

Bachelor-Thesis

zur Erlangung des akademischen Grades

Bachelor of Science (B. Sc.)

an der Hochschule für Technik und Wirtschaft des Saarlandes

im Studiengang Praktische Informatik

der Fakultät für Ingenieurwissenschaften

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

vorgelegt von

Tim Wall

betreut und begutachtet von

Prof. Dr.-Ing. Steffen Knapp

Christoph Karls, M.Sc.

Saarbrücken, Tag. Monat Jahr

Selbständigkeitserklärung

Ich versichere, dass ich die vorliegende Arbeit (bei einer Gruppenarbeit: den entsprechend gekennzeichneten Anteil der Arbeit) selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Ich erkläre hiermit weiterhin, dass die vorgelegte Arbeit zuvor weder von mir noch von einer anderen Person an dieser oder einer anderen Hochschule eingereicht wurde.

Darüber hinaus ist mir bekannt, dass die Unrichtigkeit dieser Erklärung eine Benotung der Arbeit mit der Note „nicht ausreichend“ zur Folge hat und einen Ausschluss von der Erbringung weiterer Prüfungsleistungen zur Folge haben kann.

Saarbrücken, Tag. Monat Jahr

Tim Wall

Sperrvermerk

Die vorliegende Arbeit mit dem Titel „Security-Compliant Container-Supply-Chain und Patch-Management in isolierten Kubernetes-Umgebungen“ enthält vertrauliche Daten des Unternehmens psb intralogistics GmbH.

Die Arbeit darf nur dem Erst- und Zweitgutachter sowie befugten Mitgliedern des Prüfungsausschusses zugänglich gemacht werden. Eine Veröffentlichung und Vervielfältigung der Arbeit ist – auch in Auszügen – nicht gestattet.

Eine Einsichtnahme der Arbeit durch Unbefugte bedarf einer ausdrücklichen Genehmigung des Verfassers und des Unternehmens psb intralogistics GmbH.

Zusammenfassung

1: Abstract am Ende schreiben

Kurze Zusammenfassung des Inhaltes in deutscher Sprache, der Umfang beträgt zwischen einer halben und einer ganzen DIN A4-Seite.

Orientieren Sie sich bei der Aufteilung bzw. dem Inhalt Ihrer Zusammenfassung an Kent Becks Artikel: <http://plg.uwaterloo.ca/~migod/research/beck00PSLA.html>.

*We have seen that computer programming is an art,
because it applies accumulated knowledge to the world,
because it requires skill and ingenuity, and especially
because it produces objects of beauty.*

— Donald E. Knuth [1]

Danksagung

An dieser Stelle möchte ich allen Personen meinen herzlichen Dank aussprechen, die mich während der Erstellung dieser wissenschaftlichen Arbeit unterstützt haben. Mein besonderer Dank gilt meinem Betreuer Christoph Karls für die hilfreichen Anregungen und die wertvollen Rückmeldungen während des gesamten Arbeitsprozesses. Darüber hinaus danke ich psb intralogistics GmbH für ihre Bereitschaft, mich bei der Bachelorarbeit zu unterstützen.

Ebenso danke ich meiner Familie für ihre Unterstützung, ihre Geduld und die motivierende Begleitung. Ohne ihre Hilfe wäre die Anfertigung dieser Arbeit in dieser Form nicht möglich gewesen.

Inhaltsverzeichnis

1	Einleitung	1
1.0.1	Problemstellung	1
1.0.2	Zielsetzung	1
1.0.3	Forschungsfrage	1
1.0.4	Eigener Beitrag	1
1.0.5	Aufbau der Arbeit	1
1.1	L ^A T _E X installieren und einrichten	1
1.1.1	Unter Windows	1
1.1.2	Unter Linux	2
1.2	Entwicklungsumgebungen	2
1.3	Werkzeuge	2
1.4	Struktur und Gebrauch der Vorlage	2
1.4.1	Struktur der Vorlage	3
1.4.2	Gebrauch der Vorlage	3
1.5	Theoretische Grundlagen	5
1.5.1	Kubernetes-Architektur	5
1.5.2	Air-Gapped-Umgebungen	5
1.5.3	Container-Supply-Chain Sicherheit	5
1.5.4	GitOps	6
1.5.5	Architekturbewertungsmethoden	6
1.6	Anforderungserhebung	7
1.6.1	Stakeholder-Analyse	7
1.6.2	Funktionale Anforderungen	7
1.6.3	Qualitative Anforderungen	7
1.6.4	Randbedingungen	7
1.6.5	Zielkonflikte	7
1.7	Architekturentwurf	9
1.7.1	Systemkontext	9
1.7.2	Komponenten	9
1.7.3	Laufzeit	9
1.7.4	Topographie	9
1.8	Architekturbewertung	11
1.8.1	ATAM: Utility Tree	11
1.8.2	Szenarien-Bewertung	11
1.8.3	STRIDE-Security-Bewertung	11
1.9	Proof of Concept	13
1.9.1	Bestehendes Setup	13
1.9.2	Security-Gates	13
1.9.3	Automatisierte Patch-Pipeline	13
1.10	Fazit & Ausblick	15
1.10.1	Zusammenfassung	15
1.10.2	Beantwortung der Forschungsfrage	15
1.10.3	Limitationen	15
1.10.4	Ausblick	15

Literatur	17
Abbildungsverzeichnis	19
Tabellenverzeichnis	19
Listings	19
Abkürzungsverzeichnis	21
A Erster Abschnitt des Anhangs	25

1 Einleitung

3: Einleitung schreiben am Ende (4-5 Seiten)

1.0.1 Problemstellung

- Firmen liefern Software an Kunden mit unterschiedlichen Netzwerkumgebungen (Online vs. Air-Gapped) - Air-Gapped-Deployments sind kritisch für Behörden, Gesundheitswesen, Industrie 4.0[1][2] - Aktuelle Herausforderung: Security-Updates ohne Internetverbindung, manuelle Prozesse fehleranfällig

1.0.2 Zielsetzung

- Entwurf einer Softwarearchitektur für hybrides Kubernetes-Deployment (Online + Air-Gapped) - Implementierung automatisierter Container-Supply-Chain mit Security Gates - Automatisiertes Patch-Management mit nachweisbarem CVE-Response (<48h)

1.0.3 Forschungsfrage

- Hauptfrage + 3 Unterfragen (siehe oben)

1.0.4 Eigener Beitrag

- Praktische Migration aus Praktikum (20-30 Container) als Basis - Architektur-Entwurf mit Security-by-Design - Proof of Concept mit Security-Metriken

1.0.5 Aufbau der Arbeit

- Kurze Beschreibung der Kapitel

1.1 \LaTeX installieren und einrichten

1.1.1 Unter Windows

Als LaTeX-Distribution unter Windows steht *MikTeX* zu Verfügung, die als freie Software im Internet erhältlich ist. *MikTeX* unterstützt Windows XP, Vista und Windows 7. Neben *MikTeX* wird noch ein PostScript-Interpreter benötigt, z.B. GhostScript, zu finden auf Chip.de.

Wichtig: Bei *MikTeX* unbedingt Vollinstallation auswählen, sonst sind eventuell benötigte Packages nicht vorhanden.

1 Einleitung

1.1.2 Unter Linux

Unter Linux existiert die LaTeX-Distribution *texlive*, die als aktuelle Version aus den Paketquellen geladen werden kann (unter Ubuntu mit `apt-get install texlive-full`). Auch hier ist ganz wichtig, die volle Distribution zu laden, damit alle Packages zur Verfügung stehen.

1.2 Entwicklungsumgebungen

Hat man die passende Distribution installiert, bieten sich vielerlei Möglichkeiten an ein LaTeX-Projekt anzugehen oder einzelne Dokumente zu editieren. Unter Windows könnten dies folgende sein:

TeXnicCenter Umfangreiche Entwicklungsumgebung mit Projektorganisation und Autovervollständigung

TeXLipse Eclipse-Plugin, das alle Vorteile der Eclipseumgebung mit LaTeX verbindet

TeXmaker Einfacher LaTeX-Editor mit Pdf-Direktvorschau

Unter Linux stehen bereit:

Gummi Ebenfalls einfacher LaTeX-Editor mit Direktvorschau

TeXLipse Auch für Linux erhältlich

Kile Umfangreiche Entwicklungsumgebung, ähnlich wie TeXnicCenter

Nach der Installation muss die Entwicklungsumgebung eingerichtet werden; dazu finden sich viele Anleitungen im Internet, die genau erklären, welche Distribution auf welche Weise eingerichtet wird. Insbesondere sollte der PDF-Viewer festgelegt werden, damit bei Gummi und TeXmaker die Direktvorschau funktioniert. Manchmal kommt es vor, dass die Ausgabe nach dem Kompilieren Umlaute und Sonderzeichen nicht richtig darstellt. Unter Linux hängt dies mit den unterschiedlichen Zeichensätzen zusammen, die unterstützt werden. Um diese Vorlage zu verwenden ist es notwendig, den verwendeten Zeichensatz des Editors bzw. der Entwicklungsumgebung auf den in diesem Dokument verwendeten Zeichensatz umzustellen: UTF-8 ohne BOM (Byte Order Mark).

1.3 Werkzeuge

JabRef Ein Literaturverwaltungsprogramm, welches das *BibTeX*-Format einsetzt und mithilfe einer graphischen Oberfläche das Anlegen von Literaturverzeichnissen vereinfacht.

1.4 Struktur und Gebrauch der Vorlage

Die vorliegende Vorlage für Abschlussarbeiten besteht aus einer internen Struktur, die grundsätzlich nicht verändert werden sollte.

1.4.1 Struktur der Vorlage

htw-i-mst-config.tex Enthält alle zu ladenden Packages, Styleparameter für Hyperlinks, Codelistings und Literaturverzeichnis sowie globale Parameter für Tabellen und Beschriftungen. Im Besonderen befinden sich hier die Variablen für den eigenen Namen, Titel, Datum der Arbeit, den betreuenden Professor etc.

htw-i-mst-vorlage.tex Dies ist die Hauptdatei, in der alle notwendigen *.tex-Dateien eingebunden werden, die zu dem Dokument gehören. Es empfiehlt sich die interne Struktur *nicht* zu verändern. Eigene Kapitel werden an der dafür markierten Stelle eingebunden.

Bibliography.bib Zentrale Datei für die Literaturangaben, welche man z.B. mit JabRef verwalten kann.

Chapters/ Ablageort für alle selbst angelegten Kapitel der Arbeit. Die Aufteilung in eigene Dateien erleichtert die Übersicht über den Quellcode.

Graphics/ Ablageort für alle im Dokument benötigten Grafikdateien. Gerne darf man hier Unterverzeichnisse zur besseren Strukturierung anlegen.

Examples/ Dieser Ordner enthält die in dieser Vorlage beigelegten LaTeX-Beispiele, welche vor der Abgabe der Arbeit selbstverständlich gelöscht werden sollten.

Frontbackmatter/ In diesem Ordner sind all jene Dateien abgelegt, die – außer dem Kern-text in *Chapters/* – die Gesamtheit der Abschlussarbeit ausmachen.

Titlepage.tex Definiert die Titelseite der Abschlussarbeit. Diese Datei muss normalerweise nicht verändert werden.

Abbreviations.tex Hier werden alle Abkürzungen hinterlegt, die im Dokument verwendet werden.

Abstract.tex Eine kurze Zusammenfassung der Abschlussarbeit wird in diese Datei eingefügt.

Acknowledgements.tex Dort finden Danksagungen ihren Platz.

ConfidentialityClause.tex Beinhaltet den Sperrvermerk und ist nur zu verwenden, falls dies beispielsweise vom beteiligten Unternehmen gefordert wird.

Contents.tex Enthält wichtige Eintragungen in die *Table-of-Contents*. Diese Datei muss normalerweise nicht geändert werden.

Declaration.tex Enthält die Selbständigkeitserklärung. Diese Datei darf nicht geändert werden.

Colophon.tex Enthält einen Hinweis auf die Urheber dieser Vorlage. Diese Datei darf nicht geändert werden.

ListOfs.tex Enthält die Einträge für die Tabellen- und Abbildungsverzeichnisse etc. und muss gewöhnlich nicht verändert werden.

1.4.2 Gebrauch der Vorlage

Grundsätzlich ist nicht viel zu tun, um die Vorlage für Abschlussarbeiten zu verwenden. Man entpackt den Hauptordner in das gewünschte Verzeichnis und nutzt die Dateien so, wie in Abschnitt 1.4.1 beschrieben. Danach werden *alle* Dateien gespeichert und die Hauptdatei, *htwsaar-i-mst-vorlage.tex*, mehrfach kompiliert (LaTeX benötigt mehrere Durchgänge

1 Einleitung

um z.B. Referenzen richtig zuzuordnen). Hat man Änderungen in *Bibliography.bib* bzw. *Bibliography.tex* vorgenommen oder neue Zitate z.B. mittels `\cite` eingefügt, muss erst mit *BibLaTeX* und anschließend mit dem entsprechenden LaTeX-nach-PDF-Compiler übersetzt werden.

1.5 Theoretische Grundlagen

1.5.1 Kubernetes-Architektur

1.5.1.1 Control Plane

4: (API Server, Scheduler, Controller Manager, etcd)

1.5.1.2 Worker Nodes

5: (Kubelet, Kube-Proxy, Container Runtime)

1.5.1.3 Workloads

6: Kernkonzepte: Pods, Deployments, Services, ConfigMaps, Secrets

1.5.2 Air-Gapped-Umgebungen

1.5.2.1 Definition

7: Air-Gap vs. Netzwerktrennung vs. restricted network[1]

1.5.2.2 Herausforderungen

- Herausforderungen: Image-Versorgung, Dependency-Management, Patch-Verfügbarkeit[6][1]

1.5.2.3 Best Practices

- Best Practices für Air-Gapped Kubernetes[7][8][1]

1.5.3 Container-Supply-Chain Sicherheit

1.5.3.1 Software Bill of Materials

- **SBOM** (Software Bill of Materials): Format (SPDX, CycloneDX), Tools (Syft)[1]

1.5.3.2 Image-Signing

- **Image-Signing**: Cosign + Sigstore, Notary v2[1]

1.5.3.3 Vulnerability-Scanning

- **Vulnerability Scanning**: Trivy, Grype, Clair[1]

1.5.3.4 Policy Enforcement

- **Policy Enforcement**: OPA Gatekeeper, Kyverno für Image-Verification

1 Einleitung

1.5.4 GitOps

****2.4 GitOps & Automatisierung**** (3 Seiten) - GitOps-Prinzipien: Declarative Configuration, Continuous Sync[9] - ****Flux CD****: Architektur, Sync-Strategien, Multi-Cluster[10][9] - CI/CD-Pipelines für Security Gates (GitLab CI, GitHub Actions)

8: denken ob arc42 oder nicht

****2.5 Softwarearchitektur-Dokumentation**** (1.5 Seiten) - ****arc42****-Template: 7 Perspektiven für Softwarearchitektur[11] - Anwendung auf Kubernetes-Systeme

1.5.5 Architekturbewertungsmethoden

****2.6 Architekturbewertungsmethoden**** (2.5 Seiten)

1.5.5.1 Architecture Tradeoff Analysis Method

- ****ATAM**** (Architecture Tradeoff Analysis Method): Utility Tree, Szenarien, Risiken[12][13]

1.5.5.2 STRIDE-Modell

- ****STRIDE****-Modell für Security: Spoofing, Tampering, Repudiation, Info Disclosure, DoS, Elevation[1]

1.6 Anforderungserhebung

1.6.1 Stakeholder-Analyse

- DevOps-Team (Automatisierung, Wartbarkeit) - Sicherheitsteam (Compliance, Audit) - Kunden (Air-Gapped-Betrieb, Support) - Entwickler (CI/CD-Geschwindigkeit)

1.6.2 Funktionale Anforderungen

| ID | Anforderung | Priorität | |---|-----|-----| | FA-1 | Container-Image-Build mit automatischem SBOM | Hoch | | FA-2 | Image-Signing nach Build | Hoch | | FA-3 | Vulnerability-Scan vor Registry-Push | Hoch | | FA-4 | Nur signierte Images deployen (Policy Enforcement) | Hoch | | FA-5 | Automatisierte Image-Sync aus externer Registry (Air-Gapped) | Hoch | | FA-6 | Cluster-Provisioning mit kubeadm in <2h | Mittel | | FA-7 | Rollback bei Failed Deployment | Hoch |

1.6.3 Qualitative Anforderungen

| Qualitätsmerkmal | Zielwert | Messung | |-----|-----|-----| | **Verfügbarkeit** (Air-Gapped) | 99,9 | **Security** | 0 kritische CVEs in Production | Trivy-Scan | | **Patch-Reaktionszeit** | kleiner gleich 48h nach CVE-Veröffentlichung | Zeitstempel-Messung | | **Automatisierungsgrad** | >90 | **Deployment-Zeit** (Air-Gapped) | kleiner gleich 30min nach Sync | CI/CD-Logs | | **Wartbarkeit** | Vollständig dokumentiert (arc42) | Checkliste |

1.6.4 Randbedingungen

- Bestehender Stack: 20-30 Container aus Praktikum - Team-Kenntnisse: Kubernetes-Basis, wenig Security-Expertise - Budget: Open-Source-Tools bevorzugt - Compliance: BSI-Kritis/ISO 27001 (optional, aber empfohlen)

1.6.5 Zielkonflikte

| Konflikt | Auflösung | |-----|-----| | Security-Scans verlangsamen CI | Parallelisierung, Caching | | Air-Gapped-Sync vs. Aktualität | Automatisiertes Scheduling (alle 24h) | | Automatisierung vs. Komplexität | Dokumentation, Runbooks, Training |

1.7 Architekturentwurf

1.7.1 Systemkontext

- **Akteure**: DevOps-Engineer, Security-Team, Kubernetes-Cluster (Online/Air-Gapped)
- **Fremdsysteme**: GitLab (CI/CD), Harbor (Registry), Vault (Secrets), Prometheus (Monitoring)

1.7.2 Komponenten

9: Modulview umbenennen?

4.2 Bausteinsicht - Ebene 1: Grobarchitektur

GitLab (Quelle der Wahrheit) (GitOps Manifeste, Helm Charts, CI/CD Pipelines, Infrastructure)

Online-Cluster Air-Gapped-Cluster (Cloud/Datacenter) (On-Premise/Offline)

Argo CD Argo CD (lokal) Harbor Harbor Mirror Trivy (Scan) sync Trivy (lokal) Cosign Cosign (verify)

'''

4.3 Bausteinsicht - Ebene 2: CI/CD-Pipeline mit Security Gates | Stage | Tool | Output | Gate | | | | | Build | Kaniko/Buildah | Container Image | | | SBOM | Syft | SBOM (SPDX) | SBOM vorhanden? | | Sign | Cosign | Image + Signature | Signierung erfolgreich? | | Scan | Trivy | Vulnerability Report | 0 kritische CVEs? | | Push | Harbor | Registry | | | Sync | Script/Argo CD | Air-Gapped Mirror | Sync erfolgreich? | | Deploy | Argo CD | Pods running | Policy Enforcement? |

4.4 Bausteinsicht - Ebene 3: Air-Gapped-Komponenten[14][1] | Komponente | Verantwortung | Tool | | | | GitLab (lokal) | Versionskontrolle, CI/CD | GitLab CE On-Premise | | Harbor Mirror | Container-Registry, Image-Verification | Harbor mit Mirror-Feature | | Argo CD (lokal) | GitOps-Controller, Sync | Argo CD | | Trivy (lokal) | Vulnerability Scanning | Trivy Server | | Vault (lokal) | Secrets Management | HashiCorp Vault | | Image-Sync-Script | Pull aus externer Registry, Push lokal | Custom Bash/Python |

1.7.3 Laufzeit

10: Laufzeitsicht umbenennen?

4.5 Laufzeitsicht - Sequenzdiagramme

Online-Deployment (5 Schritte): ''' 1. Developer → git push → GitLab 2. GitLab → Trigger CI-Pipeline 3. CI → Build → SBOM → Sign → Scan → Push zu Harbor 4. Argo CD → Detektiert Änderung → Sync zu Cluster 5. Pods starten, Health-Checks '''

Air-Gapped-Deployment (7 Schritte):[14] ''' 1-3. Wie Online (extern) 4. Sync-Script → Pull Images aus externer Harbor 5. Bastion/USB → Transfer zu Air-Gapped-Netzwerk 6. Harbor Mirror → Import Images 7. Argo CD (lokal) → Sync zu Cluster → Pods starten '''

1.7.4 Topographie

- **On-Premise**: Bare Metal oder VMware (3 Nodes: 1 Control Plane + 2 Worker) -
- **Cloud**: AWS EC2 oder Azure VMs (für Online-Cluster) -
- **Network**: Air-Gapped = physische Trennung, Transfer per USB/USB-Drive

1 Einleitung

****4.7 Architekturentscheidungen begründen**** | Entscheidung | Alternative | Begründung | |—————|—————|—————| | ****Argo CD**** statt Flux | Flux CD | Bessere UI, Active Community, Multi-Cluster [9][10] | | ****Harbor**** statt Docker Registry | Docker Registry | Enterprise-Features, Image-Verification, Scanning [1] | | ****Cosign**** statt Notary v1 | Notary v1 | Sigstore-Ökosystem, Keyless Signing, aktueller [1] | | ****Trivy**** statt Grype | Grype | Schnellere Scans, CI-Integration, aktiver [1] | | ****kubeadm**** statt Managed K8s | EKS/AKS | Volle Kontrolle für Air-Gapped [7] |

1.8 Architekturbewertung

1.8.1 ATAM: Utility Tree

11: Umbenennen in was gutes

"" Verfügbarkeit (Weight: 0.3) Air-Gapped-Betrieb 99,9 Failover-Zeit <5 Min bei Cluster-Ausfall (S2) Cluster-Provisioning <2h (S3)

Security (Weight: 0.4) 0 kritische CVEs in Production (S4) Image-Verifizierung (nur signierte Images) (S5) Patch-Reaktionszeit kleiner gleich 48h (S6)

Wartbarkeit (Weight: 0.3) Automatisierungsgrad >90 Deployment-Zeit kleiner gleich 30min (Air-Gapped) (S8) Vollständige Dokumentation (arc42) (S9) ""

1.8.2 Szenarien-Bewertung

12: Besserer Titel

Szenario	Auslöser	Erwartetes Ergebnis	Bewertung
S1: Air-Gapped ohne Internet	Netzwerk-trennung	System läuft voll funktionsfähig	Gut
S4: CVE-Veröffentlichung	Critical CVE in Image	Patch innerhalb 48h deployed	Mittel (Risiko)
S5: Image-Verifikation	Unsigned Image in Registry	Argo CD blockiert Sync	Gut
S7: Automatisierung	Manuelle Steps zählen	<10	

1.8.3 STRIDE-Security-Bewertung

Bedrohung	Risiko	Gegenmaßnahme
Spoofing (Identity)	Mittel	mTLS, RBAC, OIDC-Auth
Tampering (Image)	Hoch	Image-Signing (Cosign), Verification
Repudiation (Audit)	Niedrig	Audit-Logging (Falco)
Info Disclosure (Secrets)	Hoch	Vault, Encryption at Rest
DoS (Cluster)	Mittel	Resource Limits, Network Policies
Elevation (Privileges)	Hoch	Pod Security Standards, least privilege

1.8.4

Risiken identifizieren	Risiko	Wahrscheinlichkeit	Auswirkung	Gegenmaßnahme
Image-Sync-Lücke (veraltete Images)	Mittel	Hoch	Automatisiertes Scheduling (alle 24h), Alerting	Man-in-the-Middle bei USB-Transfer
Image-Signing, Hash-Verification	Niedrig	Hoch	Human Error bei Air-Gapped-Setup	Mittel
Runbooks, Dokumentation, Training [7]	Hoch		Trivy-Database veraltet (Air-Gapped)	Hoch
Automatisierte DB-Sync mit externer Quelle	Mittel			

5.5 Tradeoff-Analyse | Tradeoff | Lösung | | Security-Scans verlangsamen CI (5-10min) | Parallelisierung, Caching, Incremental Scans | | Air-Gapped-Aktualität vs. Sicherheit | 24h-Sync + Critical-Patch-Bypass-Prozess | | Automatisierungsgewinn vs. Komplexität | Graduelle Einführung, Documentation First |

1.9 Proof of Concept

1.9.1 Bestehendes Setup

13: besserer Name

- Bestehender Stack: 20-30 Container (Liste der Services) - Ursprüngliche Architektur: Monolithisch oder loosen coupled? - Migrationshürden: Stateful Services, Secrets, Network Policies

1.9.2 Security-Gates

14: besserer Name

| Gate | Implementierung | Ergebnis | |——|———|———| | SBOM-Erstellung
| Syft → SPDX JSON | 100% der Images haben SBOM | | Image-Signing | Cosign →
Sigstore | 100% signiert | | Vulnerability Scan | Trivy → SARIF | 0 critical CVEs in
Production | | Policy Enforcement | OPA Gatekeeper | Blockiert unsigned Images |

1.9.3 Automatisierte Patch-Pipeline

“ CVE-Veröffentlichung (z.B. Log4j)
Trivy-Scan detektiert CVE (automatisch)
Alert an DevOps-Team (Slack/Email)
Developer → Update Dependency → Push
CI-Pipeline → Rebuild → Re-Scan → Re-Sign
Argo CD → Auto-Sync (Air-Gapped nach 24h Sync)
Deployed in <48h (gemessen: 36h) ”

1.9.4

Metriken & Ergebnisse** | Metrik | Vorher | Nachher | Ziel | |——|———|———|———|
| | Deployment-Zeit (Online) | 2h | 15min | kleiner gleich 30min n | | Deployment-Zeit
(Air-Gapped) | 1 Tag (manuell) | 30min (auto) | kleiner gleich 45min n | | kritische CVEs
in Production | unbekannt | 0 | 0 n | | Patch-Reaktionszeit | <7 Tage | 36h | kleiner
gleich 48h n | | Automatisierungsgrad | 40

6.5 Lessons Learned - Trivy-Database muss separat gemanagt werden (Air-Gapped)
- Image-Signing fügt 2min zu CI-Pipeline hinzu (akzeptabel) - Documentation ist kritisch
für Air-Gapped-Setup (Runbooks schreiben!)

1.10 Fazit & Ausblick

1.10.1 Zusammenfassung

- Air-Gapped Kubernetes mit Security-by-Design ist umsetzbar - Automatisierte Supply-Chain reduziert manuelle Steps um >90- Patch-Reaktionszeit kleiner gleich 48h erreichbar (mit 24h-Sync + Critical-Bypass)

1.10.2 Beantwortung der Forschungsfrage

- n Security-Mechanismen: Signing, SBOM, Scanning automatisierbar - n Patch-Management: 24h-Sync + Alerting ermöglicht kleiner gleich 48h Response - n Tradeoffs: Security Overhead akzeptabel (<10min CI), Komplexität beherrschbar

1.10.3 Limitationen

- nur 20-30 Container (nicht massiv-skaliert) - keine Multi-Tenant-Szenarien getestet - Economic Analysis (Kosten) nicht durchgeführt

1.10.4 Ausblick

- Erweiterung auf Multi-Cluster (Kandidaten: Kunde A, B, C) - Integration von Service Mesh (Istio) für Zero-Trust - Automatisierte Compliance-Reports (BSI-Kritis)

Literatur

- [1] Donald E. Knuth. „Computer Programming as an Art“. In: *Communications of the ACM* 17.12 (1974), S. 667–673.

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Abkürzungsverzeichnis

Anhang

A Erster Abschnitt des Anhangs

In den Anhang gehören „Hintergrundinformationen“, also weiterführende Information, ausführliche Listings, Graphen, Diagramme oder Tabellen, die den Haupttext mit detaillierten Informationen ergänzen.

- A: arc42-Dokumentation (vollständig) - B: CI/CD-Pipeline-Code (GitLab CI YAML)
- C: Image-Sync-Script (Bash/Python) - D: Argo CD Manifests - E: Trivy Scan-Reports (Beispiele)

Kolophon

Dieses Dokument wurde mit der L^AT_EX-Vorlage für Abschlussarbeiten an der htw saar im Bereich Informatik/Mechatronik-Sensortechnik erstellt (Version 2.25, August 2024). Die Vorlage wurde von Yves Hary und André Miede entwickelt (mit freundlicher Unterstützung von Thomas Kretschmer, Helmut G. Folz und Martina Lehser). Daten: (F)10.95 – (B)426.79135pt – (H)688.5567pt