to lint toward**
V149 `flag_payslips_with_abolished_prirez` (MC #106302)	`payslips` (UPDATE)	3-branch runtime capability check (superuser/BYPASSRLS → direct; owner-only → `NO FORCE` / `UPDATE` /restore `FORCE` +assert; neither → `RAISE EXCEPTION`)	SAFE — most defensive pattern, correct on any topology

Did the ones that actually ran write real rows? Verified live, not from Flyway's flag

```
-- flyway_schema_history for every migration above, all read success=t (not trusted at face
value):
 version |             description             | success |        installed_on
---------+-------------------------------------+---------+----------------------------
 39      | fix hr demo seed rls privileges     | t       | 2026-06-15 00:59:39.09406
 61      | hr demo admin user                  | t       | 2026-06-15 01:00:14.300477
 62      | hr demo chart of accounts           | t       | 2026-06-15 01:00:15.217859
 66      | entra rls and password nullable     | t       | 2026-06-15 01:04:30.70011
 97      | hr rrif chart of accounts           | t       | 2026-06-21 09:33:57.300879
 111     | demo gl backfill sales invoices     | t       | 2026-07-11 20:34:55.346603
 144     | seed ci tenant chart of accounts    | t       | 2026-07-23 13:43:48.231407
 148     | jit org creator owner               | t       | 2026-07-26 23:05:04.874536
 149     | flag payslips with abolished prirez | t       | 2026-07-26 23:05:05.452968
```

Direct data verification, independent of the Flyway flag:

- **V144**: `SELECT count(*) FROM accounts WHERE organization_id = '2e852173-170e-5f18-adf9-253ce4922444'` (the CI tenant org id named in V144's own header) → **5 rows.** Real data, not a silent zero. It worked — but by luck, not by design.
- **V111**: `SELECT count(*) FROM transactions` → **278 rows** (non-zero; V111's whole point was that this table was empty before it ran).
- **V148**: `SELECT count(*) FROM bilko_auth.jit_orphan_owner_candidates()` (the function's own idempotent re-check — should be 0 if the backfill fully closed the gap it targets) → **0.** `SELECT role, count(*) FROM users GROUP BY role` → `owner: 9, viewer: 4, accountant: 3, admin: 1` (17 total) — owners exist, not stuck at pre-fix state.
- **V149**: `SELECT count(*) FILTER (WHERE calc_json ? 'prirez_review_required') AS flagged, count(*) AS total FROM payslips` → `flagged: 0, total: 8`. This is a **legitimate zero**, not a silent-failure zero: V149 only flags payslips with `period >= 2024-01` AND non-zero `prirezCents`, and the current 8-row demo dataset has none matching. Distinguishable from BUG-005 because V149's own code took the proven-correct capability-check branch (confirmed by V148/V149 both running successfully with their fail-loud guards intact — if BYPASSRLS/owner had been absent, the migration would have raised an exception and Flyway would show `success=f`, not `t`).

Conclusion for scope items 2-3: no migration in this repo has silently no-op'd on the live database. Every bare/weakly-guarded migration found (V144, and the UPDATE branch of V147) happened to write real data anyway, because the platform-level BYPASSRLS grant was present the whole time it ran. **This is not evidence the pattern is safe — it is evidence the repo has been gambling on an unverified, unmonitored, un-asserted environmental fact and has**

not yet lost.

---

3. The live risk is forward-looking, not the past-incident panic the task was framed around

`DEPLOY-MAP.md` OCD-5 (existing, pre-dating this audit): "*Flyway user `bilko_admin` password read at CI runtime from live ACA secret (no separate CI secret)... CI pipeline has production DB write access... Mitigation: Create dedicated `flyway_ci` role with DDL-only grants, rotate `bilko_admin` out of CI.*"

If OCD-5 is ever implemented as currently scoped ("DDL-only grants"), every migration in the table above marked SAFE-via-`SET ROLE bilko_admin`, UNGUARDED, or PARTIALLY UNGUARDED breaks identically to BUG-005 — silently — because a DDL-only role has no reason to carry BYPASSRLS, and nothing in the repo asserts that it must. Only V148/V149's pattern (runtime capability check + fail-loud) would survive that rotation without a silent regression; everything relying on `SET ROLE bilko_admin` would additionally need bilko_admin itself to still exist and still carry BYPASSRLS as a *role that can be switched into*, which the DDL-only proposal doesn't guarantee either.

This audit does not recommend blocking OCD-5 — it recommends that **whoever picks up OCD-5 reads this file first**, and that the CI lint below ships *before* OCD-5, not after.

---

4. CI lint design (proposed, NOT implemented — read-only task)

Goal: fail CI on any migration that writes to a FORCE-RLS table without a provable RLS-safety strategy, so this stops being decided by luck.

Where: new `CI_Gates` job in `azure-pipelines.yml`, alongside the existing Gitleaks/Semgrep/ Trivy jobs — same pool (`vmImage: ubuntu-latest`), so it runs on every PR before merge, not just on push to main (catches the problem before Flyway ever touches a real database, unlike the current Flyway_Migrate stage which is the first and only stage that would ever exercise this).

Script: `scripts/ci/rls-migration-lint.{js|py}`, invoked only against migration files changed in the PR diff (`git diff --name-only origin/main...HEAD -- apps/api/.../db/migration/`), so merged history is never re-litigated.

Table manifest, not hardcoded: check in `apps/api/.../db/migration/.force-rls-tables.json` — a generated list of the 47 (currently) FORCE-RLS table names, refreshed by a small script that runs `SELECT relname FROM pg_class ... WHERE relforcerowsecurity` against a scratch/CI Postgres built from the full migration history (Testcontainers, matching the existing `integrationTest` setup pattern) — **not** by hand-maintaining the list, which will drift the moment a new FORCE RLS table is added and nobody remembers to update a second file.

Detection logic per changed migration file:

1. Parse the file into top-level statements, correctly tracking `$$...$$` (and `$tag$...$tag$`) dollar-quote depth so nothing inside a quoted block is treated as top-level SQL.
2. Classify each dollar-quoted block: a block immediately preceded by `CREATE [OR REPLACE] FUNCTION ... AS` defines **runtime** code (excluded from this lint — application-time RLS safety is a different, already-existing review surface, per the 106313 witness's CLAIM 3 methodology). A bare `DO $$ ... $$` block executes **at migration time** and its contents are in scope exactly like top-level statements.
3. For every `INSERT INTO|UPDATE|DELETE FROM` (top-level or inside a migration-time `DO` block) targeting a table in the FORCE-RLS manifest, require **one** of:
   - a. An enclosing/preceding `SET ROLE <role>` in the same file with a later `RESET ROLE` bracketing the statement, where `<role>` is a role documented (in the same PR or an existing comment) to hold BYPASSRLS or ownership — OR
   - b. A preceding runtime check in the same `DO` block matching the V148/V149 shape: a `SELECT ... rolsuper OR rolbypassrls ... FROM pg_roles WHERE rolname = current_user` (or equivalent) followed by a `RAISE EXCEPTION` on the negative branch, before the DML — OR
   - c. A preceding `SET LOCAL app.current_org_id` in the same transaction/block that would satisfy the table's specific RLS policy (requires the manifest to also carry each table's policy predicate column, or a simpler conservative version: just require (a) or (b) always, and treat (c) as an allowed exception only for a short, explicitly reviewed allow-list).
4. Violation → CI fails with: file, line, table name, and a link to this evidence file plus the V148/V149 pattern as the reference implementation.

Companion runtime guard (recommended alongside the lint, cheap and closes the OCD-5 blind spot directly): a post-`Flyway_Migrate` step in the pipeline that runs exactly the deciding-fact query from §1 and fails the stage if `rolbypassrls` is ever `false` for `FLYWAY_DB_USER` — so a future credential rotation (OCD-5 or otherwise) is caught at the infrastructure layer the moment it happens, independent of whether every past migration was linted correctly.

What this does NOT settle

- Not marked done. This is READ-ONLY audit evidence for the orchestrator/team-lead to act on.

- `azdo/main` here is a cached ref (`e81db44a`, 2026-07-29 12:00:34 UTC) — live `git fetch` failed in this environment (device-code auth not available headless). Re-verify against a freshly authenticated fetch before treating "3 files newer than local checkout" as exhaustive.
- I did not attempt to check GCP for a live counter-example (billing is dead, confirmed via `DEPLOY-MAP.md`; did not independently re-verify GCP is unreachable — out of scope for a Bilko DB audit and would require separate GCP credentials this task didn't ask for).
- I have not implemented the lint or the companion runtime guard — design only, per scope.
- The `alai-cli-deployer` SP secret used here is past its own documented `rotation_due: 2026-07-26` (today is 2026-08-02) and still worked; flagging for whoever owns secret rotation, not fixing it here (out of scope, not requested).

Revision #2
Created 2026-08-02 10:59:26 UTC by John
Updated 2026-08-10 07:37:58 UTC by John