MrRowie · Modul MrsNanny

Pull, nicht Push.
MrsNanny

Das Wachsamkeits-Modul von MrRowie: modulares Health-Monitoring für Kundenstandorte — einmal ausrollen, per Script-Baukasten erweitern, täglich ein KI-Statusreport in Klartext. Ohne Cloud, ohne Agent-Software von der Stange.

Bash + Python
KEIN FRAMEWORK-BALLAST

Restricted SSH
PULL STATT PUSH

Lokales LLM
REPORT OHNE CLOUD

16 Collectoren
BELIEBIG ERWEITERBAR

DER BAUKASTEN

Alles an Bord: Sensoren, Wartung, Berichte

Die Plattform ist ein Baukasten aus kleinen, klar geschnittenen Bausteinen — jeder einzeln verteilbar, zusammen ein rundes Bild vom Zustand der Flotte. Das ist an Bord:

16 Collectoren + 4 Scanner

Messen täglich den Ist-Zustand jeder Node — jede Zahl mit Ampel-Selbstbewertung, jederzeit per Script erweiterbar.

- speedtest
- disk
- updates
- reboot
- services
- temperature
- wazuh-agent
- patchmon
- replication
- nextcloud
- wordpress
- openarchiver
- pangolin-stack
- autosnapshot
- system
- dockhand
- fancontrol
- semaphore
- apt-lock
- kvm-spin

Wartung, die mitläuft

Nicht nur melden, sondern beheben: automatische Security-Updates flottenweit und wöchentliche Nextcloud-Pflege samt täglichem Status-Snapshot — auch für mehrere Instanzen je Host.

- unattended-upgrades
- occ-Wartung
- app:update --all
- Core patch/minor
- Multi-Instanz

Berichte, Checks & Alarme

Stündlicher Sofort-Alarm autark auf der Node, morgens der LLM-Tagesreport je Kunde — dazu GF-Monatsbericht, Report-Archiv, Flotten-Webübersicht und NOC-View.

- Sofort-Alarm :30
- Tagesreport 07:10
- GF-Monatsbericht
- Flotten-Web
- Archiv 400 Tage

WORUM ES GEHT

Ein dünner Sensor auf der Node, die Intelligenz am Master

MrsNanny beantwortet jeden Tag eine einfache Frage: *„Wie geht es meinen Maschinen?"* — als Mail in verständlichem Deutsch, formuliert von einem lokal laufenden Sprachmodell, mit Ampel-Verdict pro Kunde.

Auf der überwachten Node läuft bewusst wenig: ein Runner (`export.sh`), der eine Handvoll Collector-Scripts einsammelt und deren Ergebnis als ein JSON bereitlegt. Alles Schlaue — Bewertung, KI-Zusammenfassung, HTML-Mail, Trend-Historie — passiert zentral am Master. Die Node braucht weder VPN noch LLM noch offene Ports.

Pull, nicht Push

Der Master zieht die Daten per restricted SSH-Key (forced command). Die Kunden-Node baut nie eine Verbindung nach außen auf — Vertrauen fließt nur vom Master zur Node.

Dünner Agent, schlauer Master

Die Node ist ein passiver Sensor. KI-Report, Verdict und Versand sitzen zentral — die Ollama-Last bleibt kontrolliert, kein LLM und kein Mail-Template auf der Node.

Sensoren bewerten sich selbst

Jeder Collector liefert seine eigene Ampel (green/yellow/red) mit. Der Master muss neue Sensoren nicht kennen — sie erscheinen einfach im nächsten Report.

ARCHITEKTUR

Zwei Hälften, eine Richtung

Kunden-Node /opt/monitoring-agent · User servernanny

- `export.sh` — Runner: Collectoren → Gesamt-JSON
- `collectors.d/` — die Sensoren (Liste unten)
- `scanners.d/` — Zustands-Checks, erster live
- `measure.sh` — Speedtest 4×/Tag per Cron
- `alert.sh` — Sofort-Alarm, autark via msmtp
- `agent.env` — Schwellwerte pro Node



restricted SSH forced command Pull, read-only

Master auf der Wazuh-VM

- `monitoring-report.py` — die Report-Engine
  - ├ Pull aller Sites
  - ├ Verdict (Ampel je Node)
  - ├ Ollama → Klartext-Bericht
  - ├ HTML-Mail + Detail-PDF je Kunde
  - └ Historie: 13 Monate Aufbewahrung (Trends)
- `sites/*.conf` — eine Datei je Node
- `onboard-site.sh` — Rollout in einem Durchlauf
- `build-web.py` — statische Flotten-Webübersicht

Ist eine Node beim Pull nicht erreichbar, geht automatisch ein roter Ausfall-Alarm raus — „keine Daten" ist selbst ein Befund.

Ausnahme mit System: Nodes, die nicht immer laufen (Workstations), sind als `INTERMITTENT` markiert. Ein stündlicher opportunistischer Pull nimmt mit, was gerade an ist; der Tagesreport bewertet den letzten bekannten Stand und weist dessen Alter aus. Erst wenn die Node tagelang gar nicht auftaucht, wird die Abwesenheit selbst zum gelben Befund — eine ausgeschaltete Workstation ist kein Fehlalarm mehr.

DIE SENSOREN

Collectoren: kleine Scripts, klare Zuständigkeit

Jeder Collector ist ein eigenständiges Script, das genau ein JSON-Objekt ausgibt — inklusive Selbstbewertung. Das Nummern-Präfix steuert nur die Reihenfolge.

10 **speedtest**
Bandbreite gegen Soll, fester Testserver möglich

20 **disk**
Belegung + Inodes, Sofort-Alarm bei voll

30 **updates**
ausstehende Updates (apt), gelb erst nach Karenzzeit

40 **reboot**
ausstehender Neustart; auf Proxmox via Kernel-Vergleich

50 **services**
fehlgeschlagene systemd-Units

60 **temperature**
Sensoren; NVMe am Composite; nur auf Blech

70 **wazuh-agent**
läuft der Security-Agent und ist er verbunden?

75 **patchmon**
ist die Node im Patch-Inventar gemeldet?

76 **nextcloud**
Instanz-Gesundheit aus den Pflege-Snapshots, Multi-Instanz

80 **replication**
Backup-/Replikations-Frische aus Job-Spool-Dateien

81 **autosnapshot**
laufen die ZFS-Auto-Snapshots wirklich? Soll aus den Crons, Ist aus dem Pool

85 **wordpress**
Core-/Plugin-/Theme-Updates via wp-cli, Multi-Instanz

86 **openarchiver**
erfasst das Mail-Archiv wirklich? Quellen-Status + Mail-Frische

87 **pangolin-stack**
Reverse-Proxy-Stack: Container-Health, Erkennungs-Puls, Bans

88 **system**
RAM (ARC-bewusst), Swap, Last, ZFS-Pools, NVMe-SMART — direkt gelesen

89 **dockhand**
Container-Updates der Docker-Flotte, mit Karenzzeit

Updates: melden und beheben

30-updates merkt sich, seit wann jedes Paket aussteht, und wird gelb erst bei Liegenbleibern (Security nach 2 Tagen, alle nach 14) — frisch verfügbare Updates sind grün und normal. Das Gegenstück `setup-unattended-upgrades.sh` rollt automatische Security-Updates flottenweit aus: idempotent, mit Samba-Paket-Blacklist, Proxmox-Kernel bleiben manuell. Gelb heißt jetzt: hier klemmt wirklich etwas. Für Container übernimmt **89-dockhand** dieselbe Rolle: Es liest die Update-Funde des Docker-Managements für die ganze Container-Flotte aus — neue Images hinter demselben Tag ebenso wie neuere Versions-Tags — und meldet sie mit Karenzzeit als *eine* Zeile im Tagesreport statt als Einzelmails.

Backups: Stille ist rot

80-replication liest die Spool-Dateien, die die Backup-Jobs (ZFS-Replikation, Offsite-Pull, PBS) im Checkmk-local-check-Format hinterlassen — und bewertet Zeilen-Status **und** Datei-Frische. Ein Job, der gar nicht mehr liefert, wird rot, nicht still. Ohne Spool-Verzeichnis: „nicht anwendbar", gefahrlos flottenweit verteilbar. Eine Ebene darunter sitzt **81-autosnapshot**: Es leitet das Soll aus den Auto-Snapshot-Crons des Hosts ab und vergleicht es mit den Snapshots, die im Pool tatsächlich liegen — fehlt ein Label oder ist es zu alt, wird es gelb, bewusst nie rot.

Nextcloud: pflegen statt nur messen

Zwei systemd-Timer je Node: wöchentliche occ-Wartung (db-Reparaturen, `app:update --all`, files:scan) und täglicher Status-Snapshot. Core-Updates patch/minor laufen automatisch — mit DB-Dump vorab und Verify danach; Major wird nur gemeldet, Docker-Instanzen kommen übers Image. Mehrere Instanzen je Host über `NC_INSTANCES`; Collector 76 bewertet jede einzeln, die schlechteste Ampel zählt.

Daneben gibt es `scanners.d/` — technisch derselbe Contract, inhaltlich die Frage *„stimmt ein Zustand?"*. Vier Scanner sind live: **10-fancontrol** prüft die Lüfterregelung nicht nur als Dienst, sondern misst, ob die geregelten Lüfter tatsächlich drehen — entstanden, nachdem ein Regel-Dienst nach einem Kernel-Update monatelang still tot war. **20-semaphore** wacht über den Ansible-Scheduler und meldet verwaiste Dauer-Tasks und OOM-Kills — auch das die Antwort auf einen stillen Doppel-Ausfall ohne Alert-Mail. **30-apt-lock** meldet hängende Paketmanager-Prozesse und Watchdog-Eingriffe, **40-kvm-spin** erkennt auf Proxmox-Hosts dauerhaft brennende vCPU-Threads einzelner VMs — Last, die in der Gesamt-CPU des Hosts untergeht. Weitere Kandidaten: offene Ports, Cert-Ablauf, erwartete Prozesse.

ALARMIERUNG

Zwei Ebenen: sofort und täglich

EBENE 1 · SOFORT

Autark auf der Node

Wird ein Collector mit `alert_immediate: true` rot (Platte voll, Internet eingebrochen), mailt die Node **sofort selbst** über ihr lokales msmtp — unabhängig davon, ob der Master lebt.

Stündlicher Cron, Dedup auf 1 Mail/Tag pro Sensor, re-scharf bei Erholung.

EBENE 2 · TAGESREPORT

Zentral am Master

Jeden Morgen um 07:10 — nach dem nächtlichen Update- und Reboot-Fenster — zieht der Master alle Sites, bildet je Node ein Ampel-Verdict und lässt das lokale LLM (Ollama) einen Klartext-Bericht formulieren — gruppiert **pro Kunde**: eine HTML-Mail mit allen Geräten, dazu ein Detail-PDF als Anhang (jedes Gerät auf eigener Seite).

Details erscheinen nur bei ≠ grün — der Report bleibt lesbar.

BERICHTSWESEN

Vom Tages-Digest bis zum GF-Monatsbericht

Aus derselben Trend-Historie entstehen drei weitere Sichten — ohne zusätzlichen Zugriff auf die Nodes:

Monatsbericht für die GF

Aggregiert je Kunde Prüfabdeckung, Befund-Tage, Update-Verlauf und Schutz-Software zu einem PDF mit Management-Summary — bewusst nüchtern formuliert, mit deterministischem Fallback, falls das LLM ausfällt.

Report-Archiv

Jeder Echtlauf legt HTML und PDF strukturiert nach Kunde/Jahr/Monat ab, Cleanup automatisch nach 400 Tagen. Kein Suchen in Mail-Postfächern.

Flotten-Webübersicht

Eine statische Website aus der Historie: Flotten-Tabelle, pro Kunde das Report-Archiv, pro Gerät Ampel-Verlauf, Trend-Charts und tägliche Detail-Seiten samt PDF. Ausgeliefert hinter dem Reverse-Proxy, Zugriff per IP-Allowlist. Dazu die **NOC-View**: Flotte, LibreNMS, Uptime-Kuma, Wazuh und offene Aufgaben als eine Live-Kachelübersicht, alle 5 Minuten frisch — nur intern.

ERWEITERBARKEIT

Der Collector-Contract: Script ablegen, fertig

Das Herzstück der Modularität. Ein neuer Sensor ist ein ausführbares Script in `collectors.d/`, das ein JSON-Objekt nach diesem Muster ausgibt — weder Runner noch SSH-Key noch Master müssen angefasst werden:

```
{
  "name":    "disk",
  "title":   "Festplatten",
  "status":  "green",          // die Ampel-Selbstbewertung
  "summary": "Root 42 %, Daten 67 %",
  "alert_immediate": true,     // rot → Node mailt sofort selbst
  "metrics": { "root_pct": 42 }, // fürs LLM + Trends
  "details": [ "/  42 %  (210 GB frei)" ]
}
```

- ✓ **Immer valides JSON, immer Exit 0.** Auch im Fehlerfall — der Zustand steckt im JSON, nicht im Exitcode.
- ✓ **Read-only.** Collectoren lesen nur; installiert und geändert wird ausschließlich übers Onboarding.
- ✓ **Schnell & billig.** Läuft täglich oder öfter — Timeout-gefährdete Aufrufe selbst kappen.
- ✓ **Schwellwerte aus der Umgebung.** `agent.env` pro Node (z. B. `DISK_WARN_PCT`), sonst vernünftige Defaults.
- ✓ **„Nicht anwendbar" ist grün.** Ein Collector, der auf einer Node nichts zu prüfen findet, meldet das sauber — so lässt sich derselbe Satz gefahrlos auf die ganze Flotte verteilen; der Master blendet nicht anwendbare Sensoren komplett aus dem Report aus.

ROLLOUT

Onboarding: eine neue Node in einem Durchlauf

`onboard-site.sh` läuft am Master und erledigt den kompletten Rollout — alle Fragen kommen vorab, danach läuft es durch. Die Site-Config entsteht erst, wenn der Pull-Test über den Produktionsweg bestanden ist. Pro Lauf genau **eine** Passphrase- und **eine** sudo-Abfrage — egal wie viele Schritte folgen:

1 / 8	**Fragen vorab** — Site, Host, Kunde, Collector-Set, Speedtest-Soll, Mail, optional Wazuh/PatchMon
2 / 8	**Agent-Bundle packen** — und zum Zielhost kopieren
3 / 8	**Installation auf der Node** — nicht-interaktiv; Selbsttest muss grün sein, sonst Abbruch
4 / 8	**Wazuh-Agent (optional)** — Version automatisch passend zum Manager am Master
5 / 8	**Automatische Security-Updates** — unattended-upgrades, Default an; Samba-Pakete ausgenommen
6 / 8	**PatchMon-Enrollment (optional)** — mit Preflight, der fehlende Freischaltung sauber erkennt
7 / 8	**Pull-Test** — über den echten Produktionsweg: restricted Key, forced command
8 / 8	**Site-Config anlegen** — erst nach bestandenem Pull-Test
9 / 8	**Report-Probelauf** — Dry-Run, danach optional gleich der erste echte Versand

Profile machen daraus Serien-Rollouts: eine `.profile`-Datei pro Node-Rolle bündelt Collector-Set, Wazuh-Gruppe und Mail-Defaults — hypervisor, server, server-extern, workstation sowie **samba-dc, samba-member und samba-standalone**. Die Samba-Profile schlagen die Brücke zum Security-Layer: Sie rollen nach dem Wazuh-Schritt automatisch das passende smb.conf-Audit aus dem Schwester-Repo `wazuh-config` mit aus. Precedence: CLI-Flag → Profil → Nachfrage.

```
# voll interaktiv
./onboard-site.sh

# geführt: Profil-Defaults, Rest wird gefragt
./onboard-site.sh --profile server

# Einzeiler für Serien-Rollouts
./onboard-site.sh -p server --site 'Kunde-Meier-DC1' --host 10.20.30.40 -y
```

Optionale Anbindungen im selben Durchlauf: `--trmm` installiert den Tactical-RMM-Agent gleich mit (den Install-Token erzeugt der Master selbst über die TRMM-API). `--wireguard` bindet Insel-Standorte ohne Site-VPN per WireGuard-Tunnel an die Plattform: Die Node baut den Tunnel **ausgehend** auf — NAT und dynamische IP beim Kunden spielen keine Rolle, kein Port-Forward nötig. Den Peer legt das Onboarding automatisch über die Firewall-API an, geroutet werden ausschließlich die Plattform-Ziele, und Wazuh/PatchMon enrollen anschließend direkt durch den Tunnel. Ohne bestandenen Tunnel-Wirkungstest bricht das Onboarding ab.

Für den Betrieb danach: `update-agent.sh` aktualisiert nur den Code — Config, Daten und Keys bleiben unangetastet; neue Sensoren kommen gezielt per `--add-collector` dazu. `uninstall-agent.sh` räumt sauber ab.

EINORDNUNG

Teil einer größeren Plattform

MrsNanny ist der **Health-Layer** einer dreiteiligen Security-/Monitoring-Plattform — und im MrRowie-Konzept das Wachsamkeits-Modul, dessen Schwester-Modul MrMiyagi das schlafende Offsite-Backup übernimmt. Er beißt sich nicht mit Wazuh und PatchMon — er verzahnt sich mit ihnen:

LAYER	WERKZEUG	FRAGE
Health	MrsNanny (dieses Projekt, Repo monitoring-agent)	Geht es der Maschine gut? Platte, Updates, Dienste, Temperatur, Bandbreite, Backups.
Security	Wazuh (SIEM) + Repo `wazuh-config`	Passiert etwas Verdächtiges? Der Onboarding-Schritt installiert den Wazuh-Agent gleich mit; Collector 70 überwacht ihn danach. Die Manager-Konfiguration ist im Schwester-Repo versioniert (nächster Abschnitt).
Patch-Inventar	PatchMon	Was ist installiert, was fehlt? Enrollment ebenfalls im Onboarding; Collector 75 prüft die Anbindung.

Der Kniff: Die Nachbar-Systeme werden nicht nur mitinstalliert, sondern selbst überwacht. Fällt ein Wazuh-Agent aus, meldet es der Tagesreport — die Monitoring-Ebenen kontrollieren sich gegenseitig.

DAS SCHWESTER-REPO

wazuh-config: der Security-Layer, genauso versioniert

Was der Monitoring-Agent für den Health-Layer ist, ist `wazuh-config` für den Security-Layer: die komplette Wazuh-Manager-Konfiguration als Git-Repo — Regeln, Decoder, Gruppen-Configs, Samba-Audit. Quelle der Wahrheit ist das Repo, nicht der Manager.

Änderungen werden zuerst committet und dann per `deploy.sh` ausgerollt (mit `.bak`-Sicherung, Config-Verify und Manager-Restart) — nie direkt am Manager editiert. Damit bleibt ein Manager-Neuaufbau jederzeit reproduzierbar.

manager/rules + decoders

Rule-Tuning (Brute-Force-Entrauschen, Web-Scan-Rauschen, Filter für den Monitoring-User) und eigene Samba-Regeln samt passenden JSON-Decodern — als XML im Repo, nicht als Handarbeit auf der VM.

manager/shared

Eine Agent-Config je Wazuh-Gruppe (exposed, hypervisor, linux-server, workstation, webserver — letztere mit Vhost-Logs und Echtzeit-FIM auf den Webroots): SCA-Intervalle, syscheck. Dieselben Rollen wie die Onboarding-Profile des Monitoring-Agents.

samba/*.snippet

Fertige `smb.conf`-Blöcke für Domain-Controller (Auth-Audit als JSON) und Fileserver (Datei-Audit auf Änderungs-Ops, bewusst ohne Flooding) — via Samba-Onboarding-Profil automatisch mit ausgerollt.

integrations/custom-signal

Kritische Alarme gehen zusätzlich als Signal-Nachricht aufs Handy — CVE-Meldungen pro CVE gedrosselt (24 h Ruhe) und in ein 60-Minuten-Sammelfenster gebündelt — ein Feed-Schwung über die ganze Flotte ergibt eine Nachricht, nicht Dutzende. Dazu eine generische Burst-Bremse je Regel und Host (30 Minuten Ruhe).

Das Ergebnis sind Erkennungen, die über Standard-Wazuh hinausgehen — vor allem auf den Samba-Fileservern. Die Schwellen sind im Praxisbetrieb kalibriert: hoch genug gegen Alarm-Flut, mit Wiederholungs-Sperre pro Aggregat. Entrauscht wird über eine **Level-Leiter** statt per Löschen: bekannter Lärm sinkt auf leise Audit-Level unter Alarm- und Digest-Schwelle, Aggregate zählen weiter — nichts verschwindet, es wird nur leiser. Genau so laufen die Massen-Aggregate (100120/100121) im Tagesbetrieb bewusst auf Audit-Level 7 — Groß-Kopien und Aufräumaktionen sind von außen nicht von einem Vorfall zu unterscheiden.

REGEL	LEVEL	ERKENNT
100103	10	SSH/Samba-Brute-Force: 5+ Fehl-Logins pro IP in 2 Minuten
100104	10	Passwort-Raten: 5+ Fehlversuche gegen einen Benutzer in 5 Minuten
100112	5	Einzelne Datei-Löschung (Audit-Trail)
100113	3	Papierkorb-Leeren — zählt nicht als Massen-Löschung
100120	7	**Massen-Löschung**: 50+ Löschungen pro Benutzer in 60 s (Insider-Szenario)
100121	7	**Ransomware-Verdacht**: 100+ Datei-Änderungen pro Benutzer in 60 s
100130	12	**Audit-Ausfall**: full_audit verweigert Share-Connects (Fail-Closed)

Zusammenspiel: Der Monitoring-Agent bringt den Wazuh-Agent auf die Node und überwacht ihn (Collector 70); wazuh-config bestimmt versioniert, was der Manager daraus macht. Health- und Security-Layer teilen sich dieselbe Rollen-Logik — zwei Repos, eine Plattform.

AUSBLICK

Verteilung als Baukasten (in Arbeit)

Der nächste Ausbauschritt macht die Plattform selbst verteilbar: `setup.sh` liest eine zentrale `platform.env` und orchestriert Module in `modules.d/` — core, llm, wazuh, patchmon, signal, web. Der **Modul-Contract** spiegelt den Collector-Contract eine Ebene höher: Neues Modul = Datei ablegen, `setup.sh` muss es nicht kennen.

Kernidee ist die Drei-Wege-Prüfung jedes Moduls: *bereit* (installieren), *nicht konfiguriert* (leere Pflichtwerte = gewolltes Nein, still überspringen) oder *kaputt* (konfiguriert, aber Endpoint tot = harter Fehler). Add-ons leuchten auf, sobald ihre Abhängigkeit da ist — ein leerer Abschnitt in `platform.env` ist kein Fehler, ein gefüllter mit totem Endpoint schon.

STAND

Was läuft, was kommt

■ IN PRODUKTION

- Agent-Runner, 16 Collectoren, 4 Scanner, Sofort-Alarm — inkl. System-Grundwerten (RAM, Last, ZFS-Pools, NVMe) und ZFS-Auto-Snapshots direkt gelesen, Fremd-Agent abgelöst
- Master-Report: Pull → Verdict → Ollama → HTML-Mail + Detail-PDF je Kunde
- Monats-GF-Report, Report-Archiv (400 Tage), Flotten-Webübersicht + NOC-View
- Automatische Security-Updates flottenweit + Update-Karenzzeit
- Nextcloud-Pflege als systemd-Timer: occ-Wartung, Core-Updates patch/minor, Multi-Instanz
- Onboarding mit 7 Rollen-Profilen (inkl. Samba-Audit) + Einzeiler
- Optionale Onboarding-Schritte: TRMM-Agent, WireGuard-Tunnel für Insel-Standorte
- Installer, Updater (--add-collector), Uninstaller
- Flotten-Wächter im Tagesreport: PatchMon-Stale, Newt-Drift, Container-Updates der Docker-Flotte
- Historie: 13 Monate Aufbewahrung; rund 100 Nodes, 10 Kunden-Gruppen

□ GEPLANT / BEWUSST OFFEN

- Weitere Scanner: Ports, Certs, erwartete Prozesse
- Automatische Update-Kette für Apps und Container: Snapshot → Update → Wirkungs-Prüfung → Rollback bei Fehlschlag (Konzept steht; die Erkennung läuft bereits über Collector 89)
- Modulare Verteilung: Module + Contract stehen, Praxis-Härtung offen
- Windows-Health: Entscheidung offen (Kandidat: RMM-Brücke)
- Alpine Linux: bewusst nicht unterstützt (bash/apt/systemd-Annahmen)