

**Exposé: Security-Compliant
Container-Supply-Chain und Patch-Management in
isolierten Kubernetes-Umgebungen**

von

Tim Wall

Mat. Nr. 5014365

Saarbrücken, den 10. Juni 2026

1 Forschungsfrage

Wie lässt sich eine sichere Softwarearchitektur für netzwerkisolierte Kubernetes-Cluster entwerfen, die eine automatisierte Container-Supply-Chain mit integrierter Security-Compliance (Image-Signing, Vulnerability Scanning, SBOM) und höchstmöglich automatisiertem Patch-Management ohne Internetverbindung gewährleistet?

2 Problemstellung

Moderne Software wird zunehmend über containerisierte Anwendungen verteilt und mit Orchestrations-Software wie Kubernetes verwaltet. Gleichzeitig werden hochregulierte On-Premise und Edge Umgebungen immer öfters vollständig ohne Internetanbindung betrieben. Regularien wie der Cyber-Resilience-Act setzen Software Bill of Materials (SBOMs), Schwachstellenmanagement, Incident Response und Meldepflichten vor. Diese Herausforderungen benötigen eine durchdachte Architektur, welches automatisiert SBOM-Generierung, Vulnerability-Scanning, Image-Signatur, Deployment und Policy-Enforcement in isolierten Kubernetes-Umgebungen integriert.

3 Nicht-Ziele

Außerhalb des Umfangs dieser Arbeit sind Schwachstellen, die nicht die Container-Supply-Chain, das Patch-Management oder die Cluster-Verwaltung betreffen, zum Beispiel physische Schwachstellen. Desgleichen sind Multi-Cluster-Umgebungen und Windows-Nodes kein Ziel der Arbeit. Die Basis bildet der Orchestrator Kubernetes, die Übertragbarkeit auf Alternativen wie Docker Swarm oder HashiCorp Nomad ist nicht garantiert. Die Anforderungsanalyse und das darausfolgende Architekturkonzept basiert auf dem konkreten Unternehmenskontext der psb intralogistics GmbH und ist nicht für alle AirGap-Szenarien generalisierbar.

4 Motivation

Schwachstellen in weltweit genutzter Software und deren Supply-Chains werden im KI-Zeitalter schneller gefunden, (siehe xz, Copy Fail, Dirty Frag, npm Vorfälle) und können gleichzeitig schneller ausgenutzt werden. Während netzwerkisolierte Installationen ihre Angriffsfläche minimieren, ist sie nicht Null. Die Software-Supply-Chain, physischer Zugang und alternative Extraktionsmöglichkeiten wie Elektromagnetismus bleiben Wege, um ein Netzwerk zu infiltrieren oder Daten zu extrahieren. Somit ist auch innerhalb eines AirGap-Deployments ein Architekturkonzept notwendig, um die Sicherheit zu gewährleisten.

5 Zielgruppe

Die primäre Zielgruppe sind Cybersicherheitspezialisten und DevOps-Engineers, welche in regulierten Umgebungen AirGap-Kubernetes-Cluster bereitstellen und betreiben. Darunter zählen Behörden, kritische Infrastrukturbereitsteller, Finanzinstitute, Rüstungsunternehmen oder Industriekonzerne mit strengen Netzwerkanforderungen. Compliance-Verantwortliche bilden die Sekundärzielgruppe, da sie Anhand der Bewertung des Clusters Stärken und Defizite in existierenden Kriterienkatalogen identifizieren können.

6 Heutiger Forschungsstand

AirGap-Umgebungen und Provisionierung auf Basis von kubeadm, k3s, RKE2 und OpenShift sind gut Dokumentiert. Forschung zu Schwachstellenmanagement ist für Container in Online-Umgebungen abgedeckt. Sigstore Cosign, Trivy, Gype und Syft bilden die Grundlage für eine Container-Supply-Chain. Patch-Management in AirGap-Clustern ist eine akademische Forschungslücke. Es fehlt eine Referenzarchitektur für Offline-Umgebungen, aufbauend auf einem getesteten Gesamtsystem.

7 Methodischer Ansatz

Basierend auf der Analyse eines konkreten Anwendungsszenarios anhand der psb intralogistics GmbH und einer Literatur- sowie Tool-Analyse wird ein Anforderungskatalog erarbeitet. Darauf aufbauend wird eine Referenzarchitektur entworfen, implementiert und bezüglich Compliance und Sicherheit evaluiert.

8 Notwendige Ressourcen

Schätzung der benötigten Ressourcen:

- Infrastruktur: 3-5 Linux-Maschinen (Bare-Metal oder VM) zum Instanzieren von isolierten Kubernetes-Clustern
- Software: Ansible / pyinfra (Provisionierung), kubeadm / RKE2 / k3s / Talos (Distribution), Harbor Registry / Sonatype Nexus Repository (Repo), Cosign (Signierung), Trivy / Gype / Clair (CVE-Scan / SBOM), Syft (SBOM), Spiegel (Image-Mirror), Kyverno / OPA Gatekeeper (Policy Admission), FluxCD / ArgoCD (GitOps), Hauler / Zarf (AirGap Transport)
- Zeit: 13 Wochen

9 Ergebnis

Primär entsteht eine dokumentierte Referenzarchitektur, welche als Blaupause für das Bereitstellen von sicheren AirGap-Kubernetes-Clustern mit integrierter Container-Supply-Chain-Sicherheit dient, sowie ein dazugehöriges Mapping, welches die Einhaltung der Compliance-Anforderungen aufzeigt (CRA, SLISA). Ergänzend entsteht

ein Proof-of-Concept, der die Architektur in einer Testumgebung mit den darauffolgenden Artefakte (z.B. GitOps-Directory, Helm-Charts, Ansible-Playbooks) abbildet.

10 Zeitplan

Woche Aufgabe → Ergebnis

- 1 Grundlagen-Recherche (k8s, AirGap, Supply-Chain, Patch-Management)
→ Draft Theoretische Grundlagen
- 2 Anforderungsanalyse psb und Literaturrecherche
→ funktionale & qualitative Anforderungen, Zielkonflikte, Testszenarien
- 3 Architektur-Entwurf, Entscheidungsbegründung (ATAM Utility-Tree)
→ Draft Architektur, Diagramme Komponenten & Runtime
- 4 Architektur-Feedback psb & Betreuer, Finalisierung Architektur
→ Finalisierung Architektur
- 5-6 PoC Setup: Container-Supply-Chain (CVE, Signierung)
→ Funktionale Pipeline & Draft PoC Container-Supply-Chain
- 7-8 PoC Setup: AirGap-Cluster, Provisionierung, Registry, Policies
→ Funktionales Cluster & Draft PoC Cluster
- 9 PoC: Patch-Management, mögliche Automatisierung
→ Draft Patch-Management
- 10 PoC Testing & Metrikerhebung (Deployment-Zeit, Automatisierung,)
→ Draft Metriken, Erkenntnisse, Diagramme Metriken
- 11 Compliance-Bewertung Architektur & PoC (CRA, SLSA)
Architektur-Bewertung (STRIDE, Risiken, Tradeoffs)
→ Draft Compliance, Draft Compliance-Bewertung
- 12 Diskussion, Fazit & Ausblick
→ Draft Diskussion, Fazit & Ausblick
- 13 Proof-Read, Finalisierung, Signierung, Binden, Vorbereitung Kolloquium
→ Finale Arbeit