SICHERHEITS-AUDIT — RUNDE 4

# NetMon Roots v1.1.15

Code Security Review — Juni 2026

| | |
|---|---|
| Datum | 29. Juni 2026 |
| Analysierte Version | v1.1.15 — Runde 4 |
| Geaenderte Dateien (gesamt) | 40+ |
| Sicherheitsfokus-Module | 20 |
| Bestaetigte Findings | 0 |
| Bewertete Kandidaten | 1 (K1 — FP) |
| Konfidenz-Schwelle | >= 8 / 10 |
| Methode | Statische Code-Analyse + False-Positive-Filterrunde |
| Analysewerkzeug | Claude Sonnet 4.6 (Anthropic) |
| Projektverantwortlicher | Sven Froehlich |

**KEINE KRITISCHEN SCHWACHSTELLEN GEFUNDEN**

# 1. Zusammenfassung

Die vierte statische Sicherheitsanalyse des NetMon-Arbeitsstands v1.1.15 durch ein Multi-Agenten-System (Claude Sonnet 4.6) bestaetigte kein einziges Finding. 1 Kandidat (K1) wurde identifiziert und in einer unabhaengigen Filterrunde nach vollstaendigem Lesen von vpn_controller.py mit Konfidenz 3/10 als False Positive ausgeschlossen.

K1 (FP): WireGuard-Konfig-Import kopiert .conf-Dateien ohne Inhaltspruefung nach /etc/wireguard/. PostUp/PreDown-Hooks werden von wg-quick als Root ausgefuehrt. Ausschluss: kein Privilege-Boundary-Crossing (Single-User-Desktop, explizite Nutzerauswahl per Datei-Dialog, bestehende sudo-Rechte vorausgesetzt). Angriffspfad erfordert entweder bereits volle Kompromittierung oder Social Engineering.

Alle 3 Findings aus Runde 3 (K1 PATH-Hijacking rkhunter, K2 fun_controller Pfad-Confinement, K3 agent/runner Pfad-Allowlist) im aktuellen Codestand als geschlossen verifiziert.

# 2. Methodik

### Phase 1 — Reconnaissance

Alle 40+ geaenderten Module gelesen; Fokus auf Subprocess-Aufrufe mit Nutzerdaten, sudo-Einbettung, URL-Handling, Dateioperationen mit unkontrollierten Pfaden, eval/exec und Privilegienuebergaenge. 20 Module mit sicherheitsrelevantem Fokus tiefgehend analysiert.

### Phase 2 — Kandidaten-Analyse

Datenflussketten von externen Quellen (WireGuard-Konfig-Dateien, API-Antworten, Nutzer-Dialoge, Dateisystem) bis zu sensitiven Operationen (sudo install, wg-quick up, subprocess.Popen) vollstaendig nachverfolgt. 1 Kandidat identifiziert (Konfidenz 7/10): vpn_controller.py — WireGuard-Import ohne Inhaltspruefung.

### Phase 3 — False-Positive-Filter

K1 in einem unabhaengigen Agenten re-analysiert: vollstaendiger Quelltext von vpn_controller.py gelesen, exakter Code der Import-Funktion (import_wg_profiles_from_folder, Zeilen 3154-3238) lokalisiert. Ausschluss: kein Privilege-Boundary-Crossing auf Single-User-Desktop, explizite Nutzerinteraktion via filedialog.askopenfilenames(), bestehende sudo-Rechte (ensure_sudo_session() gate). Konfidenz: 3/10 -> FP.

### Phase 4 — R3-Nachverfolgung

Alle 3 Findings aus Runde 3 auf Umsetzungsstand geprueft: rkhunter_manager.py (TRUSTED_WEB_CMD_PREFIXES), fun_controller.py (realpath() + fun_games_dir-Check), agent/runner.py (REPO_ROOT-Confinement fuer Pfadargumente). Alle drei im aktuellen Codestand geschlossen bestaetigt.

# 3. Bewerteter Kandidat (ausgeschlossen)

### K1 — WireGuard-Import: PostUp/PreDown-Hooks ohne Inhaltspruefung

Konfidenz 7/10 -> 3/10 -> FP

`netmon/features/vpn_controller.py — import_wg_profiles_from_folder(), Zeilen 3154–3238`

**Datenfluss:**

```
Nutzer waehlt .conf-Datei via filedialog.askopenfilenames() -> Dateiname via
re.fullmatch(r'[A-Za-z0-9_.-]+', name) validiert -> Zielpfad via wg_profile_path() +
os.path.commonpath() begrenzt -> sudo -n install -m 600 /etc/wireguard/.conf (Dateiinhalt wird
NICHT geprueft — PostUp/PreDown werden uebernommen) -> wg-quick up fuehrt PostUp-Hooks als Root
aus
```

Vorhandene Schutzmassnahmen: Filename-Regex, commonpath()-Confinement auf /etc/wireguard/, realpath()-Normalisierung. Fehlend: Inhaltspruefung auf Hook-Direktiven.

Filterentscheid: Kein Privilege-Boundary-Crossing auf einem Single-User-Desktop. Nutzer initiiert Import explizit ueber nativen Datei-Dialog (keine Automation). WireGuard-Hooks sind dokumentiertes Protokoll-Feature. Angreifer benoetigt entweder bereits Schreibzugriff auf das Dateisystem (volle Kompromittierung) oder Social Engineering (Supply-Chain-Problem, keine Code-Schwachstelle). Anwendung setzt sudo-Rechte bereits voraus. Konfidenz: 3/10 -> FALSE POSITIVE.

# 4. Sicherheitsverbesserungen seit Runde 3

**HDD Mountpoint-Validierung — hdd_controller.py — Security Fix [Commit 0431faf8]**

_slot_mountpoint() kanonikalisiert den Pfad via os.path.realpath() und prueft gegen _SAFE_MOUNT_BASES = ('/mnt/', '/media/', '/run/media/'). Ungueltige Mountpoints fallen auf sicheren Default /mnt/ zurueck. Fix verhindert Privilege Escalation via praepariertes Filesystem auf beliebigem Mountpoint.

**CWD-Entfernung aus CSV-Suchpfaden — linux_help_controller.py [Commit 363d64f5]**

os.getcwd()-Fallback aus den CSV-Suchpfaden entfernt. Verhindert Command-Injection via manipulierte linux-befehle.csv aus einem world-writable Arbeitsverzeichnis.

**EPERM-Behandlung in lock.py korrigiert [Commit 363d64f5]**

os.kill(pid, 0) mit PermissionError (EPERM) wird jetzt korrekt als 'Prozess laeuft' gewertet. Bisher fuehrte EPERM faelschlich zum Start einer zweiten Instanz.

**REPO_ROOT dynamisch aus __file__ — agent/runner.py [Commit 363d64f5]**

Hardcoded /workspaces/dev-lab durch Path(__file__).resolve().parent.parent ersetzt. Macht REPO_ROOT-Confinement in Nicht-Codespaces-Umgebungen wirksam.

**DNS-Token-Validierung: Falsy-Zero-Fix — dns_control_service.py [Commit 363d64f5]**

token.count('.') zu token.count('.') > 0 geaendert. Verhindert dass IP-Tokens ohne Punkt als gueltig durchgelassen werden.

**82 Proxy-Methoden entfernt — app.py [Commits a279e38c + 920cfc9b]**

Insgesamt 101 tote/Proxy-Methoden aus app.py geloescht. Alle via refactor_safety.py auf Null-Aufrufer verifiziert. Reduziert Angriffsflaeche.

**R3-Findings geschlossen bestaetigt (K1 rkhunter, K2 fun, K3 agent/runner)**

TRUSTED_WEB_CMD_PREFIXES in rkhunter_manager.py, realpath()-Check in fun_controller.py, REPO_ROOT-Pfad-Confinement in agent/runner.py — alle drei im aktuellen Codestand verifiziert.

# 5. Hardening-Empfehlungen — alle geschlossen

Saemtliche seit Runde 1 offenen Defense-in-Depth-Empfehlungen wurden im aktuellen Codestand v1.1.15 vollstaendig umgesetzt und per direktem Quelltext-Check verifiziert:

### 1. IP-Validierung fuer WireGuard-Endpoints [GESCHLOSSEN]

vpn_controller.py:2479 — parse_vpn_endpoint() verwendet ipaddress.ip_address(host). Nicht-IP-Hostnamen werden via _warn_non_ip_endpoint_once() protokolliert und uebersprungen. Nur validiertes endpoint_ip.compressed erreicht UFW-Kommandos.

### 2. Schema-Validierung homepage-URLs im Radio-Modul [GESCHLOSSEN]

radio_controller.py:106 — ALLOWED_RADIO_HOMEPAGE_SCHEMES = {'http', 'https'}. _normalize_station_homepage_url() (Zeilen 315-326) validiert via urlparse() und wird in open_selected_station_homepage() vor webbrowser.open() aufgerufen.

### 3. Shell-Quoting in Repository-Guard-Snippets [GESCHLOSSEN]

repository_guard_controller.py:799-802 — shlex.quote() fuer alle eingebetteten Werte: quoted_uri, quoted_path, quoted_host, quoted_keyring_path.

### 4. realpath() im Roots-Baseline-Guard [GESCHLOSSEN]

roots_baseline_guard_controller.py:123 — _normalize_guard_path_pattern() verwendet os.path.realpath(os.path.expanduser(raw)). Symlinks werden vor fnmatch-Vergleichen und Allowlist-Pruefungen vollstaendig aufgeloest.

### 5. Pfadpruefung in image_to_data_url() [GESCHLOSSEN]

ai_online_service.py:35-39 — resolved.relative_to(home_dir) erzwingt Home-Verzeichnis-Confinement. Zusaetzlich: resolve(strict=True) und Dateiendungs-Allowlist {png, jpg, jpeg, webp, gif, bmp}.

### 6. WireGuard-Import: Hook-Hinweis [GESCHLOSSEN]

vpn_controller.py:141 — _warn_wireguard_import_hooks() scannt jede .conf-Datei vor dem Import auf PreUp/PostUp/PreDown/PostDown-Direktiven und zeigt einen expliziten Nutzerhinweis mit Abbruch-Option an (3 Import-Einstiegspunkte: Zeilen 3229, 3328, 3363).

# 6. Gepruefe Sicherheitskategorien

| Kategorie | Ergebnis |
|---|---|
| Command Injection (subprocess, shell=True) | **Kein Finding** |
| Path Traversal (Dateioperationen mit Nutzereingabe) | **Kein Finding** |
| Privilege Escalation (sudo mit unkontrollierten Argumenten) | **Kein Finding** |
| Argument Injection (list-form subprocess mit externen Daten) | **Kein Finding** |
| Unsichere Deserialisierung (pickle, yaml.load, eval) | **Kein Finding** |
| Hardcoded Credentials / API-Keys | **Kein Finding** |
| Sensitive Data Exposure (PII in Logs) | **Kein Finding** |
| SSRF via urllib.request / externe URL-Weitergabe | **Kein Finding** |
| URL-Schema-Validierung an Media-Subprocessen | **Kein Finding** |
| Symlink-Angriffe auf privilegierte Dateioperationen | **Kein Finding** |
| Mountpoint-Manipulation via Konfig-Datei | **Kein Finding (Fix v1.1.15)** |
| WireGuard-Hook-Ausfuehrung via Import | K1 -> FP (kein Boundary Crossing) |
| PATH-Hijacking via shutil.which (R3-K1) | Geschlossen (R3) |
| Agent-CLI Pfad-Confinement (R3-K3) | Geschlossen (R3) |

# 7. Analysierte Module (Runde 4 — Sicherheitsfokus)

| Datei / Modul | Fokus | Ergebnis |
|---|---|---|
| `netmon/features/vpn_controller.py` | WireGuard-Import, wg-quick, Interface-Validation | K1 -> FP |
| `netmon/features/hdd_controller.py` | Mountpoint-Validierung, LUKS, sudo mount | Kein Finding |
| `netmon/features/linux_help_controller.py` | CSV-Suchpfade nach CWD-Entfernung | Kein Finding |
| `netmon/features/firewall_controller.py` | UFW, Port-Regex, ipaddress-Validierung | Kein Finding |
| `netmon/features/clamav_controller.py` | ClamAV, _safe_makedirs_no_symlink() | Kein Finding |

| | | |
|---|---|---|
| `netmon/features/radio_controller.py` | mpv/ffmpeg, ALLOWED_RADIO_STREAM_SCHEMES | Kein Finding |
| `netmon/features/packet_capture_controller.py` | tshark, Interface-Name aus ip link | Kein Finding |
| `netmon/features/security_audit_controller.py` | Lynis/rkhunter, Startup-Guard-Baseline | Kein Finding |
| `netmon/features/security_monitoring_controller/_core.py` | Event-System, Baseline, Shared Helpers | Kein Finding |
| `netmon/features/security_monitoring_controller/_scanning.py` | Geraete-/Integritaets-Scanning | Kein Finding |
| `netmon/features/dot_controller.py` | DNS-Dropin via sudo install, Pfad hardcoded | Kein Finding |
| `netmon/features/chad_controller.py` | KI-Integration, getattr-Fix, kein Subprocess | Kein Finding |
| `netmon/core/lock.py` | EPERM-Behandlung, Instanz-Lock | Kein Finding |
| `netmon/core/command.py` | run_cmd-Wrapper, sudo -S -v, shell=False | Kein Finding |
| `netmon/services/dns_control_service.py` | nmcli con mod, DNS-Token-Validierung | Kein Finding |
| `netmon/services/startup_guard_service.py` | SHA-256-Hashing, neue-Dateien-Reporting | Kein Finding |
| `netmon/services/path_guard_service.py` | PATH-Analyse, read-only | Kein Finding |
| `agent/runner.py` | REPO_ROOT-Fix, Subprocess-Allowlist, Pfad-Confinement | Kein Finding |
| `agent/tools.py` | Dateilesen mit relative_to(REPO_ROOT)-Guard | Kein Finding |
| `netmon/app.py (Refaktorierung)` | 82 Proxy-Methoden entfernt, keine Regression | Keine Regression |

---

## Impressum und Haftungsausschluss

**Auditor:** Claude Sonnet 4.6 (Anthropic) — Multi-Agenten-System + False-Positive-Filterrunde

**Projektverantwortlicher:** Sven Froehlich

**Datum:** 29. Juni 2026

**Codestand:** e334324a … 920cfc9b + Arbeitsstand (modified/untracked), 40+ Dateien — Runde 4

**Methode:** Statische Code-Analyse + unabhaengige False-Positive-Filterrunde — kein Laufzeit-Penetrationstest

Dieses Audit ersetzt keinen professionellen Penetrationstest durch eine zertifizierte Sicherheitsfirma. Die Analyse beschraenkt sich auf den angegebenen Codestand. Nicht erfasst: Laufzeitverhalten, Systembibliotheken,

Netzwerkkonfiguration.