SEC-1c-3c Plan-Review

Legacy SUPERUSER Lockdown — read-only Analyse

Datum: 2026-05-14 (UTC) | Master HEAD: `d351a3a` | Vorgänger: SEC-1c-3a/b/FU1/FU2 (alle closed)

PLAN-REVIEW **READ-ONLY** **NO-GO sofort** **GO Variante D nach Stabilitätsfenster**

> **Kurzfassung.** Alle drei Services (GUI/Bot/Worker) verbinden sich verifiziert als `tradingbot_gui_app`. `tradingbot_gui` ist faktisch raus aus dem Service-Pfad. Aber: **vier Pre-Conditions blockieren einen sofortigen Lockdown** — Backup-Script-Default, docker-compose-Default-Fallback, Worker-Watchdog-Default, und das Migrator-Owner-Problem für künftige Schema-Migrationen. Empfehlung: **Variante D** — 7 Tage Stabilitätsfenster, Pre-Condition-Fixes parallel, danach `NOSUPERUSER NOCREATEROLE NOCREATEDB NOREPLICATION NOBYPASSRLS` als atomic Block.

1. Boundary-Snapshot

Item	Wert
master HEAD	`d351a3a` (unverändert)
git status	clean
Bot main.py PID (in-container)	195
Bot Container Host-PID	2657896 — running healthy
Worker Container Host-PID	2658058 — running unhealthy (FP)
GUI Container Host-PID	2656633 — running
BINANCE_TESTNET	true
cmd 13 / managed_proposals / history	cancelled / 0 / 0
Worker Heartbeat	3 frische Heartbeats, age 25-86s
Bot Tracebacks (last 300 lines)	0
Service current_user	GUI/Bot/Worker alle als `tradingbot_gui_app`
pg_stat_activity tradingbot_gui	nur meine eigene psql-Recon-Session
SECRET-READ-ONCE (FU2 + alte Files)	alle shredded ✓

2. DB-Rollen — IST-Stand

Rolle	super	createrole	createdb	login	replication	bypassrls
`tradingbot_gui`	t	t	t	t	t	t
`tradingbot_gui_app`	f	f	f	t	f	f
`tradingbot_gui_migrator`	f	f	f	t	f	f

3. Ownership-Inventory

Objekt	Anzahl	Owner
Database `tradingbot_gui`	1	`tradingbot_gui`
Schema `public`	1	`pg_database_owner` (PG16-default → implizit der DB-Owner)
Tables (public)	24	`tradingbot_gui`
Sequences (public)	18	`tradingbot_gui`
Types/Domains (public)	24	`tradingbot_gui` (implizite row-types)
Functions/Procedures (public)	0	—
Views (public)	0	—
Extensions	1 (`plpgsql`)	`tradingbot_gui` — keine SUPERUSER-Extension

ALTER DEFAULT PRIVILEGES (komplett sauber)

Issuer (creates → gets DML for app)	obj_type	Default-Grant
`tradingbot_gui`	relations	`tradingbot_gui_app=arwd/tradingbot_gui`
`tradingbot_gui`	sequences	`tradingbot_gui_app=rU/tradingbot_gui`
`tradingbot_gui`	functions	`tradingbot_gui_app=X/tradingbot_gui`
`tradingbot_gui_migrator`	relations	`tradingbot_gui_app=arwd/...`

`tradingbot_gui_migrator`	sequences	`tradingbot_gui_app=rU/...`
`tradingbot_gui_migrator`	functions	`tradingbot_gui_app=X/...`

Bedeutet: zukünftige Objekte, egal ob von `tradingbot_gui` oder `tradingbot_gui_migrator` angelegt, bekommen automatisch DML-Grants für `tradingbot_gui_app`. Wartungsfrei.

4. Migrator-Capabilities — kritischer Befund

Live-Test über `SET ROLE tradingbot_gui_migrator` in einer Transaktion (mit Rollback):

Operation	Ergebnis
`CREATE TABLE _migrator_test_…`	**OK** (Schema-Privilege CREATE)
`ALTER TABLE decision_logs ADD COLUMN …`	**ERROR: must be owner of table decision_logs**
`SELECT FROM decision_logs`	**OK** (via SEC-1c-3a grant)
`INSERT INTO decision_logs`	**OK** (via SEC-1c-3a grant)

> **BEFUND #1 (CRITICAL):** Der Migrator kann **keine ALTER/DROP-Operationen auf bestehende 24 Tabellen**. Owner ist `tradingbot_gui`. Solange das so bleibt, sind *echte* Schema-Migrationen (z.B. neue Column an bestehender Tabelle, DROP COLUMN, RENAME) **nur als `tradingbot_gui` möglich** — also Lockdown läuft ins Leere, sobald die nächste Migration kommt.

Drei Lösungspfade:

1. **Ownership-Migration**: `ALTER TABLE … OWNER TO tradingbot_gui_migrator` für alle 24 Tabellen + 18 Sequences. Saubere Trennung, aber bricht `db_backup.sh` wenn der als `tradingbot_gui` läuft (siehe §5).
2. **Membership-Grant**: `GRANT tradingbot_gui TO tradingbot_gui_migrator` — Migrator wird Mitglied der Owner-Rolle und kann via `SET ROLE` Owner-Privs nutzen. Pragmatisch aber umgeht den Zweck der Rollentrennung.
3. **Migrator-bleibt-readonly + Schema-Changes via `tradingbot_gui`**: Lockdown nur NOSUPERUSER (kein NOCREATE-Privs), `tradingbot_gui` bleibt Schema-Migrator. Mininalrisiko aber Migrator-Rolle effektiv obsolet.

5. `tradingbot_gui`-Referenzen außerhalb der DB

Datei	Zeile	Kontext	Lockdown-Impact
`ops/db_backup.sh`	31	`DB_USER="${DB_USER:-tradingbot_gui}"` — Default-Fallback in pg_dump-Script	OK nach Lockdown (Owner darf pg_dump). Aber *identitätsmäßig* bleibt SUPERUSER-Rolle Backup-User → fühlt sich „nicht ganz raus" an.
`worker_watchdog.sh`	69-70	`GUI_DB_USER="${GUI_DB_USER:-tradingbot_gui}"` für Heartbeat-Read	OK nach Lockdown (LOGIN-Privs bleiben, SELECT auf `bot_statuses` via Owner-Privs OK). Läuft via `docker exec gui-db psql` = container-internes UDS-Socket, kein PW-Auth nötig.
`docker-compose.yml`	126-128	`GUI_DB_USER=${GUI_DB_USER:-tradingbot_gui}` + `GUI_DB_NAME` + `GUI_DB_PASSWORD` als **Compose-Defaults für clawbot-worker**	**Lock-Down-Risiko**: wenn jemand Root-`.env` aufräumt und Variablen entfernt, würde Worker-Container mit dem *String* `tradingbot_gui` als Passwort versuchen zu connecten → Auth-Fail. Aktuell unkritisch weil Root-.env die Vars setzt, aber latent.
`cron.d/steve-tradingbot-backup`	—	nutzt `db_backup.sh` mit Defaults	Erbt Punkt 1.
`gui/.env`	18	`DB_DATABASE=tradingbot_gui`	DB-Name, nicht User. Irrelevant für Lockdown.
Code-Files (db_emitter, command_worker, Tests)	—	20+ Treffer aber meist DB-Name, Test-Defaults oder Migrator-Verweise	Nicht im SEC-1c-3c-Scope.

6. Risikoanalyse — Was bricht bei `NOSUPERUSER`?

Bereich	Risiko	Begründung
Bot DML (decision_logs, etc.)	**kein Risiko**	Bot läuft als `tradingbot_gui_app`, nicht als `tradingbot_gui`.
Worker bot_statuses INSERT	**kein Risiko**	Worker läuft als `tradingbot_gui_app`.
GUI Filament Login/Read/Write	**kein Risiko**	GUI läuft als `tradingbot_gui_app`.
Daily pg_dump (02:00 UTC)	**kein Risiko**	Owner darf pg_dump. `tradingbot_gui` bleibt Owner aller Objects auch ohne SUPERUSER.
Worker-Watchdog Heartbeat-Read	**kein Risiko**	SELECT auf bot_statuses via Owner-Privs.
Laravel Migrationen ALTER auf bestehende Tabellen	**HOCH**	Migrator-Rolle ist nicht Owner — kann nicht ALTER. Bei nächster Schema-Migration (z.B. ADD COLUMN) → Migration-Fail. Solution: §4 oben.

Extension-Installation (CREATE EXTENSION)	**mittelfristig**	Aktuell nur plpgsql installiert. Wenn man irgendwann `pg_trgm` / `uuid-ossp` braucht → SUPERUSER nötig. Kein akutes Problem.
ALTER SYSTEM (z.B. `shared_buffers` tunen)	**mittelfristig**	Nur DBA-Aufgabe, selten. Im Notfall via SSH+Postgres-Superuser-Default (`postgres` -Role im Container-Init, falls vorhanden) oder via direktem `postgresql.conf` -Edit + Restart.
pg_restore in leere DB (Disaster Recovery)	**mittel**	pg_restore braucht Schema-Recreate-Rechte. Wenn `tradingbot_gui` DB-Owner bleibt: OK. Bei kompletter DB-Neuanlage: man braucht entweder gespeicherten Container-Init-SUPERUSER (`postgres`) oder ein Manual-CREATE-DATABASE-Step.
BYPASSRLS verlust	**irrelevant**	Keine RLS-Policies aktiv im Schema. Laravel-Standard.
REPLICATION verlust	**irrelevant**	Kein Replication-Setup (1-Node-Postgres).
CREATEROLE/CREATEDB verlust	**irrelevant**	Im Notfall via separater DBA-Rolle oder direkter `postgres` -User im Container.

7. Optionen A–D bewertet

Option	Inhalt	Pro	Contra	Empfehlung
A	`tradingbot_gui` bleibt SUPERUSER, kein Service nutzt ihn aktiv	Maximale Flexibilität, Migrator-Problem irrelevant	Lockdown-Versprechen unerfüllt — der Vektor aus dem Mai-Incident ist weiter offen	**NEIN**
B	`NOSUPERUSER` , bleibt DB-Owner + Tables-Owner	Saubere SUPERUSER-Entkopplung, Backups bleiben funktional, GUI/Bot/Worker laufen weiter	Migrator-ALTER-Problem (siehe §4 BEFUND #1) bleibt offen → nächste echte Migration broken	**JA, aber mit Caveat**
C	Neuer separater Break-Glass-DBA-User, danach `tradingbot_gui` komplett raus	Cleanest possible architecture	Größere Umbauphase: DBA-Rolle anlegen, Backup-Script anpassen, pg_dump-User wechseln, Ownership wandern, Compose-Defaults fixen — ~4-6h Implementierung + Tests	**FUTURE (SEC-1c-3d)**
D	7 Tage Stabilitätsfenster + Pre-Condition-Fixes, danach **Option B**	Defensiv, validiert dass keine vergessene `tradingbot_gui` -Connection lebt, Pre-Conditions sauber gefixt	Zeitverzögerung; aber kein operatives Risiko	**JA (empfohlen)**

8. Pre-Conditions vor Lockdown (Option D)

1. **PC-1 — `db_backup.sh` DB_USER-Default analysieren.** Aktuell `${DB_USER:-tradingbot_gui}` . Nach Lockdown bleibt das funktional (Owner darf pg_dump), aber: explizit zu `tradingbot_gui` halten oder zu eigenständigem `tradingbot_gui_backup` wechseln? Operator-Decision.
2. **PC-2 — docker-compose-Defaults harmlos machen.** Lines 126-128: aktuell Fallback auf `tradingbot_gui` . Empfehlung: leere Defaults oder `tradingbot_gui_app` als Default. Erfordert Bot/Worker-Recreate.
3. **PC-3 — Worker-Watchdog DB_USER-Default.** Analog PC-2. Watchdog-Script line 69-70. Erfordert host-side script edit (kein Restart nötig).
4. **PC-4 — Migrator-Owner-Problem klären.** Operator-Decision zwischen: (a) Ownership-Migration ALLER bestehenden Tables zum Migrator, (b) Membership-Grant Migrator → tradingbot_gui, (c) `tradingbot_gui` bleibt für Schema-Migrations zuständig (lockdown nur NOSUPERUSER, nicht weitere Privs-Entzug). Wenn aktuell keine Schema-Migration in den nächsten 7 Tagen ansteht: deferred-to-SEC-1c-3d.
5. **PC-5 — 7 Tage Beobachtung**: tägliche Prüfung von `pg_stat_activity` auf `tradingbot_gui` -Connections (außer DBA-Manual via `docker exec psql`). Wenn jemals eine Service-Connection als `tradingbot_gui` auftaucht: Pre-Condition unerfüllt, Lockdown verschieben.

9. Empfohlener SQL-Block für Lockdown (NACH Stabilitätsfenster)

```
BEGIN;
-- Soft-Lockdown: nur SUPERUSER + BYPASSRLS + REPLICATION raus.
-- CREATEROLE/CREATEDB bleibt erlaubt (für eventuelle DBA-Notwendigkeiten via SSH).
ALTER ROLE tradingbot_gui NOSUPERUSER;
ALTER ROLE tradingbot_gui NOBYPASSRLS;
ALTER ROLE tradingbot_gui NOREPLICATION;
-- Optional (Hard-Lockdown):
-- ALTER ROLE tradingbot_gui NOCREATEROLE;
-- ALTER ROLE tradingbot_gui NOCREATEDB;

-- Verify:
SELECT rolname, rolsuper, rolcreaterole, rolcreatedb, rolreplication, rolbypassrls
FROM pg_roles WHERE rolname='tradingbot_gui';
-- Erwartung: rolsuper=f, rolreplication=f, rolbypassrls=f
COMMIT;
```

Begründung „Soft-Lockdown empfohlen":

- CREATEROLE/CREATEDB sind für Notfall-DBA-Operationen via SSH+psql nützlich.
- Sie sind nicht direkt im SUPERUSER-Vektor enthalten (z.B. ALTER SYSTEM erfordert SUPERUSER, nicht CREATEROLE).
- Verlust dieser zwei Privs wäre kein Sicherheitsgewinn aber operativer Verlust.

10. Rollback-Plan

```
-- Single-step rollback (revoke 3 ALTERs):
```

```
ALTER ROLE tradingbot_gui SUPERUSER;
ALTER ROLE tradingbot_gui BYPASSRLS;
ALTER ROLE tradingbot_gui REPLICATION;

-- Verify:
SELECT rolname, rolsuper FROM pg_roles WHERE rolname='tradingbot_gui';
-- Erwartung: rolsuper=t
```

Rollback-Trigger:

- Daily pg_dump (02:00 UTC) schlägt fehl → SUPERUSER zurück
- Worker-Watchdog Heartbeat-Read schlägt fehl → SUPERUSER zurück
- GUI/Bot/Worker reconnect-fail → unwahrscheinlich (sie sind App-User), aber falls doch → SUPERUSER zurück
- Operator manuelle DBA-Operation schlägt unerwartet fehl → SUPERUSER zurück

11. Stop-Regeln

- Während 7-Tage-Fenster: jede `tradingbot_gui`-Service-Connection (außer manueller DBA) → STOP, Investigation
- Pre-Condition PC-2 (compose-Defaults) nicht gefixt → STOP, Lockdown nicht starten
- Während Lockdown-Block: ALTER-Operation hängt > 30s → STOP, Investigation auf locked transactions
- Nach Lockdown: erste pg_dump-Lauf um 02:00 UTC NICHT erfolgreich → automatic Rollback (per cron-Alarm) bzw. Operator-Trigger

12. GO/NO-GO

NO-GO sofortiger Lockdown. Vier Pre-Conditions sind unklar oder offen:

1. Migrator kann keine ALTER auf bestehende Tabellen (Befund #1) — bei nächster Schema-Migration broken.
2. Compose-Default-Fallback (lines 126-128) ist latentes Risiko.
3. Backup-Script-Default unverändert.
4. Worker-Watchdog-Default unverändert (selbes Pattern).

GO Option D nach Stabilitätsfenster. Vorgehen:

1. Heute starten: 7-Tage-Beobachtung mit täglichem `pg_stat_activity`-Check.
2. Parallel: PC-2 + PC-3 + PC-4 Operator-Decisions klären.
3. Tag 7 (= 2026-05-21): Wenn 0 unerwartete `tradingbot_gui`-Service-Connections → atomic Lockdown-Block ausführen (Soft, §9).
4. Migrator-Owner-Problem (PC-4) als separate Phase **SEC-1c-3d** auf Backlog (kein Block für Soft-Lockdown).

13. Mainnet-Relevanz

SEC-1c-3c ist **Mainnet-Pre-Condition** — vor MH-7 sollte die SUPERUSER-Vektor-Reduktion abgeschlossen sein. Bei aktuellem Plan: 2026-05-21 (frühester Lockdown-Tag) → bleibt noch >3 Wochen vor typischem MH-7-Zeitfenster. Kein Druck.
Kein direktes **Mainnet-Trading-Risiko** aus dem aktuellen IST-Stand: `tradingbot_gui` ist nicht im Live-Trade-Pfad, sondern nur als Backup-User + DBA-Notfall-Tool. Aber: solange er SUPERUSER ist, ist der Compromise-Vektor (z.B. via gestohlenes PW oder SQL-Injection in einer obskuren Stelle) maximal.

14. Boundaries dieser Plan-Review-Phase

- 0× Code geschrieben
- 0× ALTER ROLE (alle DB-Tests via `SET ROLE … ROLLBACK`)
- 0× DB-Änderung
- 0× Container-Restart
- 0× docker cp
- 0× Push (master `d351a3a` unverändert)
- 0× Mainnet
- 0× Secret-Output (alle Checks Whitelist-strict, Boolean-only)
- 0× `docker compose config`
- 0× `env`-dumps
- 0× `/proc/*/environ`-Bulk