SEC-1a — GUI/DB/OPS Security Audit

Projekt: Steve-TradingBot · Phase: **SEC-1a Read-only Audit** · Author: claude-opus-4-7[1m]
Generated: 2026-05-13 00:01 UTC · master HEAD: `cd37822` (MH-4c closed)
Audit-Scope: GUI Exposure · Filament/Laravel Auth · DB · Secrets · Logs · Docker/Ops
Audit-Modus: **READ-ONLY** — keine Config-Mutation, kein Restart, kein Patch
Status: **NO-GO für MH-5 ohne SEC-1b**

Inhaltsverzeichnis

---

1 — Executive Summary

Severity	Count	Bedeutung	Status
CRITICAL	2	Active Attack-Surface ODER Data-Exfil-Risk	**SOFORT-Fix vor MH-5**
HIGH	6	Pre-Mainnet-Blocker; mehrheitlich Pre-MH-5-Blocker	**SEC-1b/c Mandatory**
MEDIUM	8	Mitigierbar vor MH-5 ODER Mainnet	**Hardening**
LOW	6	Hardening-Backlog	**Optional**
INFO	5	Architektonische Awareness	**Doku**

Verdict: NO-GO für MH-5 bis SEC-1b komplett. Begründung in §3 + §12.

---

2 — Audit-Methodik + Scope-Abgrenzung

Methodik

Read-only Inspection aller relevanten Surface-Layer via Bash-Calls. **Keine Mutation, kein Restart, keine Config-Änderung.** Befunde wurden via:

- `ss -tlnp` für Port-Inventur (Host)
- `docker ps` + `docker inspect` für Container-Exposure und Network-Isolation
- `cat /etc/nginx/sites-*` für Reverse-Proxy + TLS + Auth-Config
- `psql \du` + `information_schema.table_privileges` für DB-Rollen
- `stat` + `find` für Filesystem-Permissions + Secret-Backup-Inventur
- `cat .env` für Secret-Surface (redacted output only)
- `tail /var/log/auth.log` + `nginx access.log` + `laravel.log` für aktuelle Attack-Aktivität
- `systemctl is-active fail2ban` + `sshd_config` für SSH-Hardening-Status
- `curl http://127.0.0.1:<port>` für Service-Identifikation (admin tools)

Scope-Abgrenzung

In Scope	Out of Scope
GUI-Exposure (Nginx, Ports, TLS, Auth, Rate-Limits)	Code-Audit der Filament-Resources (statisch, MH-5 Plan-Review-Sache)
Filament/Laravel-Auth (Sessions, CSRF, 2FA, Throttling)	Vulnerability-Scanner-Run (npm audit / composer audit) — kann SEC-1d sein
DB-User-Permissions + pg_hba	SQL-Injection-Pen-Test der Filament-Forms (MH-5 spezifisch)
Secrets (.env, htpasswd, Logs-Leak)	Key-Rotation-Strategie (SEC-1c)
Logs (Auth-Failures, Audit-Surface)	SIEM/Loki-Aggregation (SEC-1d Hardening)
Docker/Ops (exposed ports, watchdog, cron)	Bot-Side Python Code-Audit (separate Phase)

---

3 — Bedenken & Begründung: Warum SEC vor MH-5

Die Operator-Frage war explizit: *"VOR MH-5 müssen wir die Security fertigstellen!"*. Das ist begründet durch:

3.1 Risikoeskalation durch MH-5

Aspekt	Pre-MH-5 (jetzt)	Post-MH-5
Operator-Actions live	0 (alle Phasen bisher: dormant code, kein UI-Trigger)	6 (request/approve/reject/pause/resume/release)
Filament-UI-Surface	BaselineHoldings + Apply-Profile (existing)	+ ManagedProposalResource + ManagedAssets Page + Approve-Wizard
State-File-Mutation	nicht direkt (Worker MH-6 noch nicht)	nicht direkt (Worker MH-6 immer noch nicht; aber commands queued)
Approver-Privilege-Theft-Impact	niedrig (nur view)	HOCH (approve = später Live-Promote-Path)
UI-Code-Bug-Exposure	5 existing Filament-Resources	+ 5+ neue Resources + Wizard mit Multi-Step-Logic

3.2 Konkrete Risk-Multiplier

1. **Filament-Wizard-Code = neue Attack-Surface**. Jeder MH-5-Bug bei aktivem APP_DEBUG leakt Stack + DB-Queries + Pfade.
2. **Approve-Action im Wizard = potentielle Live-Trading-Brücke**. Auch wenn Worker MH-6 noch nicht greift — die Audit-Trails / Commands sind real. Approver-Account-Theft jetzt = vorhersehbarer Schaden später.
3. **SSH-Brute-Force IST aktiv**. Mid-MH-5-Implementation = Mid-Compromise. Wir bauen in einer "leakenden" Umgebung weiter.
4. **DB-SUPERUSER-Privilege**. Filament hat Form-Inputs in jeder neuen Resource — bei SQL-Injection-Bug = volle DB.
5. **Single-Factor-Auth für Admin**. Ein Password schützt alle 6 zukünftigen MH-5-Actions. Industry-Standard ist 2FA für privilegierte Actions.

3.3 Was MH-5 NICHT entschärft

MH-5 hat eigene Defense-Layer (admin-only actions, hardcoded testnet, Hard-Confirm), aber:

- Admin-only-canAccess() greift NUR wenn Login-Phase überstanden — Brute-Force gegen den 1 admin-account ist heute schon möglich
- Hard-Confirm schützt Operator vor Tippfehler, NICHT vor compromisettem Cookie
- testnet-hardcode greift NUR wenn Bot & Worker korrekt initialisiert — bei DB-Manipulation via SUPERUSER kann Operator nicht garantieren dass commands.environment-Spalte unverändert ist

---

4 — Findings: CRITICAL (2)

C1 — CRITICAL SSH PermitRootLogin yes + fail2ban INACTIVE + ACTIVE BRUTE-FORCE

Evidence	`/etc/ssh/sshd_config: PermitRootLogin yes`
	`systemctl is-active fail2ban → inactive`
	`/var/log/auth.log:` aktiv um 23:55 UTC live brute-force:
	
	`May 12 23:55:54 sshd[2727994]: Failed password for root from 45.148.10.147 port 15662 ssh2`
	`May 12 23:55:57 sshd[2728599]: Invalid user db2inst1 from 118.99.102.207 port 54350`

Risk	Server-Takeover möglich; gesamter Bot+GUI+DB-Stack kompromittierbar. Eine successful brute-force-Login = root-shell = full credential dump via /projekte/Steve-TradingBot/.env + GUI-DB + Telegram-Token + Anthropic/OpenAI Keys.
Erklärung	SSH ist der primäre Server-Access-Path. Default-Ubuntu lässt root-login mit Password zu. Ohne fail2ban gibt es keine IP-Throttling-Schicht. Active brute-force-Versuche von zwei verschiedenen IPs zeigen dass die IP bereits in Standard-Scan-Listen ist.
Recommended Fix	1. `/etc/ssh/sshd_config: PermitRootLogin prohibit-password` (oder no, falls SSH-Key vorhanden) 2. `apt install fail2ban` + ssh.local jail enabled 3. UFW oder iptables IP-allowlist falls Operator nur von festen IPs zugreift 4. `systemctl restart sshd`
Phase	SEC-1b SOFORT — vor jeder weiteren Code-Phase

C2 — CRITICAL APP_DEBUG=true + APP_ENV=local in GUI Production

Evidence	`/projekte/Steve-TradingBot/gui/.env:`
	
	`APP_ENV=local`
	`APP_DEBUG=true`

Risk	Laravel zeigt bei jedem unhandled Exception: • Full Stack-Trace (alle Files + Line-Numbers) • Aktuelle Environment-Variablen (DB_PASSWORD, APP_KEY, etc. via Whoops-Page) • Aktuelle DB-Queries (mit Params) • Server-Pfad-Struktur (/var/www/html/...) Jeder MH-5 Filament-Form-Bug = volle Disclosure für Drive-By-Visitor.
Erklärung	Laravel hat zwei Production-Sicherheits-Flags: APP_ENV + APP_DEBUG. Beide müssen `production` bzw `false` sein in Production-Deployments. Aktuell ist GUI auf `local/true` konfiguriert — das ist Development-Standard, NICHT Production-tauglich.
Recommended Fix	1. `gui/.env: APP_DEBUG=false` 2. `gui/.env: APP_ENV=production` 3. `docker exec steve-tradingbot-gui php artisan config:cache` 4. `docker exec steve-tradingbot-gui php artisan view:cache`
Phase	SEC-1b SOFORT

---

5 — Findings: HIGH (6)

H1 — HIGH DB User tradingbot_gui ist PostgreSQL SUPERUSER

Evidence	<pre>\du
	   Role name
		Attributes
		--------------- 	
		+---
		 tradingbot_gui
			Superuser, Create role, Create DB,
			Replication, Bypass RLS</pre>

Risk	App-User kann jede Tabelle ALTER, jeden Role anlegen, Replication starten — bei GUI-RCE (SQLi, deserialization, debug-disclosure) = volle DB-Kontrolle, inkl. Replikation auf externen Server (Datenexfil), audit_events-Mutation (Audit-Trail-Tampering), oder löschen.
Erklärung	Postgres-Best-Practice: App-User hat NUR die Privilegien die er braucht (SELECT/INSERT/UPDATE/DELETE/REFERENCES auf eigene Tabellen). SUPERUSER ist nur für Migration-Run und sollte separater User sein.
Recommended Fix	1. neuen App-User anlegen: `CREATE USER tradingbot_app WITH PASSWORD '...';` 2. `GRANT SELECT,INSERT,UPDATE,DELETE,REFERENCES ON ALL TABLES IN SCHEMA public TO tradingbot_app;` 3. `GRANT USAGE ON SCHEMA public TO tradingbot_app;` 4. App .env auf neuen User umstellen 5. SUPERUSER-Role auf separaten migrate-only User behalten
Phase	SEC-1c vor MH-5 (oder unmittelbar parallel)

H2 — HIGH Portainer publicly exposed auf 0.0.0.0:9443

Evidence	`docker ps: portainer 0.0.0.0:9443->9443/tcp, 0.0.0.0:8000->8000/tcp` `curl https://127.0.0.1:9443 → Portainer login page`
Risk	Portainer ist Web-UI für gesamten Docker-Stack inkl. Container-Shell-Action. Bei Login-Compromise = Pivot zu Bot-Container = Bot-Live-Trading-Kontrolle, Volume-Mount-Manipulation, Image-Pull-Replace.
Erklärung	Portainer-Default-Listening ist 0.0.0.0:9443. Für Production sollte Portainer entweder über VPN-only, hinter Nginx-Reverse-Proxy mit Basic-Auth + Rate-Limit, oder auf localhost gebunden sein.
Recommended Fix	docker-compose.yml für Portainer-Service: `ports: ["127.0.0.1:9443:9443"]` + Optional Nginx-Reverse-Proxy mit Basic-Auth-Layer für Remote-Access
Phase	SEC-1b vor MH-5

H3 — HIGH Cockpit-ws publicly exposed auf 0.0.0.0:9090

Evidence	`ss -tlnp: cockpit-ws pid=1439585 listening 0.0.0.0:9090` `curl http://127.0.0.1:9090 → Cockpit Loading page`
Risk	Cockpit ist Linux-Admin-Web-UI mit Shell-Access. Bei Login-Compromise = root-shell des Hosts = full server-takeover, DB-Direct-Access, .env-Dump.
Erklärung	Cockpit ist auf vielen Ubuntu-Systemen vor-installiert. Default-Setup hat keine IP-Allowlist. Listening auf 0.0.0.0 = von überall im Internet erreichbar.
Recommended Fix	Cockpit-systemd-socket-config (`/etc/systemd/system/cockpit.socket.d/listen.conf`) auf `ListenStream=127.0.0.1:9090` setzen + `systemctl daemon-reload + systemctl restart cockpit.socket`
Phase	SEC-1b vor MH-5

H4 — HIGH Dev-WordPress public auf 0.0.0.0:18080

Evidence	`docker ps: dev-wordpress-1 0.0.0.0:18080->80/tcp`
Risk	Standard-Wordpress-Installation, vermutlich nicht maintained. WordPress hat regelmäßige CVEs. Bei nicht-aktuellen Plugins = potentieller RCE-Vektor. Kompromiss erlaubt Pivot ins Docker-Network und damit zu anderen Services.
Erklärung	"Dev-" Prefix deutet auf Development-Container. Bleibt aber laufen seit 8 Tagen — vergessene Test-Umgebung erhöht Attack-Surface.
Recommended Fix	Falls WordPress nicht benötigt: `docker stop dev-wordpress-1 dev-wpcli-1 dev-db-1` . Falls benötigt: `ports: ["127.0.0.1:18080:80"]` + Plugin-Updates + WAF.
Phase	SEC-1b

H5 — HIGH Self-signed SSL ohne CA-Validation

Evidence	`dashboard-ssl: ssl_certificate /root/steves-tradingbot/ssl/dashboard.crt` `/etc/letsencrypt/live/` ist leer
Risk	Browser-Warning desensibilisiert Operator ("Klick auf Weiter trotz Warning"). MITM auf TLS-Schicht möglich, weil Operator-Cert-Validation in der Praxis übersprungen wird. Auch für Telegram-Webhooks problematisch.
Erklärung	Let's Encrypt ist kostenlos und automatisierbar. Self-Signed-Certs sind nur für DEV-Tests akzeptabel; in Production untergrabbar.
Recommended Fix	1. Domain für 81.169.213.37 registrieren (z.B. `steve-tradingbot.example.com`) 2. `certbot --nginx -d steve-tradingbot.example.com` 3. Nginx-Config auf ACME-validated cert umstellen
Phase	SEC-1c vor Mainnet (kein MH-5-Blocker da MH-5 testnet-only)

H6 — HIGH 2FA für Filament-Admin NICHT vorhanden

Evidence	`SELECT * FROM users:` 1 row: `id=23, email=admin@example.local, role=admin` users-Tabelle hat KEINE `two_factor_secret`, `two_factor_recovery_codes`, `two_factor_confirmed_at` Spalten.
Risk	Password-only Auth für Operator-Action-Privilege. Approve/reject/release haben High-Impact (Live-Trading-Promote). Credential-Theft (Phishing, Keylogger, Reused-Password-Database) = full Bot-Control.
Erklärung	Industry-Standard für privilegierte UI-Actions ist 2FA via TOTP (Google Authenticator). Filament hat 2FA-Plugins (z.B. `stechstudio/laravel-totp` oder Laravel Fortify). MH-5 macht Approve-Wizard real — ohne 2FA ist Single-Password ein Single-Point-of-Failure.
Recommended Fix	1. Migration für 2FA-Spalten (`two_factor_secret`, `two_factor_recovery_codes`) 2. Filament-2FA-Plugin install + Konfig 3. Admin user 2FA-Enrollment-Flow 4. Filament canAccess() prüft 2FA-confirmed-at NOT NULL
Phase	**SEC-1c vor MH-5** — MH-5 macht admin-only actions live

---

6 — Findings: MEDIUM (8)

#	Sev	Finding	Risk	Fix	Phase
M1	MED	Login-Throttling unbekannt / nicht explicit konfiguriert	Brute-Force gegen admin@example.local möglich	RateLimiter explicit in AppServiceProvider pinnen (3/min/IP + per-user-lockout 15min) + Test	SEC-1b
M2	MED	`.env` Files 644 (world-readable)	Bei jedem non-root user-shell-access leakt secret	`chmod 600 .env` + zukünftige Backups mit 0600	SEC-1b
M3	MED	`/srv/shares/backups/` enthält .env Kopien	Single chmod-Fehler exposed Secrets via shares-8088 autoindex (heute durch 0700 parent-dir mitigiert)	Backups außerhalb `/srv/shares/` verschieben ODER `.env` aus Backup-Snaps löschen	SEC-1b
M4	MED	Nur 1 admin mit default-haftem Email `admin@example.local`	Default-Account-Antipattern; Password möglicherweise schwach	Email auf real-name; Password rotieren; optional backup-admin	SEC-1c
M5	MED	`SESSION_ENCRYPT=false`	Session-Cookies in DB klartext-lesbar bei DB-Access	`SESSION_ENCRYPT=true` + config:cache	SEC-1c
M6	MED	HR-6 + recon24 temp-vhost-configs in `sites-available/` (orphan, NICHT enabled)	Risk-of-accidental-enable; cognitive load	`rm` der temp-vhosts nach Phase-Closure	SEC-1d
M7	MED	Worker-Container "unhealthy" Status pre-existing	Symptom-Mask: echte Outage-Detection erschwert; Watchdog respawnt regelmäßig	Healthcheck-Script im Worker-Image auf realen `command_worker` heartbeat anpassen	SEC-1d Ops
M8	MED	Laravel.log enthält File-Path-Disclosure	Path-Disclosure für Recon (`/var/www/html/storage/app/...`)	Log-Level auf error; APP_DEBUG=false reduziert (siehe C2)	SEC-1b (via C2)

7 — Findings: LOW (6) + INFO (5)

LOW

#	Finding	Fix
L1	CSRF-Middleware-Setup unbekannt (Laravel 11+ Pattern hat verschoben)	`bootstrap/app.php` ValidateCsrfToken-config prüfen
L2	dashboard-ssl proxied Bot-Dashboard (port 8050) NICHT Filament-GUI (port 8090) — möglicher Misconfig	Nginx-Routing prüfen ob Filament wirklich extern erreichbar; aktuell nur via HR-6/recon24-vhosts (orphans)
L3	Sendmail-MTA listening 127.0.0.1:25 — Default Ubuntu, nicht used	`systemctl disable sendmail` falls nicht benötigt
L4	Multiple .env backups (10+) im Repo + /srv/shares/backups/	Lifecycle für Backups (rotation + secure storage)
L5	netdata exposed auf 0.0.0.0:9080	Bind 127.0.0.1 oder ip-allowlist
L6	Sendmail-Mail-Submission auf 127.0.0.1:587	OK as-is (localhost only)

INFO

#	Finding	Bedeutung
I1	Public IP 81.169.213.37	Direct-IP-Exposure; kein DNS-Hostname registriert
I2	Bot port 8050 publicly exposed	Bot-internal-API extern erreichbar — Endpoint-Surface prüfen (siehe L2)
I3	Multiple Docker networks isoliert	Defense-in-Depth durch network-isolation; GUI+Bot+DB+Worker im `steve-tradingbot_clawbot-net`
I4	UFW NICHT installiert	Firewall-Layer fehlt — iptables direct, keine Chain-Inspection
I5	Watchdog läuft als root via cron `*/5`	Akzeptabel — root-execution erforderlich (docker control)

---

8 — Sofortige Stop-Trigger

Trigger	Auswirkung
C1 SSH-Brute-Force aktiv	NICHT MH-5 starten bis SSH abgesichert. Risk: Mid-Implementation-Compromise.
C2 APP_DEBUG=true	NICHT MH-5 starten — jeder Filament-Error leakt Stack + Pfade + DB-Queries
H1 DB SUPERUSER	NICHT MH-5 starten — Filament RCE = volle DB-Übernahme
H2 Portainer public	Independent vector — Container-Pivot zu Bot via Portainer-Shell-Action
H3 Cockpit-ws public	Independent vector — Server-Admin via Cookie-Theft
H6 Kein 2FA für Admin	MH-5 macht admin-only Actions REAL — Single-Password unzureichend

---

9 — Pflicht-Fixes pro Phase

SEC-1b (Pre-MH-5 Mandatory)

1. **C1**: SSH `PermitRootLogin prohibit-password` + fail2ban install/enable + UFW oder iptables IP-allowlist
2. **C2**: `APP_DEBUG=false`, `APP_ENV=production`, `php artisan config:cache`
3. **H2**: Portainer auf 127.0.0.1 binden
4. **H3**: Cockpit-ws auf 127.0.0.1 binden
5. **H4**: dev-wordpress-1 stoppen oder localhost-bind
6. **H6**: Filament 2FA install + admin user 2FA-Enroll
7. **M1**: Login-Throttling explicit pinnen (3/min, lockout 15min)
8. **M2**: `.env` chmod 600
9. **M3**: Backups mit `.env`-Inhalten außerhalb shares/ verschieben
10. **M8**: erledigt via C2 (APP_DEBUG=false)

SEC-1c (Pre-Mainnet Mandatory)

1. **H1**: DB user privileges reduzieren (kein SUPERUSER für App)
2. **H5**: Let's Encrypt SSL für Production (kein self-signed)
3. **M4**: Admin email + password rotation
4. **M5**: `SESSION_ENCRYPT=true`
5. **L1**: CSRF-Middleware Verification
6. **L2**: dashboard-ssl Routing Review

SEC-1d (Hardening Backlog, post-MH-5)

1. **M6**: orphan vhosts cleanup
2. **M7**: Worker healthcheck fix
3. **L3/L5**: sendmail/netdata bind-restriction

4. **L4**: Backup-Rotation-Lifecycle
5. Centralized Audit-Log-Aggregation (Grafana Loki)
6. Security-Headers global (CSP, HSTS, X-Frame-Options, Permissions-Policy)
7. Vulnerability-Scanner-Run (`npm audit` / `composer audit`)
8. Filament-Form-Security-Pen-Test (SQLi / XSS / CSRF)

---

10 — Zukünftige Schnittstellen + Phaseneinordnung

10.1 Phasen-Roadmap mit SEC-Integration

Phase	Inhalt	SEC-Vorbedingung
SEC-1a ✅ jetzt	Read-only Audit (dieses Dokument)	—
SEC-1b nächstes	SOFORT-Fixes (C1/C2/H2/H3/H4/H6/M1/M2/M3)	SEC-1a Operator-Approval
MH-5	Filament Wizard / ManagedHoldings UI	**SEC-1b komplett**
SEC-1c parallel	DB-Privs reduzieren, Let's Encrypt, Admin-Rotation, Session-Encrypt	parallel zu MH-5 erlaubt
MH-6	Worker-Handler (8 CommandTypes mit Two-File-Atomic)	SEC-1b zwingend; SEC-1c bevorzugt
SEC-1d Hardening	Logs / Healthcheck / Security-Headers / Pen-Test	parallel oder post-MH-5
MH-7	Bot-Wiring (Restart-Pflicht; Bot-self-emit flag_managed_drift)	SEC-1c komplett; SEC-1d bevorzugt
MH-8	Testnet-Drill End-to-End	SEC-1c komplett
MH-9 / Mainnet	Worker-Daemon Aktivierung + Mainnet-Pre-Sign-Off	**SEC-1c + SEC-1d komplett** + 5-Layer-Mainnet-Block verifiziert

10.2 Schnittstellen zu zukünftigen Komponenten

SEC ↔ MH-5 (Filament UI)

- SEC-1b H6 (2FA) wird MH-5 admin-only-Actions absichern
- SEC-1b M1 (Login-Throttle) bietet Brute-Force-Schutz für admin-account
- SEC-1b C2 (DEBUG=false) verhindert Stack-Leak bei MH-5-Form-Bugs
- MH-5 selbst hat Defense-Layer: canAccess(Admin), Hard-Confirm, environment='testnet' hardcoded

SEC ↔ MH-6 (Worker-Handler)

- SEC-1c H1 (DB-Privs) reduziert Worker-DB-Surface (Worker hat eigene Credentials sinnvoll)
- SEC-1c M5 (Session-Encrypt) hat keinen Worker-Touch (Worker ist Backend)

- MH-6 Worker schreibt managed_state.json + audit_snapshots — keine zusätzlichen SEC-Anforderungen vs. existing BaselineWriter

SEC ↔ MH-7 (Bot-Wiring)

- Bot-Restart-Pflicht in MH-7 — SEC-1b/c MUSS vorher abgeschlossen sein (kein Restart in unsicherer Umgebung)
- flag_managed_drift Bot-Self-Emit ist NEU im Worker-Flow — Audit-Trail-Surface erweitert; SEC-1d Audit-Aggregation hilft

SEC ↔ Mainnet-Pre-Sign-Off

- 5-Layer-Mainnet-Block bleibt unabhängig von SEC; SEC sichert die Layer ab (Filament-canAccess + DB-Read-Path-Trust)
- SEC-1c H5 (Let's Encrypt) ist Pre-Mainnet-Pflicht für Operator-Trust
- SEC-1d Pen-Test wäre Mainnet-Bonus

10.3 Backlog-Pin-Vorschlag

Nach SEC-1b Closure: **memory/sec_1a_audit_findings_pin.md** mit Verweis auf alle 27 Findings + Status-Tracking (offen/in-progress/closed) wäre sinnvoll.

Nach SEC-1c Closure: **memory/sec_pre_mainnet_checklist.md** mit allen pre-Mainnet-Pflichtfixes (H1/H5/M4/M5) als hard-gate vor MH-9.

---

11 — SEC-1b Aufwands-Schätzung

Item	Effort	Risk
C1 SSH-Hardening (sshd_config + fail2ban + iptables)	15min	LOW (well-known)
C2 APP_DEBUG + APP_ENV + config:cache	5min	LOW
H2 Portainer rebind to 127.0.0.1	15min (docker-compose edit + restart)	LOW
H3 Cockpit-ws rebind	10min (systemd-override)	LOW
H4 dev-wordpress-1 stop oder rebind	5min	LOW
H6 Filament-2FA install + Admin enroll	60-90min	MEDIUM (Migration + Plugin-Config + Testflow)
M1 Login-Throttling pin + Test	30min	LOW
M2 chmod 600 .env	5min	LOW
M3 Backup-Secret-Cleanup	15min	LOW
Total	**~2.5-3h**	**MEDIUM (H6 ist Hauptaufwand)**

Backup-Pflicht vor SEC-1b: `pg_dump` + `cp /etc/ssh/sshd_config` + `cp docker-compose.yml` + `cp gui/.env` als Rollback-Punkte.

---

12 — GO/NO-GO Empfehlung MH-5

Empfehlung: NO-GO für MH-5 ohne SEC-1b

Begründung:

Aspekt	Bewertung
C1 SSH-Attack-in-Progress	LIVE attack; mid-implementation-compromise-risk acute
C2 APP_DEBUG=true	Jeder MH-5 Filament-Bug leakt Stack — MH-5 hat viel neuen Filament-Code = viele potentielle Errors
H6 Kein 2FA + admin-only actions	MH-5 macht approve/reject/release Live-Path real — Single-Factor unzureichend
H1 DB SUPERUSER + MH-5 Wizard mit DB-Reads	Erhöhtes Risiko bei SQL-Injection-Vektor in Filament-Form

MH-5 ist nur akzeptabel mit:

1. **SEC-1b komplett** (alle CRITICAL + HIGH H2/H3/H4/H6 + MEDIUM M1/M2/M3) **VOR** Code-Beginn MH-5
2. **SEC-1c** kann parallel zu MH-5 laufen (H1 DB-Privs / H5 SSL / M4 Admin / M5 Session-Encrypt) — kein MH-5-Blocker
3. **SEC-1d** als Sprint-Backlog post-MH-5

Empfohlener Pfad

```
SEC-1b (Pre-MH-5 Mandatory)
   ├─ SSH-Hardening + fail2ban + UFW
   ├─ APP_DEBUG=false + config:cache
   ├─ Portainer/Cockpit/Dev-WP localhost-bind
   ├─ Filament-2FA + admin enrollment
   ├─ Login-Throttling explicit
   ├─ .env chmod 600
   └─ Backups mit Secrets aus /srv/shares/ raus
        ↓
[GO MH-5 wird hier möglich]
        ↓
SEC-1c parallel zu MH-5 (DB-Privs / SSL / Admin / Session-Encrypt)
        ↓
SEC-1d Hardening-Backlog (post-MH-5)
        ↓
[GO Mainnet wird hier möglich, alle SEC-1c + Mainnet-Pre-Conditions]
```

STOP vor SEC-1b. Erwarte Operator-GO für Security-Hardening-Phase. **NO-GO für MH-5** bis SEC-1b komplett — insbesondere C1 (SSH-Attack), C2 (APP_DEBUG), H6 (2FA).