

Paralleler Malware-Scanner

Automatisiertes Scanning & AI-Analyse mit n8n

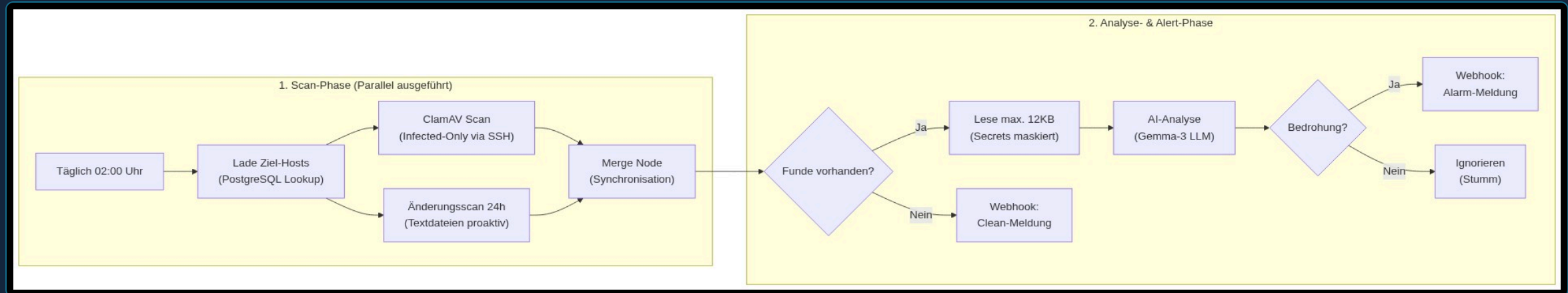
Paul Kozber - ORDIX AG

1. Herausforderungen klassischer Malware-Scans

- **Sequenzielle Latenz:** Nacheinander ausgeführte SSH-Scans über viele Hosts hinweg blockieren den gesamten Ablauf, sobald ein einzelner Host langsam antwortet.
- **Abstürze durch Binärdateien:** Proaktive Änderungsscans erfassen oft kompilierte Binärdaten. Werden diese an lokale LLMs gesendet, stürzen die Parser ab.
- **Kontextlimits & HTTP-Timeouts:** Zu große Log- oder Quellcodedateien überschreiten das Kontextfenster lokaler KI-Modelle und führen zu Fehlern.
- **Alert-Fatigue:** Ungefilterte Webhook-Benachrichtigungen für alltägliche, harmlose Log-Rotations führen zu Desinteresse bei echten Warnungen.
- **Das Ziel:** Ein performanter, parallelisierter Scan-Workflow, der nur verifizierte Bedrohungen meldet.

2. Die Lösung: Parallele n8n-Architektur

- **Parallele Ausführung:** ClamAV- und Dateiänderungsscans laufen zeitgleich per SSH auf allen Hosts.
- **Synchronisation:** Ein **Merge-Node** wartet auf alle Hosts, bevor die Auswertung startet.
- **Visualisierter Workflow:**



3. Die Ziel-Datenbank (PostgreSQL Target Hosts)

- **Dynamischer Lookup:** n8n lädt die zu scannenden Zielsysteme dynamisch aus einer Tabelle namens `target_hosts` der n8n-eigenen PostgreSQL-Datenbank.
- **Datenbank-Schema (SQL):**

```
CREATE TABLE IF NOT EXISTS target_hosts (  
  id SERIAL PRIMARY KEY,  
  server_name VARCHAR(100) NOT NULL,  
  host_ip VARCHAR(50) NOT NULL,  
  scan_path VARCHAR(255) NOT NULL,  
  ssh_private_key TEXT NOT NULL,  
  active BOOLEAN DEFAULT TRUE  
);
```

- **Vorteil:** Die Scanner-Ziele müssen nicht im n8n-Workflow hartcodiert werden, sondern werden vollautomatisch über die Host-Konfigurationen gesteuert.

4. Modularer Ansible-Sync

- **Deklaratives Inventar:** Ziel-Hosts werden über die Ansible Host-Variablen für den Scan freigeschaltet.
- **Auszug aus** `inventory/host_vars/client1/main.yml` :

```
malware_scan_enabled: true
malware_scan_paths:
  - /opt
  - /home/deploy
```

- **Datenbank-Synchronisation:** Das Playbook `playbooks/sync-malware-targets.yml` liest das Ansible-Inventar aus und trägt die aktiven Hosts, Zielpfade und SSH-Schlüssel direkt in die PostgreSQL-Tabelle ein:

```
ansible-playbook -i inventory/hosts.yml playbooks/sync-malware-targets.yml
```

5. SQL-Synchronisations-Ablauf (Ansible Tasks)

- Tabellenbereinigung und Host-Loop:
 - Das Playbook löscht alte Einträge mit:
`docker exec -i n8n-postgres psql -U n8n -d n8n -c "TRUNCATE TABLE target_hosts;"`
 - Anschließend läuft ein Insert-Loop über die aktiven Hosts.
- SQL Insert-Statement in `playbooks/tasks/sync-host-paths.yml` :

```
INSERT INTO target_hosts (server_name, host_ip, scan_path, ssh_private_key, active)
VALUES (
    '{{ target_host }}',
    '{{ hostvars[target_host].ansible_host }}',
    '{{ path_item }}',
    '{{ private_key_content | replace("'", "'") }}',
    true
);
```

6. Der ClamAV Signature Scan

- **Signaturbasiertes Scannen:** Das Bash-Skript `run_clamscan.sh` triggert ClamAV auf den Zielsystemen via SSH.
- **Exit-Code-Mapping:**
 - ClamAV gibt bei Funden standardmäßig den Exit-Code `1` zurück.
 - Um zu verhindern, dass n8n dies als Systemabsturz wertet und den Workflow abbricht, fängt das Skript den Code `1` ab und gibt ihn als normalen Erfolg (`0`) mit der Liste gefundener Viren aus.
- **Beispiel-Befehl:**

```
clamscan -r --infected --exclude-dir="/sys" --exclude-dir="/dev" /opt
```

7. Proaktive Änderungsscans & Filter

- **Erkennung von Zero-Day-Code:** Der proaktive Änderungsscan sucht nach Dateien, die in den letzten 24 Stunden modifiziert oder neu erstellt wurden.
- **Filterung direkt auf dem Client (`find_new_files.sh`):**

 - **Verzeichnis-Blacklist:** Überspringt Rausch-Ordner wie `node_modules` , `.git` , `cache` , `log` und `dist` .
 - **Dateiendungs-Whitelist:** Beschränkt die Analyse auf Klartextdateien zur Vermeidung von Binärübertragungen:

```
find /opt -mmin -1440 -type f \( -name "*.py" -o -name "*.js" -o -name "*.sh" -o -name "*.yaml" -o -name ".env" -o -name "Dockerfile" \)
```

8. LLM-Kontext-Schutz & Secrets Redaction

- **Dateigrößen-Limitierung** (`read_file.sh`):
 - Liest maximal die ersten 12 Kilobyte einer geänderten Datei aus (`head -c 12000`).
 - Dies schützt das lokale LLM vor Token-Überläufen und verhindert Timeouts.
- **Serverseitige Secrets Redaction:**
 - Das Ausleseskript auf den Clients schwärzt sensible API-Tokens, Passwörter und SSH-Keys per Regex, bevor die Daten übertragen werden:

```
sed -r 's/(password|passwd|key|token|secret)\s*[:=\s*\S+/\1: [REDACTED]/gi'
```

- **Vorteil:** Credentials verlassen niemals unverschlüsselt den Quellhost.

9. Lokale KI-Analyse (Local AI)

- **KI-Schnittstelle:** n8n verbindet sich via REST-Schnittstelle mit einer lokalen LM Studio- oder Ollama-Instanz.
- **Modell & Endpoint:**
 - **Modell:** `google/gemma-3-4b`
 - **Endpoint:** `http://192.168.1.50:1234/v1/chat/completions`
- **Prompt-Struktur:**
 - Der System-Prompt fordert eine strikte Klassifizierung am Anfang der Antwort: `[CLEAN]` , `[SUSPICIOUS]` oder `[MALICIOUS]` .
 - Das Modell liefert maximal 3 aussagekräftige Bulletpoints zur Begründung.

10. Local AI System Prompt Details

- Konfiguration des System-Prompts:

```
Du bist ein wachsamer, technischer Sicherheits-Auditor.  
Analysiere den folgenden Dateiinhalt auf schädliche Verhaltensweisen, Hintertüren,  
Obfuskation oder hartcodierte Credentials.
```

```
Antworte zwingend im folgenden Format:
```

```
[URTEIL] (Ersetze URTEIL durch CLEAN, SUSPICIOUS oder MALICIOUS)
```

- Punkt 1 zur Begründung
- Punkt 2 zur Begründung (maximal 3 Punkte)

```
Antworte kurz, präzise und rein technisch. Keine Einleitung!
```

11. Intelligente Webhook-Alarmierung

- **Rauschunterdrückung:**
 - Eine `IF-Node` im n8n-Workflow prüft die LLM-Klassifizierung.
 - Erkenntnisse mit `[CLEAN]` werden sofort verworfen.
 - Nur `[SUSPICIOUS]` und `[MALICIOUS]` triggern den Chat-Webhook.
- **Webhook Payload:**
 - Übermittelt strukturiertes JSON mit Dateipfad, Host-IP, Erkennungstyp (ClamAV oder Dateiänderung) sowie der Begründung des LLMs.
- **Vorteil:** Keine Flut an Benachrichtigungen. Der Admin wird nur bei echten Anomalien alarmiert.

12. Debugging & Manuelles Testen

- Manuelle ClamAV Ausführung (auf dem Client):

```
sudo clamscan -r --infected /opt
```

- Manuelle Abfrage der n8n-Scannerdatenbank (auf dem n8n-Host):

```
docker exec -it n8n-postgres psql -U n8n -d n8n -c "SELECT server_name, host_ip, scan_path, active FROM target_hosts;"
```

- KI-API-Erreichbarkeit prüfen (vom n8n-Host aus):

```
curl -X POST http://192.168.1.50:1234/v1/chat/completions \
-H "Content-Type: application/json" \
-d '{"model": "google/gemma-3-4b", "messages": [{"role": "user", "content": "Test"}]}'
```

13. Backup, Import & Recovery

- **Workflow-Sicherung:** Eine vollständige JSON-Sicherung des konfigurierten n8n-Workflows liegt im Git-Repository:
`submodules/homelab-services/services/n8n/malware-scanner-workflow.json`
- **Wiederherstellungsschritte:**
 1. Öffne die n8n Administrationsoberfläche.
 2. Erstelle einen neuen leeren Workflow.
 3. Klicke im Menü oben rechts (drei Punkte) auf **Import from File**.
 4. Wähle die Datei `malware-scanner-workflow.json` aus.
 5. Speichere den importierten Workflow ab und verknüpfe die PostgreSQL-Verbindung sowie den SSH-Private-Key.