# SEC-1b-6 MFA Plan-Review — Mehrfaktor-Authentifizierung vor MH-5b/c/d

Projekt: Steve-TradingBot · Phase: SEC-1b-6 (MFA) · Author: claude-opus-4-7[1m]
Generated: 2026-05-13 07:34 UTC · master HEAD: `a185182` (MH-5a closed)
Status: **NO CODE** Plan-Review only — Operator-GO erforderlich vor SEC-1b-6 Code-Phase
Empfehlung: **Variante D — Staged (TOTP jetzt, Passkey BACKLOG)**

**Ziel**: MFA-Pflicht für Admin-Logins, damit MH-5b/c/d Mutation-Actions (Approve / Reject / Pause / Resume / Release) sicher freigeschaltet werden können.

**Vorbedingung**: SEC-1b-0/1/2/3/4/5 abgeschlossen, MH-5a (Read-only ManagedProposalResource) live.

---

## 1 — Live-State-Check (Boundaries)

| Item | Value |
|---|---|
| master HEAD | `a185182` |
| `git status` | clean (außer Container-Rename-Reste, DEFERRED) |
| GUI HTTP `/admin/login` | (Reverse-Proxy-Übergang gerade aktiv — ungefährlich) |
| `APP_URL` | `http://127.0.0.1:8090` — **PLAIN HTTP** |
| `SESSION_SECURE_COOKIE` | nicht gesetzt |
| Bot in-container PID | unverändert |
| Worker in-container PID | unverändert |
| `BINANCE_TESTNET` | `true` |
| Mainnet | 5-layer block ✓ |
| `managed_proposals` rows | 0 |
| Installierte MFA-Bausteine | `pragmarx/google2fa 9.0.0` + `pragmarx/google2fa-qrcode 3.0.0` ✓ |
| Filament-native MFA | `vendor/filament/filament/src/Auth/MultiFactor/{App,Email,Contracts,Http,Pages}` ✓ |

**Major Finding**: Filament 5 hat **native TOTP-MFA** + **native Email-MFA** out-of-the-box. Pakete `pragmarx/google2fa{,-qrcode}` sind bereits installiert (transitive aus filament/filament). **Kein zusätzliches Composer-Paket nötig für Option A.**

**Blocker-Finding**: `APP_URL=http://…` → die GUI läuft aktuell **ohne HTTPS**. WebAuthn/Passkey funktioniert nur in einem „secure context" (HTTPS mit gültigem Zertifikat) — siehe SEC-1a H5 (self-signed cert). Browser akzeptieren Passkey-Enrollment praktisch nicht über plain HTTP und nur unzuverlässig über self-signed HTTPS.

---

## 2 — Option A: TOTP / Authenticator-App (Filament-native)

### Mechanik

1. User-Tabelle erhält 2 verschlüsselte Spalten:
   - `app_authentication_secret` (encrypted text)
   - `app_authentication_recovery_codes` (encrypted JSON-Array, 8 Codes by default)
2. `User` implementiert `HasAppAuthentication` + `HasAppAuthenticationRecovery`:

```
public function getAppAuthenticationSecret(): ?string;
public function saveAppAuthenticationSecret(?string $secret): void;
public function getAppAuthenticationHolderName(): string;        // → email
public function getAppAuthenticationRecoveryCodes(): ?array;
public function saveAppAuthenticationRecoveryCodes(?array $codes): void;
```

3. `AdminPanelProvider`:

```
->multiFactorAuthentication([
    AppAuthentication::make()->recoverable(),
])
->requiresMultiFactorAuthentication()   // Pflicht für alle Admins
```

4. Filament rendert automatisch:
   - Profile-Page „Set up authenticator app“ mit QR-Code (BaconQrCode bereits da via google2fa-qrcode)
   - Recovery-Codes-Anzeige nach Setup
   - 2nd-factor-Step im Login-Flow
   - Recovery-Code-Eingabe als Fallback

### Vorteile

- **Null neue Composer-Pakete** — alles bereits installiert.
- **Browser-/HTTPS-unabhängig** — TOTP ist nur ein 6-stelliger Code.
- **Operator-freundlich** — Google Authenticator / Authy / 1Password / Bitwarden funktionieren alle.
- **Recovery-Codes** built-in (verschlüsselt gespeichert, einmalig nutzbar).
- **Geringes Lockout-Risiko** — Recovery-Codes + Admin-DB-Reset als Notausgang.
- **Vollständig testbar** — Google2FA hat deterministische TOTP-Verifikation mit injizierbarer Zeit.

### Nachteile / Risiken

- **Phishing-anfällig** — TOTP-Codes können in Echtzeit phished + replayed werden (innerhalb der 30s Window).
- **Geräte-Migration** — User-Aufwand wenn das Smartphone wechselt (Recovery-Codes lösen das).
- **Secret-Storage** — `app_authentication_secret` braucht App-Encryption (Laravel `encrypted` cast); kompromittierte `APP_KEY` → MFA-Bruch.

### Aufwand

- DB-Migration: 1 (additive: 2 Spalten auf `users`).
- `User`-Modell: 5 Methoden + 2 casts.
- `AdminPanelProvider`: 2 Zeilen.
- Tests: ~10–15 (Setup-Flow / Login-2nd-Step / Recovery-Code / Reset / Lockout).
- **Geschätzt**: 2–3h inkl. Tests, kein neuer Code-Pfad außerhalb von Filament-Hooks.

---

## 3 — Option B: Passkey / WebAuthn / FIDO2

### Plugin-Landschaft (composer-search-Ergebnis)

| Paket | Filament-Version | Mainline-Stand | Notiz |
|---|---|---|---|
| `jeffersongoncalves/filament-multifactor-passkeys` | „Filament 5 multi-factor“ | unbekannt | Klingt am nächsten am Ziel — Audit nötig |
| `marcelweidum/filament-passkeys` | unbekannt | Community | |
| `statview/filament-passkeys` | unbekannt | Community | |
| Eigenbau auf `web-auth/webauthn-lib` + Filament-Hooks | n/a | mature lib | volle Kontrolle, hoher Aufwand |

**Keiner dieser Plugins ist offiziell von Filament gewartet.** Filament 5 hat **kein** natives Passkey-Modul (nur App-TOTP + Email).

### Mechanik (vereinfacht)

1. User-Tabelle bekommt neue `webauthn_credentials` Tabelle (1:n: credential_id, public_key, sign_count, transports, attestation, created_at).
2. Enrollment-Flow:
   - Browser ruft `navigator.credentials.create(…)` mit Server-Challenge.
   - Server validiert Attestation, speichert credential.
3. Login-Flow:
   - Server liefert Challenge + Allow-List.

- Browser ruft `navigator.credentials.get(…)`.
- User-Verification via Fingerabdruck / FaceID / PIN / Hardware-Key.
- Server verifiziert Signatur via stored public-key.

### Vorteile

- **Phishing-resistent** — Passkeys sind an die Origin gebunden (Relying-Party-ID), kein Replay möglich.
- **UX-Premium** — ein Tap auf Touch-ID / Windows-Hello / YubiKey statt Code abtippen.
- **Skalierbar** — kann pro User mehrere Devices binden, Hardware-Keys + Platform-Keys parallel.
- **Roadmap-konform** — strategische Investition, wenn das System wächst.

### Nachteile / Blocker

- **HTTPS-Zwang** — WebAuthn API funktioniert **nur** auf `https://` (oder `localhost`). Aktuelle `APP_URL=http://127.0.0.1:8090` ist Showstopper. SEC-1a H5 (self-signed cert / Let's Encrypt-Setup) muss zuerst gefixt sein.
- **Origin-Binding** — Relying-Party-ID muss eine **stabile Domain** sein. IP `127.0.0.1` mit Port geht nur via `localhost`-Ausnahme, kein remote-Browser-Setup.
- **Plugin-Risiko** — kein offizieller Filament-Plugin; Community-Plugins sind Audit-Verbindlichkeit + Upgrade-Risiko.
- **Server-Lib-Aufwand** — `web-auth/webauthn-lib` ist ~1200 Klassen, neues Audit-Oberflächengebiet.
- **Lockout-Risiko höher** — Hardware-Key verloren → ohne Fallback-Faktor harter Lockout. Recovery-Code-Mechanismus braucht eigene Implementierung.
- **Testbarkeit** — WebAuthn ist Browser-API; Headless-Tests brauchen entweder gemockte Crypto oder Playwright-/Cypress-WebAuthn-virtuelle-Authenticator.
- **Backup-Faktor zwingend** — Best practice ist Passkey **plus** Recovery-Code / TOTP-Fallback. Pure-Passkey-only ist operativ riskant.

### Aufwand

- DB-Migration: 1 neue Tabelle (`webauthn_credentials`).
- HTTPS-Setup zwingend vorab: Let's Encrypt + Domain + Nginx-Reload (separate Phase SEC-1c).
- Plugin-Audit oder Eigenbau: 6–10h.
- JS-Frontend-Hooks: 2–3h (Filament Livewire braucht JS-Bridge zu WebAuthn-API).
- Tests + Recovery-Flow: 4–6h.
- **Geschätzt**: 12–20h inkl. HTTPS-Vorbedingung.

---

## 4 — Vergleichsmatrix

| Kriterium | Option A: TOTP | Option B: Passkey/WebAuthn |
|---|---|---|
| **Sicherheits-Niveau** | Mittel (HOTP/TOTP-Standard RFC 6238) | **Hoch (FIDO2, origin-bound)** |
| **Phishing-Resistenz** | ✗ replay-fähig im 30s-Window | ✓ origin-bound, nicht phishable |
| **Implementierungs-Aufwand** | ✓ 2–3h (Filament-native) | ✗ 12–20h (inkl. HTTPS) |
| **Lockout-Risiko** | Niedrig (Recovery-Codes built-in) | Mittel (eigener Recovery-Flow nötig) |
| **Rollback-Risiko** | ✓ rein additive Migration | Mittel (neue Tabelle, JS-Hooks) |
| **Testbarkeit** | ✓ Google2FA deterministisch | ⚠ Browser-API, schwerer zu testen |
| **HTTPS-Vorbedingung** | ✓ keine | **HARTE Pflicht** (Blocker H5) |
| **Recovery / UX** | Recovery-Codes + Re-Setup | Recovery-Codes selbst implementieren |
| **Audit-Surface** | Klein (Filament-internes Modul) | Groß (Community-Plugin oder webauthn-lib) |
| **Operator-UX** | Authenticator-App, ~10s Code | 1-Tap Biometrie (wenn Setup okay) |
| **Composer-Pakete neu** | **0** | **1–2** (Plugin + ggf. lib) |
| **Roadmap-Wert** | Kurzfristig pragmatisch | Strategisch + premium |
| **Phasen-Risiko (MH-5b/c/d Block)** | Niedrig | Hoch |

---

## 5 — Vier Architektur-Varianten

### Variante A — TOTP-first only

- Filament-native AppAuthentication + Recovery-Codes.
- Passkey ist niemals geplant.
- **Aufwand**: 2–3h. **Risiko**: niedrig. **Sicherheit**: solide, aber phishing-anfällig.
- **Empfohlen wenn**: schnellster Pfad zu MH-5b/c/d ohne HTTPS-Detour.

### Variante B — Passkey-first only

- SEC-1c (Let's Encrypt + Domain) zuerst, dann WebAuthn-Implementierung.
- Recovery-Code-Flow eigenständig.
- **Aufwand**: 12–20h + SEC-1c vorab. **Risiko**: hoch. **Sicherheit**: top.
- **Empfohlen wenn**: keine zeitlichen Constraints, voll auf Premium-MFA gesetzt wird.
- **Aktuelles Problem**: MH-5b/c/d sind blockiert bis HTTPS + Passkey live; MH-5a allein bringt Operator wenig Wert.

### Variante C — Beide parallel (User wählt selbst)

- Filament `multiFactorAuthentication([AppAuthentication::make(), PasskeyAuthentication::make()])`.
- User aktiviert mindestens einen Faktor; bevorzugt beide.
- **Aufwand**: A + B = 15–25h. **Risiko**: hoch (zwei Code-Pfade gleichzeitig).
- **Empfohlen wenn**: Multi-Operator-Setup mit unterschiedlichen Präferenzen.
- **Aktuell**: Steve-TradingBot hat **einen** Admin — Overkill.

### Variante D — Staged: TOTP jetzt, Passkey später (BACKLOG) EMPFEHLUNG

- **Phase SEC-1b-6 jetzt**: Option A TOTP komplett, unblockt MH-5b/c/d sofort.
- **Phase SEC-1c (parallel / später)**: Let's Encrypt + Renewal + Nginx-HTTPS.
- **Phase SEC-1d (BACKLOG, post-HTTPS)**: Passkey als zusätzlicher Faktor; TOTP bleibt als Fallback.
- **Aufwand**: jetzt 2–3h, später +10h. **Risiko**: minimal pro Phase. **Sicherheit**: jetzt solid, später top.
- **Empfohlen für Steve-TradingBot**: ja.

---

## 6 — Sicherheits-Anforderungen (verbindlich für SEC-1b-6)

| Req-ID | Anforderung | erfüllt von |
|---|---|---|
| **MFA-R1** | MFA Pflicht für alle `UserRole::Admin` | `requiresMultiFactorAuthentication()` im AdminPanelProvider |
| **MFA-R2** | Kein Bypass via `php artisan tinker` / Eloquent-Direkt-Edit (außer Notfall-Reset durch DB-Admin) | dokumentiert + audit-log auf reset-event (BACKLOG) |
| **MFA-R3** | Secrets **niemals** in Logs / Telegram / PDF / GUI | encrypted column + nie in `dd()` / `Log::info()` |
| **MFA-R4** | Recovery-Codes einmalig nutzbar, encrypted at rest | Filament-default ✓ |
| **MFA-R5** | Lockout-Pfad dokumentiert (Operator-Runbook) | SOP: DB-direkt-NULL der `app_authentication_secret`-Spalte mit Audit-Log |
| **MFA-R6** | MFA-Flow respektiert `MAX_LOGIN_ATTEMPTS=3` (SEC-1b-5) | Throttle bleibt auf 1st-Step; 2nd-Step bekommt eigenen Throttle |
| **MFA-R7** | MFA-Setup-Page nur post-Login, nie public | Filament-default (Profil-Page ist auth-required) ✓ |
| **MFA-R8** | TESTNET-Banner / Mainnet-Block bleiben ungetouched | reine GUI-Auth-Layer-Änderung |
| **MFA-R9** | Test-Coverage: Setup / Login-Step / Recovery-Code / Failed-Code / Disabled-User | ≥10 PHPUnit-Tests via Safe-Runner |
| **MFA-R10** | Keine Bot-Code- / Worker- / Watchdog-Berührung | rein GUI-Layer ✓ |
| **MFA-R11** | Backup vor Migration (durable rule: pg_dump + state-snapshot) | Pflicht-Schritt im Implementations-Plan |
| **MFA-R12** | Restart-Anforderung dokumentiert (web-server reload reicht; kein Bot/Worker) | klar abgegrenzt |

---

## 7 — Empfehlung

**Variante D — Staged (TOTP-first jetzt, Passkey post-HTTPS später)**.

### Begründung

1. **Filament 5 native TOTP** + bereits installierte `pragmarx/google2fa{,-qrcode}` reduzieren Implementierungs-Risiko auf nahezu Null.
2. **HTTPS-Blocker (SEC-1a H5)**: Passkey braucht zwingend HTTPS mit gültigem Zertifikat. `APP_URL=http://127.0.0.1:8090` ist heute Showstopper. Diese Infrastruktur-Vorbedingung (Let's Encrypt + Domain) gehört in eine **eigene SEC-1c-Phase**, nicht in SEC-1b-6.
3. **MH-5b/c/d Operator-Wert**: Approve / Reject / Pause / Resume / Release sind blockiert bis MFA da ist. TOTP unblockt diesen Pfad innerhalb von 2–3h.
4. **Operator-Realität**: 1-Admin-Setup; Passkey als sole-factor wäre Lockout-anfälliger als TOTP+Recovery-Codes.
5. **Phishing-Restrisiko mit TOTP** ist im TESTNET-only-Setup mit `MAX_LOGIN_ATTEMPTS=3` + fail2ban + 169-IP-Banlist **akzeptabel**, solange Mainnet hard-blocked bleibt.
6. **Passkey als BACKLOG-Phase SEC-1d** bleibt offen, sobald SEC-1c (HTTPS/Let's Encrypt) live ist.

### NO-GO-Bedingungen für SEC-1b-6

- Implementierung ohne pg_dump + state-snapshot vorab → **NO-GO** (durable rule).
- Implementierung ohne Test-Run via Safe-Runner → **NO-GO** (durable rule post-INCIDENT).
- Bot/Worker-Restart wird **nicht** benötigt — sollte einer geplant sein, ist das ein Indikator für Scope-Bruch.

### Phasen-Sequenz (Vorschlag)

| Phase | Inhalt | Block für |
|---|---|---|
| **SEC-1b-6 (Variante D Step 1)** | TOTP-MFA Filament-native, 2 DB-cols, Recovery-Codes, ~10 Tests | unblockt MH-5b/c/d |
| MH-5b | Request + Reject Actions | nach SEC-1b-6 |
| MH-5c | Approve-Wizard 5 Steps | nach SEC-1b-6 |
| MH-5d | Pause/Resume/Release Actions | nach SEC-1b-6 |
| SEC-1c | DB-Privs / Let's Encrypt / Admin / Session-Encrypt | parallel zu MH-5x erlaubt |
| **SEC-1d (BACKLOG)** | Passkey/WebAuthn als zusätzlicher Faktor (Variante D Step 2) | nach SEC-1c HTTPS + Domain |
| MH-6 | Worker-Handler für 8 CommandTypes | nach MH-5 + SEC-1b-6 |
| MH-7 | Bot-Wiring (Restart!) | nach SEC-1c |

---

## 8 — Implementierungs-Skizze für Variante D Step 1 (TOTP) — NUR PLAN

**Geliefert würde** (im späteren Implementations-Patch, nicht jetzt):

1. **Migration** `users` add 2 columns:
   - `app_authentication_secret` (string, encrypted, nullable)
   - `app_authentication_recovery_codes` (text, encrypted, nullable)
2. **`User`-Modell** implementiert `HasAppAuthentication` + `HasAppAuthenticationRecovery`; 2 `encrypted` casts.
3. **`AdminPanelProvider`**:

```
->multiFactorAuthentication([
    \Filament\Auth\MultiFactor\App\AppAuthentication::make()->recoverable(),
])
->requiresMultiFactorAuthentication()
```

4. **`Login`-Page**: keine Änderung. 2nd-step wird von Filament automatisch eingehängt.
5. **Operator-Runbook**: `docs/ops/sec_1b_6_mfa_runbook.md` mit Setup-Flow, Recovery-Code-Storage-Empfehlung, Lockout-Reset-SOP.
6. **Tests** (≥10) via Safe-Runner:
   - Setup-page accessible only post-login.
   - Setup persists encrypted secret + recovery codes.
   - Login-2nd-step verifies TOTP code (mocked time).
   - Login-2nd-step accepts valid recovery code (and burns it).
   - Login-2nd-step rejects invalid code.
   - User without setup is forced to enrollment (admin route).
   - Admin-Role required ist orthogonal.
   - Throttle Login-2nd-step (separat zu 1st-step).
   - Recovery-Codes nicht im Cleartext in DB.
   - `Log::*` schreibt nie das Secret.
7. **Backup**: pg_dump `tradingbot_gui` + `.env` + state-snapshot **VOR** Migration.

### Drift-Audit-Liste (vor docker cp / Restart — durable rule)

- Pre-Restart: GUI-DB pg_dump, `.env` snap, `live_portfolio.json` snap, Bot-PID-stamp, fail2ban-status-snap.
- Migration ist additive only — Rollback via `down()` möglich.
- Kein `docker cp` notwendig (Migration läuft via `php artisan migrate` im GUI-Container).
- Kein Bot/Worker-Restart.

---

## 9 — Operator-Entscheidungs-Punkte

Bevor SEC-1b-6 Code-Phase startet, GO/NO-GO erforderlich auf:

1. **Variante** A / B / C / D — Empfehlung: **D**.
2. **Recovery-Code-Anzahl**: 8 (Filament-default) ✓?

3. **Enrollment-Strategy für bestehenden Admin-User**:
   - (a) On-first-login nach Migration zwingend (Filament-default), oder
   - (b) Email-Notification + Grace-Period.

   Empfehlung: **(a)** — wir haben nur 1 Admin, sofortige Enrollment unkritisch.
4. **Lockout-Reset-Pfad**: DB-direkt + Audit-Log-Eintrag — okay?
5. **Telegram-Notification bei MFA-Setup / Lockout-Reset**: ja (durable rule SEC) oder nein?
6. **SEC-1c (HTTPS/Let's Encrypt) als nächste SEC-Phase parallel** oder erst nach MH-5d?

---

**Status**: PLAN-REVIEW Plan fertig. Warte auf Operator-GO für Variante + Antworten auf Punkte 1–6.
**Kein Code geschrieben.** Kein git / docker / test-Touch durchgeführt — pure analysis only.