SICHERHEITS-AUDIT — RUNDE 2

NetMon Roots v1.1.14

Code Security Review — Juni 2026

Datum	28. Juni 2026
Analysierte Version	v1.1.14 — Runde 2
Geaenderte / neue Dateien	100+
Neue / revidierte Module	25+
Kritische Findings	0
Findings gesamt	0 (alle 4 Kandidaten ausgeschlossen)
Bewertete Kandidaten	4
Konfidenz-Schwelle	>= 8 / 10
Methode	Statische Code-Analyse + manuelle Verifikation
Analysewerkzeug	Claude Sonnet 4.6 (Anthropic)
Projektverantwortlicher	Sven Froehlich

KEINE KRITISCHEN SCHWACHSTELLEN GEFUNDEN

1. Zusammenfassung

Die zweite statische Sicherheitsanalyse des NetMon-Arbeitsstands v1.1.14 (100+ Dateien, 25+ neue bzw. revidierte Module) durch ein automatisiertes Multi-Agenten-System auf Basis von Claude Sonnet 4.6 ergab keine ausnutzbaren Sicherheitsluecken. Alle vier Kandidaten wurden als False Positives ausgeschlossen.

Besonderer Befund: K1 (SSRF-Verdacht im Radio-Modul) wurde initial mit Konfidenz 8/10 als bestaetigt eingestuft, nach manueller Datenfluss-Verifikation jedoch widerlegt: _normalize_station_stream_url() mit ALLOWED_RADIO_STREAM_SCHEMES = {'http', 'https', 'rtsp', 'rtmp'} (Zeile 103) validiert jede URL bereits vor der Aufnahme ins Station-Dict. mpv und urllib.request empfangen ausschliesslich vorvalidierte URLs.

Erstmalig analysiert: radio_controller.py (5 531 Zeilen), ai_analysis_service.py, lang_service.py, network_helpers.py, coordinator.py, state.py, fun_controller.py, speedtest_controller.py, dot_controller.py, security_audit_controller.py sowie neue UI-Tabs. Regressionscheck: 82 Proxy-Methoden aus app.py entfernt — keine Regression.

Sicherheitsverbesserungen seit Runde 1: (1) ClamAV-Tempdir-Symlink-Schutz _safe_makedirs_no_symlink() implementiert. (2) URL-Schema-Validierung im Radio-Modul bereits vollstaendig vorhanden. (3) 82 Proxy-Methoden aus app.py entfernt (Angriffsflaechenreduktion).

2. Methodik

Phase 1 — Reconnaissance

Lesen aller neuen und geaenderten Module; Fokus auf externe API-Anbindungen, Subprocess-Aufrufe mit externen Daten, URL-Handling, Datei-Operationen mit Nutzereingabe, Deserialisierung.

Phase 2 — Kandidaten-Analyse

Datenflussketten von externen Quellen bis zu sensitiven Operationen nachverfolgt. Besonderes Augenmerk: URL-Schemata an mpv, ffmpeg, urllib.request und webbrowser.open().

Phase 3 — False-Positive-Filter

Jeder Kandidat in einem unabhaengigen Agenten re-analysiert. Ausschluss: DoS, Single-User-Desktop ohne Fremd-Input, stdout/stderr=DEVNULL + Re-Encoding verhindert Exfil, Browser-Sandbox.

Phase 4 — Manuelle Verifikation (K1)

Zusaetzliche Nachpruefung des Radio-Kandidaten durch direktes Lesen der Funktionskette: normalize_station_entry() -> _normalize_station_stream_url() -> ALLOWED_RADIO_STREAM_SCHEMES. Kandidat als False Positive widerlegt.

Konfidenz-Schwelle: Nur Findings mit Score >= 8/10 waeren aufgenommen worden.

3. Neu analysierte Module (Runde 2)

Datei / Modul	Fokus	Ergebnis
`netmon/features/radio_controller.py`	mpv, ffmpeg, radio-browser.info API, ICY-Fetch, URL-Schema-Validierung	Kein Finding
`netmon/features/chad_controller.py`	KI-Integration, getattr-Muster, Datei-Import	Kein Finding
`netmon/features/fun_controller.py`	Subprocess-Aufrufe, Paketpruefung	Kein Finding
`netmon/features/speedtest_controller.py`	Speedtest-Subprocess, Ergebnis-Parsing	Kein Finding
`netmon/features/dot_controller.py`	Dropin-Generierung, systemd-Konfiguration	Kein Finding
`netmon/features/security_audit_controller.py`	Lynis/rkhunter, Startup-Guard-Baseline-Lesen	Kein Finding
`netmon/services/ai_analysis_service.py`	KI-Analyse-Pipeline, Datenuebertragung	Kein Finding
`netmon/services/lang_service.py`	Lokalisierung, JSON-Laden	Kein Finding
`netmon/services/network_helpers.py`	Netzwerk-Hilfsfunktionen, IP-Parsing	Kein Finding
`netmon/core/coordinator.py + state.py`	Koordinator, globaler Zustand	Kein Finding
`netmon/ui/tabs/radio_tab.py`	Radio-UI, Benutzerinteraktion	Kein Finding
`netmon/ui/tabs/ (cleanup, download, fun, chad_settings)`	Neue UI-Tabs, Dateioperationen	Kein Finding
`scripts/refactor_safety.py`	AST-Analyse, Caller-Check, Read-Only-Tool	Kein Finding
`netmon/app.py (Refaktorierung)`	82 Proxy-Methoden entfernt	Keine Regression

4. Bewertete Kandidaten (alle ausgeschlossen)

Alle vier Kandidaten wurden als False Positives ausgeschlossen. K1 wurde initial mit Konfidenz 8/10 als bestaetigt gewertet und anschliessend durch manuelle Datenfluss-Verifikation widerlegt: _normalize_station_stream_url() mit ALLOWED_RADIO_STREAM_SCHEMES = {'http', 'https', 'rtsp', 'rtmp'} (Zeile 103) validiert jede URL bevor sie ins Station-Dict uebernommen wird.

K1 — Radio-URL an mpv / urllib.request — manuell widerlegt

Konfidenz: 8/10 -> FP (nach manueller Verifikation)

`Datei: netmon/features/radio_controller.py, Zeilen 103, 298-309, 4835`

Initiale Bewertung: Radio-URLs aus radio-browser.info-API gelangen ohne Schema-Validierung an mpv (Zeile 4835) und urllib.request.urlopen() (Zeile 3150). Erstes Filter-Ergebnis: Konfidenz 8/10.

> Filterentscheid: WIDERLEGT: _normalize_station_stream_url() (Zeile 298) wird von normalize_station_entry() (Zeile 317) aufgerufen und prueft das URL-Schema via urllib.parse.urlparse() gegen ALLOWED_RADIO_STREAM_SCHEMES = {'http', 'https', 'rtsp', 'rtmp'} (Zeile 103). Nicht erlaubte Schemata (file://, lavfi://, etc.) geben '' zurueck; Station wird als None verworfen. mpv und urllib.request empfangen ausschliesslich vorvalidierte URLs.

K2 — webbrowser.open mit nicht validierter API-URL

Konfidenz: 3/10

`Datei: netmon/features/radio_controller.py, Zeile 5197`

webbrowser.open(url, new=2) wird mit dem homepage-Feld aus der API-Antwort ohne Schema-Validierung aufgerufen.

> Filterentscheid: AUSGESCHLOSSEN: Moderne Browser erzwingen strikte Same-Origin-Policy fuer file://-URLs — keine Exfiltration moeglich. javascript:-URIs werden als Top-Level-URL nicht ausgefuehrt.

K3 — ffmpeg -i mit nicht validierter API-URL

Konfidenz: FP

`Datei: netmon/features/radio_controller.py, Zeile 2160`

_recording_command(stream_url, output_path) uebergibt die Station-URL als 'ffmpeg -i '-Argument. Zusaetzlich: URL bereits durch _normalize_station_stream_url() vorvalidiert.

> Filterentscheid: AUSGESCHLOSSEN: (1) URL durch _normalize_station_stream_url() vorvalidiert. (2) stdin/stdout/stderr nach DEVNULL; libmp3lame-Transcoding verhindert Exfil beliebiger Binaerdaten. Kein Exfiltrationspfad.

K4 — Dateilese-Operation via Startup-Guard-Baseline

Konfidenz: FP

`Datei: netmon/features/security_audit_controller.py, Zeilen 1173-1184`

format_startup_guard_entry_details() oeffnet beliebige Dateipfade aus einer nutzerschreibbaren JSON-Baseline-Datei.

Filterentscheid: AUSGESCHLOSSEN: Single-User-Desktop — kein Privilege-Boundary-Crossing. Baseline im Konfig-Verzeichnis des Nutzers (nur von diesem schreibbar). Angreifer mit Baseline-Zugriff hat bereits vollen Lesezugriff auf alle open()-erreichbaren Dateien.

5. Sicherheitsverbesserungen seit Runde 1

Die folgenden Hardening-Empfehlungen aus Audit Runde 1 wurden im aktuellen Codestand umgesetzt.

ClamAV Tempdir: Symlink-Schutz implementiert — clamav_controller.py

Runde 1 empfahl, os.makedirs(exist_ok=True) durch eine Sequenz zu ersetzen, die Symlinks vor dem Anlegen des Verzeichnisses prueft (Punkt 4). Umgesetzt als _safe_makedirs_no_symlink(path) (Zeile 71): prueft via os.lstat(), wirft RuntimeError bei Symlink. Empfehlung geschlossen.

URL-Schema-Validierung im Radio-Modul — radio_controller.py

Das Radio-Modul enthaelt bereits eine vollstaendige Schema-Allowlist: ALLOWED_RADIO_STREAM_SCHEMES = {'http', 'https', 'rtsp', 'rtmp'} (Zeile 103), durchgesetzt durch _normalize_station_stream_url() (Zeile 298-309) via urllib.parse.urlparse(). Nicht erlaubte Schemata werden vor der Speicherung im Station-Dict abgelehnt — mpv und urllib.request empfangen nur validierte URLs.

Angriffsflaechenreduktion: 82 Proxy-Methoden aus app.py entfernt

Im Zuge der God-Object-Extraktion wurden 82 Proxy-Methoden aus app.py geloescht, deren Controller-Gegenstuecke in netmon/features/ vorhanden waren. Alle Methoden vorab per refactor_safety.py auf Null-Aufrufer verifiziert. Commit 920cfc9b. Reduzierung von 34 294 auf 34 005 Zeilen.

6. Hardening-Empfehlungen (keine kritischen Bugs)

Die folgenden Punkte sind keine Sicherheitsluecken, sondern Defense-in-Depth-Massnahmen. Punkte 1-5 aus Runde 1 unveraendert offen.

1. Git-Subcommand-Allowlist in agent/runner.py

Sobald LLM-Output in run_command() fliesst, Subcommand-Allowlist ergaenzen (status, diff, log, show). Derzeit latentes Architektur-Risiko.

2. IP-Validierung fuer WireGuard-Endpoints

parse_vpn_endpoint() sollte den Host-Teil mit ipaddress.ip_address() validieren bevor er in UFW-Kommandos eingebettet wird.

3. Pfadpruefung in image_to_data_url()

Optional: Dateipfad auf das Benutzer-Homeverzeichnis einschraenken. Aktuell kein ausnutzbarer Angriffspfad vorhanden.

4. Schema-Validierung fuer homepage-URLs im Radio-Modul

webbrowser.open(station['homepage']) validiert das Schema nicht. Analoge Allowlist-Pruefung als defensive Best Practice empfohlen.

5. Shell-Quoting in Repository-Guard-Snippets

URI- und Pfadwerte in get_repository_guard_quick_action_snippet() mit shlex.quote() maskieren.

6. os.path.realpath() im Roots-Baseline-Guard

_normalize_guard_path_pattern() sollte os.path.abspath durch os.path.realpath() ersetzen, um Symlinks vor dem fnmatch-Check aufzuloesen.

7. Geprufte Sicherheitskategorien

Kategorie	Ergebnis
Command Injection (subprocess, shell=True)	Kein Finding
Path Traversal (Dateioperationen mit Nutzereingabe)	Kein Finding
Privilege Escalation (sudo, Argument-Injektion)	Kein Finding
Unsichere Deserialisierung (pickle, yaml.load)	Kein Finding
Eval/Exec mit Nutzereingabe	Kein Finding
Hardcoded Credentials / API-Keys	Kein Finding
Authentication Bypass / Privilege Escalation	Kein Finding
Sensitive Data Exposure (PII in Logs)	Kein Finding
Externe API-Aufrufe mit unkontrollierten Daten	Kein Finding
SSRF via urllib.request / externe URL-Weitergabe	Kein Finding
Kryptographische Schwachstellen	Kein Finding
Symlink-Angriffe auf privilegierte Dateioperationen	Kein Finding
Clipboard-Injection / Snippet-Generierung ohne Auto-Exec	Kein Finding
URL-Schema-Validierung an Media-Subprocess	Kein Finding

Impressum und Haftungsausschluss

Auditor: Claude Sonnet 4.6 (Anthropic) — Automatisiertes Multi-Agenten-Sicherheits-Review + manuelle Verifikation

Projektverantwortlicher: Sven Froehlich

Datum: 28. Juni 2026

Analysierter Codestand: e334324a ... 920cfc9b + Arbeitsstand (unstaged), gesamt 100+ Dateien — Runde 2

Methode: Statische Code-Analyse + manuelle Datenfluss-Verifikation — kein Laufzeit-Penetrationstest