SICHERHEITS-AUDIT — RUNDE 10

NetMon Roots v1.1.20

Code Security Review — Juli 2026

Datum	19. Juli 2026
Analysierte Version	v1.1.20
Schwerpunkt	Security-Baseline-/Sysctl-Apply-Stufe, Baseline-Transferpaket, UFW-Locale, Test-Infrastruktur
Kritische Findings	0
Hohe Findings	2 (beide behoben & verifiziert)
Mittlere Findings	0
Haertungen umgesetzt	2 + 15 neue Regressionstests
Gesamt-Testsuite	2133 bestanden (3 bekannte Umgebungsfehler)
Analysewerkzeug	Claude Fable 5 (Anthropic)
Projektverantwortlicher	Sven Froehlich

0 KRITISCHE — 2 HOHE — 0 MITTLERE FINDINGS (BEIDE UMGESETZT & VERIFIZIERT)

1. Zusammenfassung

Die zehnte Sicherheitsanalyse pruefte den kompletten Code-Diff seit dem Runde-9-Stand — rund 15.300 neue Zeilen, ueberwiegend die neue Security-Baseline-/Lynis-Funktionalitaet, die neue Sysctl-Apply-Stufe samt Baseline-Transferpaket (Export/Import zwischen Rechnern), die locale-unabhaengige UFW-Auswertung, das neue Daten-Backup und die neuen, rein lesenden System-Checks (Home-/Dateirechte, Secure Boot, APT-Sicherheitsupdates). Ergaenzend im Fokus: die privilegierten Senken (sudo -n, pkexec, root-netmon_network_service) und die Test-/Entwicklungsinfrastruktur.

Diese Runde brachte zwei belastbare Schwachstellen hohen Schweregrads. Beide wurden direkt umgesetzt, mit Regressionstests abgesichert und verifiziert. Keine kritischen, keine remote-ausloesbaren Befunde.

F1 (HIGH, Konfidenz 9/10, behoben): Das neue Baseline-Transferpaket — eine per Dateidialog importierte JSON-Datei, gedacht zum Teilen gehaerteter Profile zwischen Rechnern — reichte die Felder id/key/expected ohne Zeichen- oder Whitelist-Pruefung bis in eine Datei unter /etc/sysctl.d/ durch, die als root per 'sudo -n sysctl --system' geladen wird. Eine praeparierte Datei mit id=sysctl.kernel.core_pattern und expected=|/tmp/evil.sh erzeugte die Zeile 'kernel.core_pattern = |/tmp/evil.sh' — der Kernel fuehrt danach bei jedem Prozess-Crash das Programm als root aus (klassische Rechteausweitung). Eingebettete Zeilenumbrueche ueberlebten das .strip() und erlaubten das stille Herabsetzen der Kernel-Haertung. Ein dedizierter Validierungs-Sub-Agent bestaetigte das mit Konfidenz 9/10 und lauffaehigem Repro. Gehaertet: importierte Sysctl-Eintraege werden jetzt gegen eine Allowlist der zehn dokumentierten Baseline-Schluessel plus ein strenges Schluessel-/Wertformat geprueft (keine Zeilenumbrueche, Pipes, Shell-Metazeichen); die Drop-in-id wird nur aus dem validierten Schluessel abgeleitet; eine zweite Verteidigungslinie filtert direkt an der Schreibsenke.

F2 (HIGH, empirisch entdeckt, behoben): Zwei Tests fuer die neuen Systemaktionen (Reboot/Shutdown/Logout) liessen einen Kommando-Generator das echte ['systemctl','reboot'] zurueckgeben, ohne run_cmd bzw. den ausfuehrenden Job-Thread zu mocken. Da ein aktiver Desktop-Nutzer ueber polkit ohne Passwort neu starten darf, fuehrte jeder Lauf der Testsuite auf einem Desktop-System zu einem tatsaechlichen Neustart des Rechners. Dieses Finding wurde nicht statisch, sondern empirisch entdeckt: Der Audit-Host startete beim Testlauf reproduzierbar neu; die Boot-Zeitstempel deckten sich exakt mit den Testlaeufen. Betroffen ist die Test-/Entwicklungs- und CI-Infrastruktur, nicht der ausgelieferte Anwendungscode (dessen Release-Pfad durch Bestaetigungsdialog, Sicherungsphrase und Strict-Protection abgesichert ist). Gehaertet: beide Tests brechen vor dem Job-Thread ab und nutzen inerte Stubs; zusaetzlich blockiert ein suiteweiter Guard (tests/conftest.py) destruktive Kommandos hart auf subprocess-Ebene — auch hinter sudo/pkexec-Wrappern.

Ergaenzend: Eine latente Shell-Injektion ueber das aus dem importierten Schluessel gebildete 'sysctl -n {key}'-Kommando-Template (aktuell nicht erreichbar) wird durch die strenge Schluessel-Pruefung aus F1 automatisch mitentschaerft. Die Game-Mode-UFW-Ausnahmen sind kein ausnutzbarer Befund (argv-Token, ufw lehnt Ungueltiges ab). Die Kindersicherung bleibt als bewusste Design-Entscheidung dokumentiert; die postinst-DNS-Policy (8.8.8.8) und die Nur-Hash-Schnellpruefung auf der Downloadseite bleiben als Runde-9-Hinweise offen. Alle Runde-7/8/9-Haertungen sind intakt; Versionskonsistenz (App, deb, Changelog) ist durchgaengig 1.1.20.

2. Methodik

Phase 1 - Diff-Recon

Vollstaendige, datei-weise Sichtung des Branch-Diffs seit dem Runde-9-Stand (Commit ac412cbc). Versionskonsistenz-Abgleich (App = deb = Changelog = 1.1.20). Suche nach neuen Datenfluessen von Fremdeingaben (importierte Dateien, geparster Kommando-Output) zu privilegierten Senken.

Phase 2 - Kandidaten-Analyse

Ein Analyse-Sub-Agent verfolgte importierte und geparste Daten bis zu sudo/pkexec-Aufrufen und Datei-Schreibpfaden. Drei Kandidaten identifiziert — einer belastbar (Sysctl-Injektion), zwei als Defense-in-Depth (latente command-Injektion, Game-Mode-Port).

Phase 3 - False-Positive-Validierung

Der belastbare Kandidat wurde von einem eigenen Validierungs-Sub-Agenten mit lauffaehigem Repro gegengeprueft: Konfidenz 9/10, HIGH. Ein Detail-Ueberclaim (angeblich angreiferkontrollierter Schreibpfad) wurde dabei widerlegt — der Zielpfad ist fix; fuer die Ausnutzbarkeit irrelevant, da der feste Pfad ohnehin als root geladen wird.

Phase 4 - Haertung & Regressionscheck

Beide HIGH-Findings wurden umgesetzt und mit 15 neuen Regressionstests abgesichert (8 fuer die Sysctl-Allowlist, 7 fuer den Kommando-Guard). Gesamtsuite erneut ausgefuehrt: 2133 bestanden. Die drei verbleibenden Fehler sind bekannte, umgebungsbedingte Faelle ohne Sicherheitsbezug (installiertes Radio-Desktop-File gewinnt gegen den Laufzeitpfad; echtes proton-de-VPN-Profil des Hosts).

3. Bestaetigte Findings

F1 — Sysctl-Direktiven-Injektion ueber importiertes Baseline-Transferpaket [HIGH, Konfidenz 9/10, behoben]

`netmon/features/security_audit_controller.py — _sanitize_transfer_sysctl_entries() / _build_sysctl_dropin_content() / apply_security_baseline_sysctl() / import_security_baseline_transfer().`

Das neue Baseline-Transferpaket (per Dateidialog importierte JSON, gedacht zum Teilen gehaerteter Profile zwischen Rechnern) durchlief eine Sanitisierung, die nur auf nicht-leer prueft — ohne Zeichen- oder Whitelist-Validierung von id/key/expected. Die Werte wurden als 'key = value' in eine Datei unter /etc/sysctl.d/ geschrieben und per 'sudo -n sysctl --system' als root geladen. Eine praeparierte Datei mit id=sysctl.kernel.core_pattern und expected=|/tmp/evil.sh erzeugte 'kernel.core_pattern = |/tmp/evil.sh' — der Kernel fuehrt danach bei jedem Prozess-Crash das Programm als root aus (Rechteausweitung). Eingebettete Zeilenumbrueche ueberlebten das .strip() und erlaubten mehrere eingeschleuste Direktiven bzw. das stille Herabsetzen der Kernel-Haertung, waehrend die UI 'Haertung angewendet' meldete. Kein Zero-Click (Import + Bestaetigung + sudo-Session noetig), aber ein realistischer Social-Engineering-Pfad, da Transferpakete explizit zum Teilen gedacht sind. Anders als die parallele UFW-Validierung (ValueError bei ungueltigen Regeln) filterte der Sysctl-Pfad nichts.

Behebung: Gehaertet & verifiziert: Neue Allowlist SECURITY_BASELINE_ALLOWED_SYSCTL_KEYS (zehn dokumentierte Schluessel, siehe docs/security/KERNEL_SYSCTL.md) plus Format-Regex fuer Schluessel und Werte. _sanitize_transfer_sysctl_entries() lehnt unerlaubte Schluessel und ungueltige Werte (Zeilenumbrueche, Pipes, Shell-Metazeichen) per ValueError ab und leitet die Drop-in-id nur aus dem validierten Schluessel ab, nie aus dem freien id-Feld. Zweite Verteidigungslinie in _build_sysctl_dropin_content() filtert direkt an der Schreibsenke. Mit acht neuen Regressionstests abgesichert.

F2 — Testsuite fuehrte echtes 'systemctl reboot' aus (Host-Neustart) [HIGH, empirisch entdeckt, behoben]

`tests/test_netmon_diagnostics.py (run_system_action_async-Tests); netmon/app.py (_system_action_commands / Job-Thread).`

Zwei Tests fuer die neuen Systemaktionen (Reboot/Shutdown/Logout) prueften die Bestaetigungsphrase, stubbten aber den Kommando-Generator so, dass er das echte ['systemctl','reboot'] zurueckgab — ohne run_cmd bzw. den ausfuehrenden Job-Thread zu mocken. Da ein aktiver Desktop-Nutzer ueber polkit ohne Passwort neu starten darf, fuehrte jeder Lauf der Testsuite auf einem Desktop-System zu einem tatsaechlichen Neustart des Rechners. Empirisch entdeckt: Der Audit-Host startete beim Testlauf reproduzierbar neu; die Boot-Zeitstempel deckten sich exakt mit den Testlaeufen. Einordnung: betrifft die Test-/Entwicklungs- und CI-Infrastruktur, nicht den ausgelieferten Anwendungscode (dessen Release-Pfad durch Bestaetigungsdialog, Sicherungsphrase AUSFUEHREN und Strict-Protection abgesichert ist). Wirkung dennoch hoch: unerwarteter Neustart mit Risiko fuer ungespeicherte Arbeit auf jeder Maschine, die die Suite ausfuehrt.

Behebung: Gehaertet & verifiziert: Beide Tests brechen jetzt vor dem Job-Thread ab und verwenden nur inerte Stub-Kommandos. Zusaetzlich blockiert ein suiteweiter Guard (tests/conftest.py) destruktive Kommandos (systemctl reboot/poweroff/halt, shutdown, reboot, loginctl terminate-*) hart auf subprocess-Ebene — auch hinter sudo/pkexec/env-Wrappern und in Shell-String-Form. Mit sieben neuen Regressionstests (test_destructive_command_guard.py) abgesichert.

4. Weitere Beobachtungen & Design-Entscheidungen

Latente Shell-Injektion ueber das command-Feld — mitbehoben

Der importierte Sysctl-Schluessel floss auch in ein 'sysctl -n {key}'-Kommando-Template. Aktuell nicht erreichbar (die Auswertung nutzt die statische Baseline-Liste), aber ein spaeteres Refactoring haette daraus eine Kommando-Injektion machen koennen. Durch die neue strenge Schluessel-Validierung aus F1 automatisch mitentschaerft.

Game-Mode-UFW-Ausnahmen — kein ausnutzbarer Befund

Der Port des Spielmodus wird nicht per Regex vorgeprueft, geht aber als einzelnes argv-Token an 'sudo -n ufw allow ...' (kein Shell-Parsing; ufw lehnt Ungueltiges ab); der benutzerdefinierte Pfad validiert bereits streng (isdigit, 1-65535, Protokoll-Whitelist). Nur zur Vollstaendigkeit notiert.

Kindersicherung — weiterhin bewusste Design-Entscheidung

Die PBKDF2-gestuetzte Kindersicherung im Secure-Browser bleibt wie in Runde 8 dokumentiert: Alltagshuerde gegen Gelegenheitszugriff im selben Konto, keine harte Grenze gegen technisch versierte Nutzer mit Kontozugriff. Keine Codeaenderung, unveraendert gueltig.

postinst-DNS & Nur-Hash-Schnellpruefung — Policy-Hinweise offen

Die beiden Policy-Hinweise aus Runde 9 bleiben bestehen: Das root-postinst biegt aktive NM-DNS bei der Installation auf 8.8.8.8 (Opt-in empfohlen), und die Schnellpruefung auf der Downloadseite empfiehlt nur sha256sum -c statt der minisign-Vollpruefung. Kein ausnutzbarer Befund — als Design-Entscheidungen des Projektverantwortlichen offen.

5. Umgesetzte Haertungen (Runde 10)

Sysctl-Allowlist + Wert-Validierung fuer importierte Transferpakete — security_audit_controller.py

Neue Konstante SECURITY_BASELINE_ALLOWED_SYSCTL_KEYS (zehn dokumentierte Schluessel) plus Format-Regex fuer Schluessel und Werte. _sanitize_transfer_sysctl_entries() lehnt unerlaubte Schluessel und ungueltige Werte per ValueError ab und leitet die id nur aus dem validierten Schluessel ab; _build_sysctl_dropin_content() filtert als zweite Verteidigungslinie an der Schreibsenke.

Suiteweiter Guard gegen destruktive Systemkommandos — tests/conftest.py

Ein subprocess.Popen-Wrapper blockiert systemctl reboot/poweroff/halt, shutdown, reboot, poweroff, halt, telinit und loginctl terminate-* — auch hinter sudo/pkexec/env-Wrappern und in Shell-String-Form. Kein Test kann je wieder ein echtes destruktives Kommando bis zum System durchreichen.

15 neue Regressionstests

Acht Tests in test_netmon_lynis_output.py fuer die Sysctl-Allowlist (Ablehnung von core_pattern, Pipe-/Newline-Werten, id-Ableitung, End-to-End-Import-Refusal) und sieben in test_destructive_command_guard.py fuer den Kommando-Guard. Gesamtsuite: 2133 bestanden.

6. Geprüfte Sicherheitskategorien

Kategorie	Ergebnis
Direktiven-Injektion in privilegierte Konfig (/etc/sysctl.d/)	**F1 -> behoben (Allowlist+Wert-Check)**
Destruktive Kommandos in Test-/CI-Infrastruktur	**F2 -> behoben (Guard+Tests)**
Command-/Argument-Injection (ufw/nmcli/wg-quick, subprocess)	**list-form; command-Feld mitbehoben**
Locale-unabhaengige UFW-/Lynis-Auswertung (Output-Parsing)	**Kein Finding (Anzeige-Logik)**
Native Rechte-Checks (Home/kritische Dateien, Symlink/TOCTOU)	**Kein Finding (lstat, kein Follow)**
Root-Service (netmon_network_service, pkexec)	**Kein neuer Endpunkt, argv-validiert**
Daten-Backup (Symlink-Escape, Rekursion, Pfadgrenzen)	**Kein Finding (kein Follow, $HOME)**
Repository-Guard (Keyring-/URI-Parsing, Quick-Actions)	**Kein Finding (shlex.quote)**
Unsichere Deserialisierung (pickle, yaml.load, eval, shell=True)	**Kein Finding**
Autorisierung / Kindersicherung	**Bewusste Design-Entscheidung**
Regression der Runde-7/8/9-Haertungen	**Alle intakt**
Versionskonsistenz (App / deb / Changelog)	**Durchgaengig 1.1.20**

Impressum und Haftungsausschluss

Auditor: Claude Fable 5 (Anthropic) — Analyse-Sub-Agent + dedizierter False-Positive-Validierungs-Sub-Agent (mit Repro)

Projektverantwortlicher: Sven Froehlich

Datum: 19. Juli 2026

Codestand: Arbeitsstand v1.1.20 (Diff seit dem Runde-9-Stand ac412cbc), einschliesslich der neuen Security-Baseline-/Sysctl-Apply-Stufe und des Baseline-Transferpakets — Runde 10

Methode: Statische Code-Analyse + manuelle Datenfluss-Verifikation des kompletten Branch-Diffs + Konfidenz-gescorte FP-Validierung mit lauffaehigem Repro + Umsetzung von zwei Haertungen mit 15 Regressionstests + Regressions-Gegenpruefung der Vorrunden-Haertungen (Gesamtsuite 2133 bestanden, drei bekannte umgebungsbedingte Fehler ohne Sicherheitsbezug) — 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.