SEC-1d Plan-Review

Passkey/WebAuthn-Integration für Filament-Admin-Panel

Datum: 2026-05-14 (UTC) | Master HEAD: `31ef276` (lokal) | Vorgänger: SEC-1c-1 HTTPS+Domain, SEC-1b-6 TOTP-MFA, SEC-1c-3c-FU1

PLAN-REVIEW **READ-ONLY** **NICHT-TRIVIAL** **5 OPERATOR-DECISIONS**

> **Kurzfassung.** Alle technischen Pre-Conditions für WebAuthn sind erfüllt: HTTPS+stable Domain (RP-ID-fähig), HSTS aktiv, Filament 5 MFA-Provider-Contract ist passkey-tauglich. **Aber: Filament 5 hat keinen nativen Passkey-Provider** (nur TOTP+Email). Und der Stack ist sehr neu (Laravel 13.7 / PHP 8.5 / Filament 5.0), sodass etablierte Community-Plugins (rawilk/laravel-webauthn, asbiin/laravel-webauthn) typischerweise nicht aus dem Karton kompatibel sind. **Empfehlung**: kein sofortiger Code-Start; stattdessen **Operator-Decisions D1–D5** klären und parallel SEC-1c-3c Soft-Lockdown (2026-05-21) durchziehen — bis dahin reicht TOTP-MFA als zweite Stufe vor Mainnet.

1. Boundary-Snapshot

Item	Wert
master HEAD	`31ef276` (SEC-1c-3c-FU1, lokal — kein Remote)
git status	clean
Bot main.py PID	195 (in-container)
Bot Container Host-PID	2657896 — healthy
Worker Container Host-PID	2658058 — running (unhealthy FP)
GUI Container Host-PID	2656633 — running
BINANCE_TESTNET	true
cmd 13 / mp / history	cancelled / 0 / 0
HTTPS gui.*	HTTP/2 200, HSTS `max-age=604800` (1 Woche), CSP report-only
MFA-Status	Steve (id 23) MFA-ENROLLED via TOTP; Admin (id 24) NO-MFA (Legacy/Test)

2. WebAuthn-Pre-Conditions — alle erfüllt

Anforderung	Status	Beleg
HTTPS (Browser erlaubt WebAuthn nur über sicheren Kontext)	**erfüllt**	LE-Cert valid 89d, HTTP/2, alle 3 vhosts
Stable Domain (= RP-ID, darf sich nicht ändern)	**erfüllt**	`gui.gewerbespeicher-rechner.de`
HSTS aktiv (gegen Cert-Stripping)	**erfüllt**	`max-age=604800` (Plan: 1m → 1y staged)
Browser-Compat im Operator-Pool	**unklar**	Operator-Decision D2 nötig
Authenticator-Hardware	**unklar**	Platform-Authenticator (Touch ID / Windows Hello / Android Fingerprint) *oder* Roaming-Authenticator (YubiKey)? Operator-Decision D3

3. Filament 5 native MFA-Provider-Inventory

Provider	Status
`Filament\Auth\MultiFactor\App\AppAuthentication` (TOTP)	**verfügbar** + aktuell aktiv (SEC-1b-6)
`Filament\Auth\MultiFactor\Email\EmailAuthentication`	**verfügbar** (nicht genutzt)
Passkey/WebAuthn	**FEHLT in Filament 5.0**

Filament 5 bietet aber das Interface `Filament\Auth\MultiFactor\Contracts\MultiFactorAuthenticationProvider` — eine schlanke API mit 5 Methoden (`isEnabled`, `getId`, `getLoginFormLabel`, `getManagementSchemaComponents`, `getChallengeFormComponents`). Ein eigener `PasskeyAuthenticationProvider` wäre also implementierbar und über `multiFactorAuthentication([…])`-Array parallel zu TOTP konfigurierbar.

4. Plugin-Optionen — bewertet

Option	Pro	Contra	Aufwand
A. Custom Filament-Provider + `web-auth/webauthn-lib` (Spomky-Labs, framework-agnostic, Symfony-rooted)	Volle Kontrolle, kein Vendor-Lock-in, stable Lib (production-tested), implementiert WebAuthn Level 3	~200-400 LOC Custom-Code + Tests + Migration für Credentials-Tabelle	1-2 Tage Code + Tests
B. `asbiin/laravel-webauthn`	Drop-in Laravel-Integration	Laravel 13-Compat ungewiss (typisch Laravel 11 letzte stable). Filament-Wiring trotzdem manuell.	1-3 Tage (Compat-Fix + Filament-Glue)

C. `rawilk/laravel-webauthn`	analog B	analog B; Maintenance-Status nicht aktiv	analog B
D. Community-Filament-Plugin (z.B. `filament-passkey`)	Wenn vorhanden, sauberste Integration	Erfordert Recherche/Web-Search (Filament-Plugin-Marketplace); Reife & L13-Compat ungeprüft	1 Tag wenn vorhanden, sonst Fallback auf A
E. Externer IdP (Auth0, Cloudflare Access, Authelia)	SSO + WebAuthn ausgelagert	Vendor-Bindung / Kosten / Komplexität für 1-Admin-Setup overkill; SSO-Refactor in Laravel nötig	3-5 Tage

Empfehlung: **A** als primäre Option. Falls Operator-Recherche ergibt, dass D (existierendes Filament-Passkey-Plugin) für L13/Filament 5 maintained ist: D bevorzugen.

5. Coexistence-Modi (Operator-Decision D1)

Filament 5 erlaubt mehrere MFA-Provider parallel im Array:

Modus	Beschreibung	Empfehlung
1. WebAuthn als zweiter MFA-Faktor (zusätzlich TOTP)	User loggt mit Password+(WebAuthn ODER TOTP)	**Sicherster Mittelweg** — Lockout-Schutz, da TOTP als Fallback bleibt
2. WebAuthn replaces TOTP (TOTP wird deaktiviert)	Reine WebAuthn-MFA	**Riskant** — Authenticator-Loss = Lockout. Recovery-Codes als Pflicht.
3. Passwortloser Login (WebAuthn-only, Password optional)	Kein Password mehr nötig; Passkey ersetzt beides	**Long-term-Ziel** — aber große Umbauphase, separater Plan
4. WebAuthn nur als TOTP-Recovery-Fallback	TOTP bleibt primär, WebAuthn nur wenn TOTP-Token weg	**Niedriges Risiko**, aber Operator-Komfortgewinn fraglich

Empfehlung: **Modus 1** — beide Provider parallel aktiv, Operator kann pro Login frei wählen. Bei Authenticator-Loss bleibt TOTP+Recovery-Codes als Fallback.

6. Risikoanalyse

Risiko	Bewertung	Mitigation
Operator-Lockout bei Authenticator-Loss + TOTP-Loss	**mittel-hoch**	Recovery-Codes aktivieren (Filament-Standard 8 Codes), Coexistence mit TOTP (Modus 1), separater SSH-only Break-Glass-Admin als zweiter User
Browser-Compat unzureichend (z.B. ältere mobile Browser, Tor)	**niedrig-mittel**	WebAuthn ist Pflicht-Komponente in allen modernen Browsern seit 2020. Falls Operator mobile-Login auf älterem Gerät nutzt → vorab klären
RP-ID-Drift (z.B. wenn Domain mal wechselt)	**mittel**	Aktuelle Domain stabil seit SEC-1c-1. Falls Domain je wechselt → alle Credentials werden ungültig → Recovery via TOTP.
Custom-Code-Maintenance (bei Option A)	**mittel**	`web-auth/webauthn-lib` hat aktive Maintainer und stabile API. Filament-MFA-Contract ist klein und stabil.
CSP-Konflikte mit WebAuthn-JS	**niedrig**	Aktuelle CSP (report-only) ist permissiv genug. Permissions-Policy `publickey-credentials-get/create` ggf. explizit erlauben.
DB-Schema Drift / Credentials-Migration	**niedrig**	Neue Tabelle `webauthn_credentials` (Migration via tradingbot_gui_migrator — siehe SEC-1c-3d-Block). Bis SEC-1c-3d clear: Migration über aktuelle SUPERUSER-Connection (vor 2026-05-21 Lockdown).
Recovery-Code-Hygiene	**mittel**	Recovery-Codes erscheinen einmal beim Generate — Operator muss diese sicher aufbewahren. Pattern analog zu SECRET-READ-ONCE.

7. Operator-Decisions D1–D5

1. **D1 — Coexistence-Modus**: Modus 1 (TOTP+WebAuthn parallel) / Modus 2 (WebAuthn replaces TOTP) / Modus 3 (Passwortlos) / Modus 4 (Recovery-only)?
2. **D2 — Browser-Pool**: Welche Browser/Geräte nutzt du primär? Desktop Chromium-Edge? Mobile Safari? Tor-Browser? Bringt das Compat-Constraints?
3. **D3 — Authenticator-Typ**: Platform-Authenticator (Touch ID / Windows Hello / Android-FP) und/oder Roaming-Hardware (YubiKey, NitroKey)? Mehrere Geräte registrieren?
4. **D4 — Plugin-Strategie**: Option A (Custom-Code) starten oder zuerst D-Recherche (Community-Plugin) machen?
5. **D5 — Reihenfolge zu SEC-1c-3c**: SEC-1d **vor** Soft-Lockdown 2026-05-21 (parallel laufen lassen) oder **nach** (also frühestens 2026-05-22)?

8. GO/NO-GO

NO-GO für sofortigen Code-Start.

- 5 Operator-Decisions ungeklärt (siehe §7)
- Filament 5 + Laravel 13 + PHP 8.5 ist neuer Stack — Plugin-Kompat-Recherche (D4) erforderlich vor Auswahl
- SEC-1c-3c Stabilitätsfenster läuft noch (bis 2026-05-21) — paralleles Schema-Changes (neue Tabelle `webauthn_credentials`) berühren das Migrator-Owner-Thema

GO als Plan-Phase: Operator klärt D1-D5, dann separate Implementierungs-Phase **SEC-1d-1** mit:

1. Composer-Package installieren (web-auth/webauthn-lib oder Plugin)
2. Migration `webauthn_credentials` + `users`-Erweiterung
3. Custom `PasskeyAuthenticationProvider` implementieren
4. Tests (Setup, Challenge, Login, Recovery)
5. AdminPanelProvider-Edit: `multiFactorAuthentication([AppAuthentication, Passkey])`
6. GUI-Container-Recreate (Composer-install, optimize:clear)
7. Operator-Enrollment (mit existierendem TOTP als Recovery-Anker)

Aufwand-Schätzung Option A: 1-2 Tage Code + 1 Tag Tests + 30 min Cutover. SEC-1c-3c-Soft-Lockdown vorher abschließen ist sauber, aber technisch nicht zwingend (sind getrennte Konzerne).

9. Mainnet-Relevanz

SEC-1d ist **NICHT** zwingender Mainnet-Pre-Condition: TOTP-MFA (SEC-1b-6) ist bereits aktiv und reicht regulatorisch als zweite Stufe. Passkey ist ein **Komfort- und Phishing-Resistance-Upgrade**, kein Sicherheits-Pflicht-Gate.
Andererseits: vor Mainnet ist die Admin-Login-Surface höchst-kritisch (jede Compromise = potentiell Order-Manipulation). Phishing-resistente WebAuthn auf der Admin-Surface ist *sehr* wünschenswert vor MH-7.

Realistisch: SEC-1c-3c (2026-05-21) → SEC-1c-3d (Migrator-Cleanup) → SEC-1d (WebAuthn) → MH-7 wäre die saubere Reihenfolge.

10. Boundaries dieser Plan-Review-Phase

- 0× Code geschrieben
- 0× Container-Restart / docker cp
- 0× DB-Migration / DB-Change
- 0× Push (master `31ef276` lokal unverändert)
- 0× Mainnet
- 0× Secret-Output (Whitelist-Checks Boolean-only)
- 0× `env`-Dumps / `compose config` / `/proc/*/environ`