

Sicherheitsanforderung

Orchestrator

Deutsche Telekom Gruppe

Version	5.0
Datum	01.07.2025
Status	Freigegeben

Impressum

Herausgeber
Deutsche Telekom AG
Vorstandsbereich Technology & Innovation
Chief Security Officer

Reuterstrasse 65, 53315 Bonn
Deutschland

Dateiname	Dokumentennummer 3.34	Dokumententyp Sicherheitsanforderung
Version 5.0	Stand 01.07.2025	Status Freigegeben
Fachlicher Ansprechpartner Telekom Security psa.telekom.de	Gültigkeitsdauer 01.07.2025 - 30.06.2030	Freigegeben von Stefan Pütz, Leiter SEC-T-TST

Zusammenfassung

Dieses Dokument beschreibt die funktionalen Sicherheitsanforderungen, die zum Sichern von Orchestratoren verwendet werden, zusätzlich zu den CIS-Benchmark- und Hardening-Handbüchern, die auch die implementierungsorientierten Anforderungen abdecken.

Copyright © 2025 by Deutsche Telekom AG.

Alle Rechte, auch die des auszugsweisen Nachdrucks, der fotomechanischen Wiedergabe (einschließlich Mikrokopie) sowie der Auswertung durch Datenbanken oder ähnliche Einrichtungen, vorbehalten.

Inhaltsverzeichnis

1.	Einleitung	4
2.	Requirements	5
2.1.	Allgemein	5
2.1.1.	Control plane hardening	6
2.1.2.	Worker Nodes	8
2.2.	Secrets	10
2.3.	Network	12
2.4.	Authentication and authorization	15
2.5.	Audit Logging / Threat Detection	16
2.6.	Pod Sicherheit	17
2.7.	Deployment/Application Level Security	19
3.	Private Clouds	25
3.1.	Cloud-Infrastruktur-Netzwerk und Cloud-Management	25
3.1.1.	Rund um das Cloud Management	25
3.2.	Cloud-Nutzung	26
3.2.1.	Zweckgebundene Trennung der Cloud-Infrastruktur	26

1. Einleitung

Dieses Dokument wurde auf Basis der Vorgaben aus den im Konzern gültigen Sicherheitsrichtlinien erstellt.

Die Sicherheitsanforderung dient u.a. als Grundlage zur Freigabe im PSA-Prozess. Sie dient darüber hinaus als Umsetzungsempfehlung für Einheiten, die nicht am PSA-Verfahren teilnehmen. Diese Anforderungen müssen bereits während der Planungs- und Entscheidungsprozesse berücksichtigt werden. Bei der Umsetzung der Sicherheitsanforderung ist das jeweils vorrangige nationale, internationale und supranationale Recht zu beachten.

Sollte die Einhaltung der beschriebenen Anforderungen in individuellen Fällen nicht oder nur eingeschränkt umsetzbar sein, sind gemeinsam mit einem Sicherheits- und/oder Datenschutzexperten (entsprechend der betroffenen Anforderungen) Risikobewertungen vorzunehmen und mögliche alternative Schutzmaßnahmen abzustimmen.

2. Requirements

2.1. Allgemein

Es kann eine geteilte Verantwortung geben, abhängig von der Art des Clusters.

Selbstverwaltetes Kubernetes Cluster-Administratoren haben die volle Kontrolle über alle Komponenten.

Verwaltetes Kubernetes (EKS, GKE, AKS, Drittanbieter, Telekom Managed) Cluster-Administratoren haben eingeschränkte oder keine Kontrolle über die Komponenten der Steuerungsebene (API-Server, Controller Manager, etc.). Nur der Cloud-Anbieter/Provider verwaltet die Steuerungsebene. Cluster-Administratoren können Worker-Knoten und einige Konfigurationen wie CoreDNS, kube-proxy und Netzwerkrichtlinien ändern.

Komponenten des Worker-Knotens (z.B. Kubelet, Container Runtime Interface) In selbstverwalteten Clustern haben Administratoren vollen Zugriff. In Cloud-verwalteten/verwalteten Clustern ist der Zugriff eingeschränkt (z.B. AWS beschränkt Änderungen an der Kubelet-Konfiguration auf EKS-verwalteten Knoten).

Aufschlüsselung der Kubernetes-Komponenten und Zugriffskontrolle (kann je nach Anbieter variieren - dies als Beispiel)

Component	Self-Managed Kubernetes (Cluster Admins)	Managed Kubernetes (AKS, GKE, EKS, 3rd Party, Telekom Managed)
API Server	Volle Kontrolle	Kein Zugriff
Controller Manager	Volle Kontrolle	Kein Zugriff
Cloud Controller Manager	Volle Kontrolle	Kein Zugriff
Scheduler	Volle Kontrolle	Kein Zugriff
etcd (Cluster Database)	Volle Kontrolle	Kein Zugriff(Ausnahme , möglicherweise Snapshot Funktion der Cloud)
CoreDNS, Other DNS	Volle Kontrolle	Limitierter Zugriff(Konfiguration kann ggf. geändert werden)
kube-proxy	Volle Kontrolle	Limitierter Zugriff(Konfiguration kann ggf. geändert werden)
Kubelet (node agent)	Volle Kontrolle	Limitierter Zugriff(Konfiguration kann ggf. geändert werden, Provider Betriebssystem, Kubelet Konfiguration)
Container Runtime	Volle Kontrolle	Limitierter Zugriff(Runtime spezifische Konfiguration kann ggf. verändert werden)

Req 1	Es muss einen Patch-Management-Prozess für die gesamte Cloud-Umgebung geben.
-------	--

Der Patchmanagementprozess in einer Cloud-Umgebung muss so implementiert werden, dass das Patchen aller Cloud-Komponenten unter Einhaltung der SLA für die Mandanten möglich ist. Wenn die SLA dies erfordert, muss der Patching-Prozess so konzipiert sein, dass eine Live-Migration virtueller Maschinen für das Patching von Rechenknoten möglich ist, oder dass Cloud-Management-APIs während des Patchings zur Verfügung stehen, so dass die Mandanten in der Lage sind, Arbeitslasten, die von der Aktualisierung betroffen sind, neu zu planen. Bei anderen SLA-Typen könnte man sich auch dafür entscheiden, die gesamte Cloud-Umgebung auf einmal zu aktualisieren und alle Arbeitslasten neu zu starten.

Alle Komponenten der Cloud-Umgebung müssen in das Patch-Management-Verfahren einbezogen werden, z. B. Ma-

nagement, Orchestrierung, Rechen-, Speicher- und Netzwerkfunktionen. Alle unterstützenden Systeme (Überwachung, Protokollierung, Jump Hosts usw.) müssen ebenfalls in das Verfahren einbezogen werden.

Auch in solchen Fällen, in denen eine kurzfristige Korrektur erforderlich ist, muss das reguläre Prüfverfahren eingehalten werden.

Motivation: Da täglich neue Sicherheitslücken auftauchen, ist es notwendig, den Patching-Prozess so zu gestalten, dass er schnell durchgeführt werden kann.

Umsetzungsbeispiel: Das Patching-Verfahren sollte so gestaltet werden, dass das Patching aller Rechenknoten innerhalb von 24-48 Stunden nach Aufdeckung einer kritischen Sicherheitslücke und Verfügbarkeit eines Patches abgeschlossen werden kann.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigter Zugang zum System
- Unberechtigte Einsichtnahme oder Abgriff von Daten
- Unberechtigte Modifikation von Daten
- Unberechtigte Nutzung von Diensten oder Ressourcen
- Störung der Verfügbarkeit
- Abstreiten von durchgeführten Aktionen
- Unbemerkt durchführbare Angriffe
- Angriffe motiviert und erleichtert durch Informationspreisgabe oder erkennbare Sicherheitsschwachstellen

ID: 3.86-17/2.0

2.1.1. Control plane hardening

Die Control plane besteht aus den folgenden logischen Komponenten, die gehärtet werden müssen:

- Controller manger
- Cloud controler manager
- API
- Etcd
- Scheduler

Req 2	Die Konfigurationsdateien des Control Plane Nodes müssen auf die restriktivsten Berechtigungen festgelegt werden.
-------	---

Diese Empfehlungen gelten möglicherweise nicht direkt für Cluster-Betreiber in Umgebungen, in denen diese Komponenten von einem Drittanbieter verwaltet werden. Diese Anforderung kann nur vom Cluster-Operator/Administrator erfüllt werden.

Stellen Sie sicher, dass die Pod-Spezifikationsdatei des API-Servers über Berechtigungen von "600" oder restriktiver verfügt.

Stellen Sie sicher, dass der Besitz der API-Server-Pod-Spezifikationsdatei auf "root:root" festgelegt ist.

Stellen Sie sicher, dass die Pod-Spezifikationsdatei des Controller-Managers über Berechtigungen von "600" oder restriktiver verfügt.

Stellen Sie sicher, dass der Besitz der Pod-Spezifikationsdatei des Controller-Managers auf "root:root" festgelegt ist.

Stellen Sie sicher, dass die Scheduler-Pod-Spezifikationsdatei über Berechtigungen von "600" oder restriktiver verfügt.

Stellen Sie sicher, dass der Besitz der Scheduler-Pod-Spezifikationsdatei auf 'root:root' festgelegt ist.

Stellen Sie sicher, dass die Datei "/etc/kubernetes/manifests/etcd.yaml" über die Berechtigungen "600" verfügt.

Stellen Sie sicher, dass der Dateibesitz '/etc/kubernetes/manifests/etcd.yaml' auf 'root:root' festgelegt ist.

Stellen Sie sicher, dass die Container-Netzwerkschnittstellendateien über Berechtigungen von "600" oder restriktiver verfügt.

Stellen Sie sicher, dass die Eigentümerschaft für die Container-Netzwerkschnittstellendateien auf 'root:root' festgelegt ist.

Stellen Sie sicher, dass das etcd-Datenverzeichnis über die Berechtigungen '700' verfügt.

Stellen Sie sicher, dass der Besitz des etcd-Datenverzeichnisses auf 'etcd:etcd' gesetzt ist.

Stellen Sie sicher, dass die Datei 'admin.conf' über die Berechtigungen '600' verfügt.

Stellen Sie sicher, dass der Dateibesitz 'admin.conf' auf 'root:root' gesetzt ist.

Stellen Sie sicher, dass die Datei 'scheduler.conf' die Berechtigungen '600' hat.
Stellen Sie sicher, dass der Dateibesitz der Datei 'scheduler.conf' auf 'root:root' gesetzt ist.
Stellen Sie sicher, dass die Datei 'controller-manager.conf' Berechtigungen von 600 .
Stellen Sie sicher, dass der Eigentümer der Datei 'controller-manager.conf' auf 'root:root' gesetzt ist.
Stellen Sie sicher, dass der Besitz des Kubernetes-PKI-Verzeichnisses und der Datei auf "root:root" festgelegt ist.
Stellen Sie sicher, dass Kubernetes PKI-Zertifikatsdateien über Berechtigungen von "600" oder restriktiver verfügt.
Stellen Sie sicher, dass Kubernetes PKI-Schlüsseldateien über die Berechtigungen "600" verfügen.

Motivation: Das setzen von restriktiven Berechtigungen für Konfigurationsdateien der Control-Plane-Knoten minimiert das Risiko unbefugten Zugriffs oder Manipulationen.

ID: 3.34-2/5.0

Req 3 Der API-Server muss gehärtet sein.

In einer typischen Orchestrierungsumgebung können Clusteradministratoren die API-Konfiguration ändern. Sie verwalten die Kubernetes-Steuerungsebene, einschließlich des API-Servers. Sie haben direkten Zugriff auf die Änderung von API-Serverkonfigurationen (z. B. Flags, Sicherheitseinstellungen, Zugangscontroller).
In verwalteten Kubernetes-Systemen (EKS, AKS, GKE, 3rd Party usw.) steuert der Anbieter die API-Serverkonfiguration und schränkt ein, was Administratoren ändern können.

API-Server-Konfigurationsflags für die Härtung, die festgelegt werden müssen:

Deaktivieren Sie anonyme Anfragen an den API-Server.

Verwenden Sie keine tokenbasierte Authentifizierung.

Aktivieren Sie die zertifikatbasierte Kubelet-Authentifizierung.

Überprüfen Sie das Zertifikat von Kubelet, bevor Sie eine Verbindung herstellen.

Beschränken Sie Kubelet-Knoten so, dass nur Objekte gelesen werden, die ihnen zugeordnet sind.

Aktivieren Sie die rollenbasierte Zugriffssteuerung.

Begrenzen Sie die Rate, mit der der API-Server Anforderungen akzeptiert.

Images sollten immer gepullt werden.

Aktivieren Sie die Überwachung auf dem Kubernetes-API-Server und legen Sie den gewünschten Pfad für das Überwachungsprotokoll fest. Bewahren Sie die Protokolle mindestens 30 Tage lang oder je nach Bedarf auf.

Legen Sie das globale Anforderungstimeout für API-Serveranforderungen entsprechend fest.

Überprüfen Sie das Dienstkonto, bevor Sie das Token überprüfen.

Legen Sie explizit eine Datei mit dem öffentlichen Schlüssel für Dienstkonten auf dem API-Server fest.

Etcd sollte so konfiguriert werden, dass die TLS-Verschlüsselung für Clientverbindungen verwendet wird.

Richten Sie eine TLS-Verbindung auf dem API-Server ein.

Etcd sollte so konfiguriert werden, dass die TLS-Verschlüsselung für Clientverbindungen verwendet wird.

Verschlüsseln Sie den etcd-Schlüssel-Wert-Speicher.

Wenn die "etcd"-Verschlüsselung verwendet wird, sollten geeignete Anbieter konfiguriert werden.

Motivation: Das Absichern des API-Servers verstärkt den Schutz vor potenziellen Angriffen und gewährleistet die Sicherheit und Stabilität des Kubernetes-Clusters.

ID: 3.34-3/5.0

Req 4 Der Controller Manager / Cloud Controller Manager muss gehärtet sein.

Clusteranbieter/Clusteradministratoren sind für die Erfüllung dieser Anforderung verantwortlich.

In **verwaltetem Kubernetes (EKS, GKE, AKS, 3rd Party)** werden diese Komponenten vom Anbieter verwaltet, und Administratoren haben möglicherweise nur eingeschränkten oder keinen Zugriff, um sie zu ändern.

Controller Manager-Konfigurationsflags, die festgelegt werden müssen:

Aktivieren Sie den Garbage Collector bei der Pod-Beendigung.

Deaktivieren Sie die Profilerstellung, wenn sie nicht benötigt wird.

Verwenden Sie für jeden Controller individuelle Anmeldeinformationen für das Dienstkonto.

Legen Sie explizit eine Datei mit dem privaten Schlüssel für Dienstkonten für Dienstkonten auf dem Controller-Manager fest.

Erlauben Sie Pods, das Serving-Zertifikat des API-Servers zu überprüfen, bevor sie Verbindungen herstellen.

Aktivieren Sie die Rotation von Kubelet-Serverzertifikaten auf dem Controller-Manager.
Binden Sie den Controller-Manager-Dienst nicht an unsichere Adressen, die keine Loopbacks sind.

Motivation: Die Absicherung des Controller Managers und des Cloud Controller Managers erhöht deren Widerstandsfähigkeit gegen Angriffe und gewährleistet eine sichere Orchestrierung von Kubernetes-Ressourcen.

ID: 3.34-4/5.0

Req 5 Der Scheduler muss gehärtet werden.

Clusteranbieter/Clusteradministratoren sind für die Erfüllung dieser Anforderung verantwortlich.

In **verwaltetem Kubernetes (EKS, GKE, AKS, 3rd Party)** werden diese Komponenten vom Anbieter verwaltet, und Administratoren haben möglicherweise nur eingeschränkten oder keinen Zugriff, um sie zu ändern.

Flags für die Scheduler-Konfiguration:

Deaktivieren Sie Profiling, falls es nicht benötigt wird

Binden Sie den Scheduler-Dienst nicht an unsichere Adressen, die keine Loopbacks sind.

Motivation: Die Absicherung des Schedulers erhöht die Sicherheit und gewährleistet eine zuverlässige und geschützte Planung von Arbeitslasten im Kubernetes-Cluster.

ID: 3.34-5/5.0

Req 6 Der Etcd-Dienst muss gehärtet sein.

Clusteranbieter/Clusteradministratoren sind für die Erfüllung dieser Anforderung verantwortlich.

In **verwaltetem Kubernetes (EKS, GKE, AKS, 3rd Party)** werden diese Komponenten vom Anbieter verwaltet, und Administratoren haben möglicherweise nur eingeschränkten oder keinen Zugriff, um sie zu ändern.

Etcd speichert Zustandsinformationen und Clustergeheimnisse. Etcd muss im Ruhezustand verschlüsselt werden.

Sichern Sie den etcd-Datenspeicher mit Authentifizierungs- und RBAC-Richtlinien.

Konfigurieren Sie die TLS-Verschlüsselung für den etcd-Dienst.

Aktivieren Sie die Clientauthentifizierung für den etcd-Dienst. Verwenden Sie keine selbstsignierten Zertifikate für TLS.

etcd sollte so konfiguriert werden, dass die TLS-Verschlüsselung für Peer-Verbindungen verwendet wird.

Etcd sollte für die Peerauthentifizierung konfiguriert werden.

Verwenden Sie keine automatisch generierten selbstsignierten Zertifikate für TLS-Verbindungen zwischen Peers.

Motivation: Die Absicherung von Etcd gewährleistet die Integrität und Vertraulichkeit der Clusterdaten und schützt vor unbefugtem Zugriff und möglichen Sicherheitsverletzungen.

ID: 3.34-6/5.0

Req 7 TLS muss aktiviert sein.

TLS muss für jede Komponente aktiviert sein, die dies unterstützt, um Traffic Sniffing zu verhindern.

3_50_Kryptographische_Algorithmen Dokument, Es muss TLS in der Version 1.2 oder 1.3 verwendet werden .

Motivation: Gewährleistet den Datenschutz und Datenintegrität zwischen zwei oder mehreren Systemen.

Umsetzungsbeispiel: Orchestratorkomponenten müssen TLS für die Kommunikation verwenden.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigte Einsichtnahme oder Abgriff von Daten
- Unberechtigte Modifikation von Daten

ID: 3.34-7/5.0

2.1.2. Worker Nodes

Req 8 Die Dateiberechtigungen für Worker Nodes müssen restriktiv festgelegt werden.

Clusteranbieter/Clusteradministratoren sind für die Erfüllung dieser Anforderung verantwortlich. In **verwaltetem Kubernetes (EKS, GKE, AKS, 3rd Party)** werden diese Komponenten vom Anbieter verwaltet, und Administratoren haben möglicherweise nur eingeschränkten oder keinen Zugriff, um sie zu ändern.

Konfigurationsdateien auf den Worker-Knoten:

Stellen Sie sicher, dass die Dateiberechtigungen des Kubelet-Diensts auf 600 festgelegt sind.

Stellen Sie sicher, dass der Besitz der kubelet-Dienstdatei auf root:root festgelegt ist.

Wenn die Proxy-kubeconfig-Datei vorhanden ist, stellen Sie sicher, dass die Berechtigungen auf 600 festgelegt sind.

Wenn die Proxy-kubeconfig-Datei vorhanden ist, stellen Sie sicher, dass die Eigentümerschaft auf root:root festgelegt ist.

Stellen Sie sicher, dass die Dateiberechtigungen --kubeconfig kubelet.conf auf 600 festgelegt sind.

Stellen Sie sicher, dass der Dateibesitz --kubeconfig kubelet.conf auf root:root festgelegt ist.

Stellen Sie sicher, dass die Dateiberechtigungen der Zertifizierungsstellen auf 600 festgelegt sind.

Stellen Sie sicher, dass der Dateibesitz der Clientzertifizierungsstellen auf root:root festgelegt ist.

Wenn die Konfigurationsdatei kubelet config.yaml verwendet wird, überprüfen Sie, ob die Berechtigungen auf 600 festgelegt sind.

Wenn die Konfigurationsdatei kubelet config.yaml verwendet wird, ist der Besitz der Validierungsdatei auf root:root festgelegt.

Motivation: Das Setzen restriktiver Dateiberechtigungen auf Worker Nodes verhindert unbefugten Zugriff und gewährleistet die Sicherheit von Workloads und sensiblen Daten.

ID: 3.34-8/5.0

Req 9 Der Kubelet-Dienst muss gehärtet sein.

Clusteranbieter/Clusteradministratoren sind für die Erfüllung dieser Anforderung verantwortlich.

In **verwaltetem Kubernetes (EKS, GKE, AKS, 3rd Party)** werden diese Komponenten vom Anbieter verwaltet, und Administratoren haben möglicherweise nur eingeschränkten oder keinen Zugriff, um sie zu ändern.

Härtung der Kubelet-Konfiguration:

Stellen Sie sicher, dass das Argument --anonymous-auth auf false gesetzt ist.

Stellen Sie sicher, dass das Argument --authorization-mode nicht auf AlwaysAllow festgelegt ist.

Stellen Sie sicher, dass das Argument --client-ca-file entsprechend festgelegt ist.

Stellen Sie sicher, dass das Argument --read-only-port auf 0 festgelegt ist.

Stellen Sie sicher, dass das Argument --streaming-connection-idle-timeout nicht auf 0 festgelegt ist.

Stellen Sie sicher, dass das Argument --make-iptables-util-chains auf true gesetzt ist.

Stellen Sie sicher, dass das Argument --hostname-override nicht festgelegt ist.

Stellen Sie sicher, dass das eventRecordQPS-Argument auf eine Ebene festgelegt ist, die eine angemessene Ereigniserfassung gewährleistet.

Stellen Sie sicher, dass die Argumente --tls-cert-file und --tls-private-key-file entsprechend festgelegt sind.

Stellen Sie sicher, dass das Argument --rotate-certificates nicht auf false gesetzt ist.

Stellen Sie sicher, dass das Argument RotateKubeletServerCertificate auf true festgelegt ist.

Stellen Sie sicher, dass ein Grenzwert für Pod-PIDs festgelegt ist.

Motivation: Die Absicherung des Kubelets gewährleistet eine sichere Kommunikation und Kontrolle über Node-Operationen und schützt den Cluster vor potenziellen Schwachstellen.

ID: 3.34-9/5.0

Req 10 Management Nodes und Worker Nodes müssen auf dedizierten Hosts laufen.

Ein Orchestrator muss mit mindestens einem Master Node und zwei Worker Nodes auf dedizierten Hosts eingerichtet sein (entweder physische, oder virtuelle Maschine).

Motivation: Master Nodes sind für den Betrieb des Clusters von entscheidender Bedeutung. Wenn kein Master Node läuft, kann der Cluster keine Anwendungen schedulen und ausführen.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigter Zugang zum System
- Unberechtigte Einsichtnahme oder Abgriff von Daten
- Unberechtigte Modifikation von Daten
- Unberechtigte Nutzung von Diensten oder Ressourcen

ID: 3.34-10/5.0

2.2. Secrets

Req 11	Tenant Network Protection und Secret Management der Plattform müssen, falls verfügbar, verwendet werden.
--------	--

Jeder Tenant ist verpflichtet, Netzwerkschutzmechanismen und Secret Management entsprechend den verfügbaren Technologien der Kubernetes-Plattform ordnungsgemäß zu verwenden.

Dies sind zwei der häufigsten Bereitstellungsszenarien für Netzwerkschutzdienste in Kubernetes-Umgebungen:

- Netzwerkschutzdienste werden von der Kubernetes-Plattform bereitgestellt (z. B. Netzwerkrichtlinien, Service Meshes oder CNI-Plug-ins mit erweiterten Sicherheitsfunktionen), und der Mandant konfiguriert den Netzwerkschutz auf geeignete Weise für seine Anwendungen. Dies ist die bevorzugte Schutzmethode.
- Netzwerkschutzdienste werden nicht von der Kubernetes-Plattform bereitgestellt oder sie werden bereitgestellt, aber der Mandant kann sie nicht verwenden. In diesem Fall sollte der Mandant dedizierte Knoten für die Ausführung von Netzwerkschutzmechanismen (z. B. dedizierte Sicherheitspods oder DaemonSets, auf denen Netzwerksicherheitstools ausgeführt werden) nach der Bewertung durch den zuständigen Platform Security Manager bereitstellen.

In ähnlicher Weise benötigen Mandanten in Kubernetes in der Regel eine geheime Verwaltung (z. B. Schlüssel, Kennwörter, Zertifikate) und andere Verwaltungs- und Supportfunktionen (z. B. Konfigurationsmanagement), die bei Bedarf verfügbar sind, um ihre Workloads zu unterstützen. Daher gibt es zwei gängigste Bereitstellungsszenarien:

- Die Geheimnisverwaltung wird von der Kubernetes-Plattform bereitgestellt (unter Verwendung nativer Kubernetes Secrets, externer Secret Stores wie HashiCorp Vault, der in Kubernetes integriert ist, oder Kubernetes Secrets Store CSI Driver), und der Mandant verwendet sie entsprechend für seine Anwendungen. Dies ist die bevorzugte Methode (dies gilt auch für den Fall, dass Management-Services separat verfügbar sind - nicht direkt Teil der Kubernetes-Plattform, sondern über Operatoren oder benutzerdefinierte Ressourcen verwendet werden können).
- Die Verwaltung von Geheimnissen und zusätzliche Verwaltungsfunktionen werden von der Kubernetes-Plattform nicht bereitgestellt oder werden bereitgestellt, aber der Mandant kann sie nicht verwenden. In diesem Fall muss der Mandant dedizierte Knoten für die Ausführung der Geheimnisverwaltung und anderer notwendiger Dienste bereitstellen, nachdem der zuständige Platform Security Manager dies bewertet hat.

Motivation: Die Sicherheit von Anwendungen in Kubernetes ist nicht gewährleistet, wenn Sicherheitsfunktionen nicht richtig genutzt werden.

Implementierungsbeispiel: Verwendung von Sicherheitsdiensten, die von der Kubernetes-Plattform angeboten werden, wie z. B. Netzwerkrichtlinien, RBAC, Geheimnisse, Pod-Sicherheitsstandards und Service Mesh-Implementierungen, um Anwendungssicherheit zu erreichen. Nutzen Sie für komplexere Anforderungen Kubernetes-Operatoren, die sich in externe Sicherheitstools und -dienste integrieren lassen.

Motivation: Die Sicherheit von Anwendungen in der Cloud ist nicht gegeben, wenn die Sicherheitsfunktionen nicht richtig genutzt werden.

Umsetzungsbeispiel: Nutzung der von der Plattform angebotenen Sicherheitsdienste, um die Sicherheit der Anwendung zu gewährleisten.

ID: 3.34-11/5.0

Req 12	Ein Schlüsselverwaltungssystem (Key Management System - KMS) muss vorhanden sein und für den Umgang mit Schlüsselmaterial verwendet werden.
--------	---

Die Verwaltung von Schlüsselmaterial spielt eine entscheidende Rolle bei der Sicherheit von Kubernetes. Zu den Geheimnissen in diesem Sinne gehören:

- Schlüssel
- Passwörter
- Zertifikate
- Tokens

Hier finden Sie eine Liste von Anwendungsbeispielen für KMS:

- Speicherverschlüsselung
- Schlüsselverwaltung für SSH-Zugang
- Verschlüsselung der API-Kommunikation
- Verschlüsselung von Mandantendaten
- Bescheinigungen für Pächter
- geheime Verwaltung für Mandanten

Es ist praktisch unmöglich, eine sichere Cloud-Plattform oder sichere Tenant-Workloads in der Cloud ohne KMS zu betreiben. Das KMS kann als Teil der Cloud-Plattform oder separat bereitgestellt werden. Selbst wenn das KMS nicht als Teil der Cloud-Plattform bereitgestellt wird, ist es daher notwendig, dass das KMS erreichbar und auf automatisierte Weise nutzbar ist (wie andere Cloud-APIs).

Wenn ein KMS nicht verfügbar ist oder es nicht möglich ist, ein KMS zu verwenden, müssen die Secrets auf sichere Weise mit Tools wie SOPS (<https://github.com/getsops/sops>) verwaltet werden.

In einem solchen Fall, in dem KMS nicht genutzt werden kann und Tools wie SOPS verwendet werden, wird der Prozess entscheidend für die Aufrechterhaltung der Sicherheit, da die Automatisierung der Schlüsselverwaltung, Zugangskontrollen und Audit-Fähigkeiten, die ein KMS normalerweise bietet, durch manuelle oder halbautomatisierte Prozesse ersetzt wird. Es ist wichtig, Folgendes zu berücksichtigen:

Regelmäßiger Rotationsplan: Etablieren und befolgen Sie einen strikten Plan zur Schlüsselrotation (mindestens einmal jährlich).

Onboarding/Offboarding: Implementieren Sie Prozesse für den Umgang mit Schlüsseln (Rotation/Löschung) während Personalwechseln.

Re-Verschlüsselungsprozess: Definieren Sie klare Verfahren für die Neuverschlüsselung aller Secrets bei Schlüsseländerungen.

Notfallrotation: Dokumentieren Sie Verfahren für eine Notfallrotation von Schlüsseln im Falle einer Kompromittierung einschließlich vermuteter Kompromittierungen.

Bei Kompromittierung oder Verdacht auf Kompromittierung : Rotieren Sie alle Schlüssel/Secrets sofort.

Least Privilege : Gewähren Sie Zugriff nur an Teammitglieder, die diesen unbedingt benötigen.

Aktivitätsprotokollierung: Implementieren Sie eine umfassende Protokollierung aller Verschlüsselungs-/Entschlüsselungsaktivitäten, um eine unveränderliche Auditspur zu erstellen.

Git-Integration: Beim Speichern verschlüsselter Dateien in Git konfigurieren Sie .gitattributes, um versehentliche unverschlüsselte Commits zu verhindern.

Tools: Ziehen Sie die Verwendung von AGE-Verschlüsselung als einfachere Alternative zu PGP in Betracht, sofern möglich.

PGP-Schlüssel: Wenn PGP-Schlüssel anstelle eines KMS verwendet werden, stellen Sie sicher, dass eine ordnungsgemäße Schlüsselverwaltung mit Tools wie GnuPG erfolgt.

AGE (Actually Good Encryption) ist ein modernes Verschlüsselungstool, das entwickelt wurde, um einfach, sicher und benutzerfreundlich zu sein. Es verwendet modernste kryptografische Primitive wie X25519, ChaCha20-Poly1305 und HMAC-SHA256, um Sicherheit zu gewährleisten, während die Komplexität und die Altlasten älterer Tools wie PGP vermieden werden.

Motivation: KMS ermöglicht die Erzeugung, Speicherung und Verteilung von Geheimnissen in einer Cloud-Umgebung. Würden Geheimnisse auf Systemen in der Cloud im Klartext gespeichert, könnten sie möglicherweise von unbefugten Benutzern eingesehen werden.

Umsetzungsbeispiel: Eine Lösung wie HashiCorp Vault kann als Teil der Cloud-Infrastruktur installiert werden.

ID: 3.34-12/5.0

Req 13	Encryption-Keys müssen rotiert werden.
--------	--

Die symmetrischen Encryption-Keys, die der API-Server zum Verschlüsseln von etcd-Werten verwendet, werden nicht automatisch rotiert.
Sie müssen mindestens einmal im Jahr manuell rotiert werden.

Motivation: Orchestratoren (z. B. Kubernetes) rotieren einige Zertifikate automatisch. Orchestratoren tun dies nicht für alle Schlüssel, das muss überprüft werden.

ID: 3.34-13/5.0

2.3. Network

Req 14	Network und Namespace Policies müssen verwendet werden, um die Kommunikation einzuschränken.
--------	--

Eine restriktive Policy muss aktiviert sein (default-deny). Kommunikation zwischen Pods muss explizit auf eine Whitelist gesetzt werden.

Stellen Sie sicher, dass die verwendete CNI Netzwerkrichtlinien unterstützt. Stellen Sie sicher, dass für alle Namespaces Netzwerkrichtlinien definiert sind.

Motivation: Standardmäßig sind Pods nicht isoliert. Sie akzeptieren Datenverkehr aus jeder Quelle.

Pods werden isoliert, indem sie über eine NetworkPolicy verfügen, die sie auswählt.

Sobald in einem Namespace eine NetworkPolicy vorhanden ist, die einen bestimmten Pod auswählt, lehnt dieser Pod alle Verbindungen ab, die von keiner NetworkPolicy zugelassen werden. (Andere Pods im Namespace, die nicht von einer NetworkPolicy ausgewählt wurden, akzeptieren weiterhin den gesamten Datenverkehr.) Verwenden Sie eine Standardrichtlinie, um den gesamten ein- und ausgehenden Datenverkehr zu verweigern. Stellt sicher, dass nicht ausgewählte Pods in allen Namespaces außer kube-system isoliert sind.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigter Zugang zum System
- Unberechtigte Einsichtnahme oder Abgriff von Daten
- Unberechtigte Modifikation von Daten
- Unberechtigte Nutzung von Diensten oder Ressourcen

ID: 3.34-14/5.0

Req 15	Der Zugang zu Verwaltungsschnittstellen, z.B. der Kubernetes API, Dashboards und Systemen muss durch Firewalls oder Proxys auf höherer Ebene kontrolliert und geschützt werden (vorzugsweise versteckt hinter einem Jump Host), während die Kommunikation verschlüsselt sein muss.
--------	--

Gemäß der Anforderung, dass keine direkte IP-Konnektivität von externen Netzen und Tenant-Netzen zu Kubernetes-API-Endpunkten bestehen darf, wäre der Zugang für Verwaltungsfunktionen von in der Cloud laufenden Anwendungen und für externe Personen und/oder Maschinen erforderlich. Der Zugang zu Kubernetes-API-Endpunkten muss über Firewalls auf höherer Ebene (über L4 - TCP), WAFs oder Proxys erfolgen.

Die Kommunikation von Endkunden (sowohl von externen als auch von Mandanten-Netzwerken) zur Fire-

wall/WAF/zum Proxy und von der Firewall/WAF/dem Proxy zu den exponierten Cloud-API-Endpunkten muss verschlüsselt werden, einschließlich der Überprüfung von Zertifikaten in beide Richtungen, um Man-in-the-Middle-Angriffe zu verhindern. Interne Cloud/Kubernetes-APIs müssen für die Mandanten natürlich nicht erreichbar sein.

Für webbasierte Benutzeroberflächen (z. B. Dashboards) müssen spezielle Webanwendungsschecks verwendet werden.

Für den Zugang zu allen API-Endpunkten muss eine starke Authentifizierung verwendet werden, entsprechend den bestehenden Sicherheitsanforderungen für Webanwendungen und Webdienste (z. B. Dokument 3.02 - "Web Services").

Motivation: Cloud-API-Endpunkte haben eine große Angriffsfläche mit relativ geringem eingebautem Schutz gegen unerwartete Eingaben und potenziell großen Auswirkungen, wenn sie verletzt werden.

Umsetzungsbeispiel: Es wird bevorzugt, dass die gesamte administrative Kommunikation über den Jump-Host abgewickelt wird. Wenn ein Ansatz mit Jump Host nicht möglich ist, können strenge VPN oder ACLs verwendet werden, um den Zugriff auf administrative Cloud-Schnittstellen, APIs, Dashboards usw. nur für Cloud-Administratoren zu beschränken.

ID: 3.34-15/5.0

Req 16	Der Zugang zu Cloud-API-Endpunkten muss durch Firewalls oder Proxys auf höherer Ebene kontrolliert und geschützt werden, und zwar sowohl für den Zugang von externen Netzen als auch von Mandanten. Die Kommunikation muss verschlüsselt sein. Authentifizierung ist obligatorisch.
--------	---

Gemäß der Anforderung, dass keine direkte IP-Konnektivität von externen Netzen und Tenant-Netzen zu Cloud-API-Endpunkten bestehen darf, wäre der Zugang für Verwaltungsfunktionen von in der Cloud laufenden Anwendungen und für externe Personen und/oder Maschinen erforderlich. Der Zugang zu Cloud-API-Endpunkten muss über Firewalls auf höherer Ebene (über L4 - TCP), WAFs oder Proxys erfolgen.

Die Kommunikation von Endkunden (sowohl von externen als auch von Mandanten-Netzwerken) zur Firewall/WAF/zum Proxy und von der Firewall/WAF/dem Proxy zu den exponierten Cloud-API-Endpunkten muss verschlüsselt werden, einschließlich der Überprüfung von Zertifikaten in beide Richtungen, um Man-in-the-Middle-Angriffe zu verhindern. Interne Cloud-APIs müssen für die Mandanten natürlich nicht erreichbar sein.

Neben allen regulären APIs gilt dies auch für Nova Metadata Server. In typischen modernen Open-Stack-Implementierungen gibt es einen Nova-Metadaten-Proxy, der diese Anforderung für Nova-Metadaten-Server erfüllt. Derzeit gibt es keine Verschlüsselung für Nova-Metadaten, so dass dieser Endpunkt nicht verschlüsselt werden muss.

Für webbasierte Benutzeroberflächen (z. B. Dashboards) müssen spezielle Webanwendungsschecks verwendet werden.

Für den Zugang zu allen API-Endpunkten muss eine starke Authentifizierung verwendet werden, entsprechend den bestehenden Sicherheitsanforderungen für Webanwendungen und Webdienste (z. B. Dokument 3.02 - "Web Services").

Motivation: Cloud-API-Endpunkte haben eine große Angriffsfläche mit geringem eingebautem Schutz gegen unerwartete Eingaben und potenziell großen Auswirkungen, wenn sie verletzt werden.

Umsetzungsbeispiel: Die Sicherung von Cloud-APIs mit Proxys, die Begrenzung von API-Anfragen (einschließlich der Beschränkung möglicher API-Aufrufe auf das Notwendigste) und die Filterung von Eingaben sind entscheidend, um mögliche Angriffe über Cloud-APIs zu verhindern. Firewalls, WAFs und Proxys können mit Protokollierung und Überwachung integriert werden, so dass Angriffe frühzeitig erkannt und angemessen abgewehrt werden.

In einigen sehr speziellen Anwendungsfällen, in denen APIs nicht ordnungsgemäß gesichert werden können, während es nur eine sehr kleine Anzahl interner Mandanten gibt, kann eine weitere zusätzliche Maßnahme eingesetzt werden - das Verstecken von API-Endpunkten hinter Jump-Hosts, vorausgesetzt, die Anwendungsfälle der Mandanten las-

sen dies zu. Dann wäre der Zugriff auf Cloud-API-Endpunkte für den Tenant nur von Jump-Hosts aus möglich, wobei jeder Tenant einen Jump-Host erhalten könnte, der bei der Erstellung des Tenants erstellt wird.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigter Zugang zum System
- Unberechtigte Einsichtnahme oder Abgriff von Daten
- Unberechtigte Modifikation von Daten
- Unberechtigte Nutzung von Diensten oder Ressourcen
- Störung der Verfügbarkeit
- Abstreiten von durchgeführten Aktionen
- Angriffe motiviert und erleichtert durch Informationspreisgabe oder erkennbare Sicherheitsschwachstellen

ID: 3.86-12/2.0

Req 17 Mandanten müssen in der Lage sein, ihre Kubernetes-Cluster von anderen zu trennen.

Der Kubernetes-Netzwerkcontroller muss jedem Mandanten erlauben, private Namespaces zu erstellen, die nur für den Mandanten selbst sichtbar und erreichbar sind. Der Netzwerkcontroller muss außerdem jedem Mandanten die Möglichkeit bieten, bestimmte Teile seiner Namespaces von anderen Namespaces innerhalb des Clusters desselben Mandanten zu isolieren.

Motivation: Dies ermöglicht es Mandanten, n-Tier-Anwendungen zu erstellen.

Umsetzungsbeispiel: Kubernetes mit regulärer CNI-Vernetzung (Standardbeispiel mit Calico oder Flannel) erfüllt diese Anforderung.

ID: 3.34-17/5.0

Req 18 Alle Netzwerkverbindungen zum Speicher müssen verschlüsselt sein.

Alle Netzverbindungen (Zugriff) auf Daten, entweder von Hypervisoren oder Objektspeicher-Proxys für Kunden, müssen verschlüsselt werden.

Motivation: Dies verhindert, dass Angreifer im Netz an Daten im Klartext gelangen.

Umsetzungsbeispiel: Verwendung von TLS für alle Verbindungen zum Speichersystem.

ID: 3.34-18/5.0

Req 19 Die Daten des Mandanten müssen während der Übertragung oder im Ruhezustand verschlüsselt werden.

Je nach Datenklassifizierung und Sicherheitsstufe der Cloud-Infrastruktur (und des Netzes, das die verschiedenen Cloud-Umgebungen miteinander verbindet) ist es erforderlich, dass:

- der gesamte Netzwerkverkehr zwischen virtuellen Maschinen (oder Containern, die sich in verschiedenen virtuellen Maschinen befinden) verschlüsselt wird.
- Daten, die auf Block- oder Objektspeichern gespeichert sind, verschlüsselt werden.

Die in der Cloud bereitgestellten Anwendungen können auf einem einzigen Hypervisor, innerhalb derselben Cloud-Umgebung (Rechenzentrum) oder verteilt auf verschiedene Cloud-Umgebungen (Rechenzentren) laufen. Obwohl die Konnektivität für den Mandanten transparent sein kann, müssen sich die Mandanten darüber im Klaren sein, dass die Kommunikation zwischen ihren Workloads über Netzkabel auf öffentlichem Grund erfolgt. Daher muss der gesamte Datenverkehr des Mandanten, der eine einzelne Workload-Einheit verlässt (virtuelle Maschine oder ein Container, der mit anderen Containern kommuniziert, die sich an einem anderen Ort befinden können), verschlüsselt werden. Es empfiehlt sich, den gesamten Datenverkehr zu verschlüsseln, um sicherzustellen, dass die Anwendung im Fal-

le von Änderungen in den Cloud-Umgebungen oder der weiteren Bereitstellung der Anwendung an mehreren Standorten nicht neu archiviert werden muss. Die Verschlüsselung innerhalb des Rechenzentrums bietet auch zusätzlichen Schutz vor Hackern, die in die Cloud-Infrastruktur eingedrungen sind, oder vor unseriösen Cloud-Administratoren, so dass sie auch dann von Vorteil ist, wenn die Cloud-Umgebungen über verschlüsselte Verbindungen verbunden sind.

Motivation: Nicht alle Cloud-Umgebungen können an vollständig sicheren Standorten bereitgestellt werden (z. B. "Edge Clouds"), und trotz der von der Plattform angebotenen Festplattenverschlüsselung wären Angreifer, die sich Zugang zur Cloud-Infrastruktur verschafft haben, nicht in der Lage, die Daten leicht zu extrahieren, wenn die auf Volumes und in Objektspeicher-Buckets gespeicherten Daten verschlüsselt sind. Verschlüsselte Daten in Speicher-Buckets mit ordnungsgemäßer Verwendung verschiedener Schlüssel für verschiedene Buckets helfen auch, versehentliche Datenlecks zu verhindern, entweder für die Öffentlichkeit oder für andere Arbeitslasten (z. B. versehentliche Übertragung von der Produktion auf das Testsystem).

Umsetzungsbeispiel: Verwendung von sicheren Kommunikationsprotokollen (z. B. HTTPS, TLS und SSH) für die Kommunikation und Verwendung von Verschlüsselung für ruhende Daten - verschlüsselte Festplatten oder Objekte.

ID: 3.34-19/5.0

2.4. Authentication and authorization

Req 20	Role-based access control(RBAC) muss benutzt werden, um Zugriffe zu kontrollieren.
--------	--

RBAC, (standardmäßig aktiviert) ist eine Methode, um den Zugriff auf Clusterressourcen basierend auf den Rollen von Einzelpersonen innerhalb einer Organisation zu steuern. RBAC kann verwendet werden, um den Zugriff für Benutzerkonten und Dienstkonten einzuschränken. Um zu überprüfen, ob RBAC in einem Cluster aktiviert ist, führen Sie `kubectl api-versions` aus. Die API-Version `:rbac.authorization.k8s.io/v1` sollte aufgelistet werden, wenn RBAC aktiviert ist. Zwei Arten von Berechtigungen können festgelegt werden:

- Rollen – Berechtigungen für bestimmte Namespaces festlegen
- ClusterRoles – Berechtigungen für alle Clusterressourcen unabhängig vom Namespace festlegen Sowohl Rollen als auch ClusterRoles können nur zum Hinzufügen von Berechtigungen verwendet werden. RBAC sollte immer dem Konzept der geringsten Rechte folgen und ClusterRoles möglichst nicht benutzen.

Die RBAC-Rolle `cluster-admin` bietet umfassende Befugnisse über die Umgebung und sollte nur bei Bedarf verwendet werden.

Die Kubernetes-API speichert Secrets, die entweder Service-Account-Tokens für die Kubernetes-API oder Anmeldeinformationen sein können, die von Workloads im Cluster verwendet werden.

Der Zugriff auf diese Secrets sollte auf die kleinstmögliche Gruppe von Benutzern beschränkt werden, um das Risiko einer Privilegieneskalation zu minimieren.

Kubernetes Roles und ClusterRoles gewähren Zugriff auf Ressourcen basierend auf einer Menge von Objekten und Aktionen, die auf diesen Objekten ausgeführt werden können. Es ist möglich, bei beiden das Wildcard-Zeichen "*" zu setzen, das alle Elemente abdeckt. Die Verwendung von Wildcards muss vermieden werden. Diese Anforderung kann einer geteilten Verantwortung unterliegen. Entweder stellt ein Cluster-Anbieter vordefinierte RBAC-Rollen zur Verfügung, oder er gewährt z. B. die Cluster-Admin-Rolle. In diesem Fall ist der Nutzer des Clusters verantwortlich für die Einrichtung geeigneter RBAC-Kontrollen nach dem Prinzip der minimalen Rechtevergabe (Least Privilege).

Es gibt keine Ablehnungsregeln. Wenn ein Cluster für die Verwendung von RBAC konfiguriert ist und der anonyme Zugriff deaktiviert ist, verweigert der Kubernetes-API-Server Berechtigungen, die nicht ausdrücklich erlaubt sind.

Motivation: Zugriff auf Ressourcen einschränken.

ID: 3.34-20/5.0

Req 21	Es muss ein zentrales Authentifizierungssystem für den API-Server verwendet werden.
--------	---

Drittanbieter-Authentifizierungssysteme wie z.B. OpenID Connect Identity (OIDC) und OAuth 2 sollten benutzt werden. Ein zentrales Authentifizierungssystem hilft Administratoren dabei den API-Server nicht neu konfigurieren zu müssen wenn sie z.B. Benutzer hinzuzufügen oder entfernen möchten.

Motivation: Die Zentralisierung der Authentifizierung und Autorisierung (auch bekannt als Single Sign On) hilft beim

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigter Zugang zum System

ID: 3.34-21/5.0

2.5. Audit Logging / Threat Detection

Req 22	Sicherheitsrelevante Ereignisse müssen protokolliert werden und Audit logs müssen aktiviert sein.
--------	---

Ereignisse die protokolliert werden sollten :

- API request history
- Performance metrics
- Deployments
- Operating system calls
- Protocols,
- permission changes
- Network traffic
- Pod scaling
- Volume mount actions
- Image and container modification
- Privilege changes
- Scheduled job (cronjob) creations and modifications

Stellen Sie sicher, dass eine minimale Überwachungsrichtlinie erstellt wird.

Stellen Sie sicher, dass die Überwachungsrichtlinie die wichtigsten Sicherheitsaspekte abdeckt.

Motivation: Die Protokollierung von Orchestrierungsereignissen für Mandanten ist eine Voraussetzung für deren Sicherheit (z. B. wer hat wann eine bestimmte Cloud-Ressource erstellt, geändert oder gelöscht).

Umsetzungsbeispiel: Einige Beispiele umfassen die Protokollierung von:

- Anlegen, Löschen und Ändern von Tenants (Benutzer, Projekte, Rollen)
- Anlegen, Löschen und Ändern von Mandantennetzen
- Erstellung, Löschung und andere Aktionen der virtuellen Maschine
- Anlegen, Löschen, Ändern und Lesen von Objektspeicherbereichen und Objekten

ID: 3.34-22/5.0

Req 23	Protokolle aus allen Komponenten der Kubernetes-Infrastruktur müssen nahezu in Echtzeit an die zentrale Protokollierungslösung weitergeleitet werden. Darüber hinaus müssen die Protokolle einer geeigneten SIEM-Lösung zur Sammlung, Speicherung und Analyse sicherheitsrelevanter Ereignisse zur Verfügung gestellt werden. Das SIEM muss Sicherheitsereignisse korrelieren und Alarme für Angriffe generieren.
--------	---

Die Speicherung von Protokollen reicht nicht aus, sie müssen analysiert und richtig korreliert werden. Ein einzelnes System kann ein Ereignis erkennen und protokollieren, aber die Informationen reichen möglicherweise nicht aus, um einen laufenden Angriff zu erkennen. Dies gilt vor allem in Cloud-Umgebungen, in denen ein einzelner Mandant auf viele Server verteilt ist. Die Erkennung von Anomalien sollte zusätzlich zur Definition der Regeln für die Protokollfilterung, Überwachung und Alarmierung eingesetzt werden.

Daher muss zusätzlich zur zentralen Protokollierungslösung eine SIEM-Lösung (Security Information and Event-Management) eingesetzt werden, um Sicherheitsereignisse zu analysieren und zu korrelieren, Anomalien zu erkennen und im Falle von Problemen Alarm zu schlagen. Dies ermöglicht die erfolgreiche Implementierung eines Incident-Management-Prozesses, der eine 24/7-Nachverfolgung von Alarmen beinhalten muss.

Aufgrund der Angriffsfläche einer Cloud-Umgebung ist es erforderlich, dass die Protokollerfassung, -analyse und -korrelation sowie die Alarmierung nahezu in Echtzeit (mit möglichst geringer Verzögerung, z. B. 5 Minuten) erfolgen, um laufende Angriffe zu erkennen und zu entschärfen.

Unter der Annahme, dass die SIEM-Lösung vom zuständigen SOC (Security Operations Center) zur Verfügung gestellt wird, ist die Cloud-Plattform dafür verantwortlich, die Protokolle und ausreichende Informationen für die Analyse und Alarmierung bereitzustellen und weiterzuleiten sowie ausreichende Informationen und Hilfe bei der Einrichtung von Alarmen zu bieten.

Motivation: Die Speicherung von Protokollen auf den Servern, von denen sie stammen, reicht nicht aus, da der Angreifer die Protokolle leicht manipulieren und die Beweise für einen erfolgreichen Angriff löschen kann. Daher müssen alle Protokolle an das zentrale System weitergeleitet werden.

Umsetzungsbeispiel: SyslogNG, CheckMK, Nagios, Splunk etc. können als zentrales System eingesetzt werden.

ID: 3.34-23/5.0

Req 24	Jeder Kubernetes-Knoten muss über eine Endpoint Detection and Response- oder Container-Runtime-Lösung verfügen.
--------	---

Eine aktuelle Endpoint Detection and Response (EDR)-Lösung muss auf dem Betriebssystem eingesetzt werden. Eine EDR-Lösung sammelt sicherheitsrelevante Aktivitätsdaten von Prozessen und wertet diese zentral aus. Alarme können durch böses Verhalten von Prozessen oder durch eine spezifische Signatur (ähnlich wie bei klassischen Virenskannern) ausgelöst werden. Darüber hinaus ist es möglich, direkt auf verdächtiges Programmverhalten über eine zentrale Konsole zu reagieren. Beispielsweise kann das Betriebssystem bzw. der Client isoliert oder der böse Prozess beendet werden. Außerdem wird eine Übersicht über die Schwachstellen aller Betriebssysteme an eine zentrale Stelle weitergeleitet.

Alternativ kann Container Runtime Security implementiert werden, insbesondere in Umgebungen, die stark auf containerisierte Workloads angewiesen sind. Container Runtime Security konzentriert sich darauf, Container während ihrer Ausführung zu überwachen und zu schützen, wodurch die Erkennung und Reaktion auf böse Aktivitäten in Echtzeit ermöglicht wird. Dazu gehört die Identifizierung von nicht autorisierten Prozessen, anomalen Änderungen am Dateisystem und verdächtigem Netzwerkverhalten. Durch die Durchsetzung von Sicherheitsrichtlinien und die Überwachung des Laufzeitverhaltens bietet Container Runtime Security eine robuste Alternative oder Ergänzung zu EDR, insbesondere in modernen Cloud-nativen und Kubernetes-basierten Infrastrukturen.

Motivation: Normale Virenskanner basieren ausschließlich auf Signaturen, während eine EDR-Lösung auch nach Anomalien im Prozessverhalten sucht. Signaturen haben den Nachteil, dass für neue Virusvarianten erst neue Signaturen erstellt werden müssen, bevor sie erkannt werden können.

Umsetzungsbeispiel: Schauen Sie sich die PSA-Wiki-Seite - Sicherheitsanforderungen: Endpoint Detection and Response (EDR) an, um weitere Anleitungen zu erhalten.
<https://yam-united.telekom.com/pages/psa/apps/wiki/wiki-sicherheitsanforderungen/list/view/7214872e-276f-4cb0-acd7-4c85860d0ee8>

Für Container Runtime Security könnten Sie etwas in Betracht ziehen wie:
<https://github.com/falcosecurity/falco>
<https://github.com/cilium/tetragon/>

ID: 3.34-24/5.0

2.6. Pod Sicherheit

Pods sind die kleinste einsetzbare Einheit im Kubernetes Umfeld. Ein Pod besteht aus einem oder mehreren Containern.

Req 25 Pod Security Admissions oder 3rd party admission plugins müssen verwendet werden, um Container-Berechtigungen einzuschränken und Plattform Regeln bereitzustellen.

Eine restriktive Policy muss aktiviert sein. Ressourcen und Volumes, auf die ein Pod zugreifen kann, müssen begrenzt sein.

Es gibt verschiedenen Möglichkeiten Policies zu implementieren:

- Pod Security Admissions <https://kubernetes.io/docs/concepts/security/pod-security-admission/>
- Open Policy Agent / Gatekeeper <https://www.openpolicyagent.org/>
- Kyverno <https://kyverno.io/>

Policies **müssen** limitieren:

Verbieten Sie die Zulassung von privilegierten Containern.

Verbieten Sie die Zulassung von Containern, die den Host-Prozess-ID-Namespace teilen.

Verbieten Sie die Zulassung von Containern, die den Host-IPC-Namespace teilen.

Verbieten Sie die Zulassung von Containern, die den Host-Netzwerk-Namespace teilen möchten.

Verbieten Sie die Zulassung von Containern mit erlaubter Privilegieneskulation (allowPrivilegeEscalation).

Verbieten Sie die Zulassung von Root-Containern.

Verbieten Sie die Zulassung von Containern mit der NET_RAW-Funktion.

Verbieten Sie die Zulassung von Containern mit hinzugefügten Funktionen (Capabilities).

Verbieten Sie die Zulassung von Containern mit zugewiesenen Funktionen (Capabilities).

Verbieten Sie die Zulassung von Windows HostProcess Containern.

Verbieten Sie die Zulassung von HostPath-Volumes.

Verbieten Sie die Zulassung von Containern, die HostPorts verwenden.

Stellen Sie sicher, dass für Container ein schreibgeschütztes Dateisystem (ReadOnly Filesystem) erzwungen wird.

Motivation: Die meisten Pods benötigen keinen privilegierten Zugriff oder gar Host-Zugriff.

Ein solcher Zugriff muss explizit auf die Whitelist gesetzt werden. Standardmäßig sollte niemand in der Lage sein, Berechtigungen anzufordern, die über der default Policy liegen.

Umsetzungsbeispiel: Geteilte Verantwortung: Die Plattform (Orchestrator) muss diese Funktion bereitstellen. Eine Applikation muss diese Funktion nutzen.

Auf den meisten Plattformen sind Richtlinien definiert, die Anwendungen befolgen müssen.

Container/Pods müssen einen Security Context definieren <https://kubernetes.io/docs/tasks/configure-pod-container/security-context/>

<https://kubernetes.io/docs/concepts/security/pod-security-standards/>

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigter Zugang zum System
- Unberechtigte Einsichtnahme oder Abgriff von Daten
- Unberechtigte Modifikation von Daten
- Unberechtigte Nutzung von Diensten oder Ressourcen

ID: 3.34-25/5.0

Req 26 Resource policies müssen benutzt werden.

Resource policies LimitRanges, ResourceQuotas, und Process ID Limits, schränken die Benutzung von Ressourcen innerhalb von Namespaces, Nodes und Containern ein.

Motivation: Schutz des Systems vor Überlast

Umsetzungsbeispiel: Legen Sie die minimale und maximale Nutzung von Rechenressourcen pro Pod oder Container in einem Namespace fest.

Legen Sie die minimale und maximale Speicheranforderung pro PersistentVolumeClaim in einem Namespace fest.

Legen Sie ein Verhältnis zwischen Request und Limit für eine Ressource in einem Namespace fest.
Legen Sie für Requests/Limits für Rechenressourcen in einem Namespace fest und fügen Sie diese zur Laufzeit automatisch in Container ein.

ID: 3.34-26/5.0

Req 27	Nicht vertrauenswürdige Workloads müssen isoliert werden, wenn sie ein Risiko für die Kubernetes-Plattform und ihre Management-Komponenten darstellen.
--------	--

Workload Isolation ist die Praxis, Sicherheitsgrenzen zwischen Containern und ihrer Host-Umgebung zu etablieren, um das Risiko von Container-Escape-Schwachstellen zu mindern und den potenziellen Schaden bei kompromittierten Containern zu begrenzen.

Verschiedene Workloads benötigen möglicherweise unterschiedliche Isolationsstufen basierend auf ihrem Risikoprofil:

Basis-Isolation (Standard-Container mit Security Best Practices)

Für vertrauenswürdige Workloads mit minimalem Risiko.

Basiert auf Linux namespaces und cgroups.

Ergänzt durch Security Context-Einschränkungen, seccomp, AppArmor, etc.

Erweiterte Isolation (User namespaces)

<https://kubernetes.io/docs/concepts/workloads/pods/user-namespaces/>

Für Workloads, bei denen Prozesse innerhalb des Containers als root laufen sollen, aber mit eingeschränkten Rechten auf dem Host.

Mindert bestimmte Kategorien von Container-Escape-Schwachstellen.

Schützt vor UID/GID-Konflikten zwischen Containern.

Starke Isolation, Sandboxing (gVisor, Kata Containers)

Für nicht vertrauenswürdige oder hochriskante Workloads.

Anwendungen, die nicht in eine Cloud-Native-Umgebung passen.

Security Policies, die zu stark von Best Practices abweichen und ein Risiko für die Plattform darstellen.

Die geeignete Isolationsstrategie hängt von Ihrem spezifischen Bedrohungsmodell, Leistungsanforderungen und dem Vertrauensgrad in die Workloads ab, die auf Ihrem Cluster ausgeführt werden. Mit steigendem Risiko des Workloads sollte die Isolation von der Basis-Isolation über die erweiterte Isolation (User namespaces) bis hin zur starken Isolation (Sandboxing-Runtimes) eskaliert werden, je nach Bedarf.

ID: 3.34-27/5.0

2.7. Deployment/Application Level Security

This section describes the security measures Applications have to follow when running on top of an Orchestrator.

Req 28	Die Verwendung des Standard-Namespace von Kubernetes ist verboten.
--------	--

Ein Kubernetes-Namespace ermöglicht es Ihnen, erstellte Ressourcen in logisch benannte Gruppen zu partitionieren. Ressourcen, die in einem Namespace erstellt wurden, können in anderen Namespaces ausgeblendet werden. Standardmäßig wird jede Ressource, die von einem Benutzer in einem Kubernetes-Cluster erstellt wird, in einem Standardnamespace mit dem Namen default ausgeführt. Sie können zusätzliche Namespaces erstellen und Ressourcen und Benutzer an diese anfügen.

Verwenden Sie Namespaces, um Ihre Kubernetes-Objekte zu isolieren.

ID: 3.34-28/5.0

Req 29	Vertrauliche Informationen dürfen nicht in YAML-Ressourcen gespeichert werden.
--------	--

Die Verwendung eines KMS ist die bevorzugte Methode.

Wenn ein KMS nicht verfügbar ist oder es nicht möglich ist, ein KMS zu verwenden, müssen die Secrets auf sichere

Weise mit Tools wie SOPS (<https://github.com/getsops/sops>) verwaltet werden.

Sensible Informationen dürfen nicht in YAML-Ressourcen vom Typ Pod (Deployments, Pods, Sets, etc.) gespeichert werden, und sensible ConfigMaps und Secrets müssen verschlüsselt werden.

Folgendes muss berücksichtigt werden:

Regelmäßiger Rotationsplan: Etablieren und befolgen Sie einen strikten Plan zur Schlüsselrotation (mindestens einmal jährlich).

Onboarding/Offboarding: Implementieren Sie Prozesse für den Umgang mit Schlüsseln (Rotation/Löschung) während Personalwechseln.

Re-Verschlüsselungsprozess: Definieren Sie klare Verfahren für die Neuverschlüsselung aller Secrets bei Schlüsseländerungen.

Notfallrotation: Dokumentieren Sie Verfahren für eine Notfallrotation von Schlüsseln im Falle einer Kompromittierung, einschließlich vermuteter Kompromittierung.

Kompromittierung oder Verdacht auf Kompromittierung: Rotieren Sie alle Schlüssel/Secrets sofort, wenn eine Kompromittierung vermutet wird.

Motivation: Verhindern Sie, dass Secrets in yaml-Dateien sichtbar werden.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigter Zugang zum System
- Unberechtigte Einsichtnahme oder Abgriff von Daten
- Unberechtigte Modifikation von Daten
- Unberechtigte Nutzung von Diensten oder Ressourcen

ID: 3.34-29/5.0

Req 30	Network Policies müssen verwendet werden, um den Datenverkehr zwischen Pods einzuschränken und eine ordnungsgemäße Netzwerksegmentierung sicherzustellen.
--------	---

Definieren Sie Eingangs- und Ausgangsregeln, um den Datenverkehrsfluss zwischen Pods, Namespaces und externen Systemen zu steuern.

Motivation: Die Verwendung von Network Policies gewährleistet eine Netzwerksegmentierung und beschränkt den Datenverkehr zwischen Pods, wodurch das Risiko unbefugten Zugriffs und seitlicher Bewegungen (lateral movement) im Cluster verringert wird.

Umsetzungsbeispiel: apiVersion: networking.k8s.io/v1

kind: NetworkPolicy

metadata:

name: allow-app-to-db

spec:

podSelector:

matchLabels:

app: my-app

ingress:

- from:

- podSelector:

matchLabels:

app: my-database

ports:

- protocol: TCP

port: 3306

ID: 3.34-30/5.0

Req 31	RBAC muss verwendet werden, um den Zugriff auf Kubernetes-Ressourcen basierend auf dem Prinzip der geringsten Rechte zu steuern.
--------	--

Definieren Sie Rollen und RoleBindings:

Erteilen Sie Benutzern, Gruppen und Dienstkonten nur die erforderlichen Berechtigungen.

Stellen Sie sicher, dass Dienstkonten, die von Anwendungen verwendet werden, über eingeschränkte Berechtigungen verfügen, insbesondere für vertrauliche Ressourcen wie Geheimnisse und Konfigurationszuordnungen.

Motivation: Die Verwendung von RBAC zur Zugriffskontrolle auf Kubernetes-Ressourcen setzt das Prinzip der minimalen Rechte um und verringert das Risiko unbefugter Aktionen und potenzieller Sicherheitsverletzungen.

ID: 3.34-31/5.0

Req 32	Deployments/Anwendungen müssen ihren Sicherheitskontext angeben.
--------	--

Im Security Context muss folgendes definiert werden:

<https://kubernetes.io/docs/tasks/configure-pod-container/security-context/>

Die folgenden Einschränkungen müssen im Security Context definiert werden, um die Sicherheit von Kubernetes-Workloads zu gewährleisten:

Verbot von privilegierten Containern

Im Security Context muss privileged: false gesetzt werden.

Dies verhindert, dass Container mit Root-Rechten auf dem Host ausgeführt werden.

Verbot der Nutzung der Host-PID-Namespaces

Im Pod-Spec muss hostPID: false gesetzt werden.

Verhindert, dass Container Prozesse auf dem Host-System sehen oder manipulieren können.

Verbot der Nutzung der Host-IPC-Namespaces

Im Pod-Spec muss hostIPC: false gesetzt werden.

Verhindert die Nutzung von Host-Inter-Process-Communication-Mechanismen.

Verbot der Nutzung des Host-Netzwerk-Namespaces

Im Pod-Spec muss hostNetwork: false gesetzt werden.

Verhindert direkten Zugriff auf das Netzwerk-Interface des Hosts.

Verbot von Privilege Escalation

Im Security Context muss allowPrivilegeEscalation: false gesetzt werden.

Verhindert, dass Container-Prozesse ihre Rechte erweitern können.

Verbot von Root-Containern

Im Security Context muss runAsNonRoot: true gesetzt werden.

Erzwingt die Ausführung von Containern mit Nicht-Root-Benutzern.

Verbot der NET_RAW Capability

Im Security Context muss die NET_RAW Capability explizit nicht in der Liste der capabilities.add enthalten sein.

Diese Capability ermöglicht das Erstellen von Raw-Sockets und sollte aus Sicherheitsgründen entfernt werden.

Verbot von zusätzlichen Capabilities

Im Security Context dürfen keine zusätzlichen Capabilities durch capabilities.add hinzugefügt werden.

Limitiert die Möglichkeiten für Container-Escape-Angriffe.

Entfernen aller Capabilities

Im Security Context muss capabilities.drop: ["ALL"] gesetzt werden.

Entfernt standardmäßig alle Linux-Capabilities und erhöht die Sicherheit.

Verbot von Windows HostProcess Containern

Für Windows-Container muss windowsOptions.hostProcess: false gesetzt werden.

Verhindert, dass Container direkt mit dem Windows-Host interagieren können.

Verbot von HostPath Volumes

Im Pod-Spec dürfen keine hostPath-Volumes definiert werden.

Verhindert direkten Zugriff auf das Host-Dateisystem.

Verbot der Nutzung von Host-Ports

Im Container-Spec dürfen keine hostPort-Definitionen vorhanden sein.
Verhindert, dass Container direkt an Host-Ports binden können.

Erzwingen von ReadOnly-Dateisystemen

Im Security Context muss readOnlyRootFilesystem: true gesetzt werden.
Verhindert Schreibzugriffe auf das Container-Dateisystem.

Motivation: Diese Anforderungen implementieren mehrere Schutzschichten gegen Container-Escape-Schwachstellen, die es böswilligen Containern andernfalls ermöglichen könnten, auszubrechen und auf das Host-System zuzugreifen.

Prinzip der geringsten Privilegien: Durch die Einschränkung von Capabilities, Zugriffsrechten und Namespaces arbeiten Container nur mit den minimal notwendigen Privilegien, was die Angriffsfläche erheblich reduziert.

Verhinderung von Privilege Escalation: Diese Kontrollen verhindern, dass Container während der Laufzeit zusätzliche Privilegien erlangen, was eine häufige Technik bei Container-Escape-Angriffen ist.

Schutz des Host-Systems: Durch die Isolierung von Containern von Host-Ressourcen (Netzwerk, IPC, PID-Namespaces) ist das Host-System vor direkter Manipulation durch containerisierte Anwendungen geschützt.

Umsetzungsbeispiel: apiVersion: apps/v1

kind: Deployment

metadata:

name: compliant-nginx

labels:

app: compliant-nginx

spec:

replicas: 3 # Multiple replicas for high availability

selector:

matchLabels:

app: compliant-nginx

template:

metadata:

labels:

app: compliant-nginx

annotations:

SECURITY: Applies seccomp profile to restrict syscalls, reducing attack surface

seccomp.security.alpha.kubernetes.io/pod: "runtime/default"

SECURITY: Applies AppArmor profile to restrict file access and capabilities

container.apparmor.security.beta.kubernetes.io/compliant-nginx: "runtime/default"

spec:

securityContext:

SECURITY: Prevents container from running as root user

runAsNonRoot: true

SECURITY: Specifies non-root user ID to run container processes

runAsUser: 101

SECURITY: Sets group ID for container processes

runAsGroup: 101

SECURITY: Sets group ID for mounted volumes

fsGroup: 101

SECURITY: Applies seccomp profile at pod level (newer API)

seccompProfile:

type: RuntimeDefault

containers:

- name: compliant-nginx

SECURITY: Uses specific version instead of "latest" to ensure consistency and proper security testing

image: nginx:1.25.3-alpine

SECURITY: Avoids automatically pulling potentially compromised new images

imagePullPolicy: IfNotPresent

securityContext:

```

# SECURITY: Prevents processes from gaining additional privileges
allowPrivilegeEscalation: false
# SECURITY: Prevents modification of container filesystem during runtime
readOnlyRootFilesystem: true
capabilities:
# SECURITY: Drops all Linux capabilities by default
drop: ["ALL"]
# SECURITY: Explicitly disables privileged mode which would give host-level access
privileged: false
resources:
# SECURITY: Sets resource limits to prevent resource exhaustion attacks
limits:
cpu: "500m"
memory: "256Mi"
requests:
cpu: "100m"
memory: "128Mi"
ports:
# SECURITY: Uses non-privileged port (>1024) to minimize required capabilities
- containerPort: 8080
name: http
volumeMounts:
# SECURITY: Mounts required writable directories as isolated volumes since root filesystem is read-only
- name: tmp-volume
mountPath: /tmp
- name: nginx-cache
mountPath: /var/cache/nginx
- name: nginx-run
mountPath: /var/run
# SECURITY: Mounts configuration as read-only to prevent runtime modification
- name: nginx-conf
mountPath: /etc/nginx/conf.d
readOnly: true
# SECURITY: Implements health probes to ensure container is functioning properly
livenessProbe:
httpGet:
path: /
port: http
initialDelaySeconds: 10
periodSeconds: 20
readinessProbe:
httpGet:
path: /
port: http
initialDelaySeconds: 5
periodSeconds: 10
volumes:
# SECURITY: Uses ephemeral volumes instead of persistent storage for writable directories
- name: tmp-volume
emptyDir: {}
- name: nginx-cache
emptyDir: {}
- name: nginx-run
emptyDir: {}
# SECURITY: Uses ConfigMap for configuration instead of hardcoded values
- name: nginx-conf
configMap:
name: compliant-nginx-config
# SECURITY: Prevents pod from accessing host network interfaces
hostNetwork: false
# SECURITY: Prevents pod from viewing or manipulating host processes
hostPID: false

```

SECURITY: Prevents pod from accessing host IPC mechanisms
hostIPC: false
SECURITY: Prevents automatic mounting of service account token to limit access to Kubernetes API
automountServiceAccountToken: false

ID: 3.34-32/5.0

Req 33	Die Anpassung der Cloud-Infrastruktur darf von den Mandanten nicht verzögert oder blockiert werden (weder technisch noch administrativ).
--------	--

Es ist zwingend erforderlich, dass die SLA die Tatsache widerspiegelt, dass kein Mandant das Patchen der zugrunde liegenden Infrastruktur blockieren oder verhindern kann.

Kunden, die die Cloud unter der Annahme nutzen wollen, dass Patching oder andere Wartungsarbeiten durch sie verzögert oder blockiert werden kann, dürfen unter keinen Umständen auf die Cloud migrieren.

Motivation: Patching schützt alle anderen Workloads und die Cloud-Infrastruktur, und es wäre extrem unvernünftig und gefährlich, die Sicherheit der gesamten Cloud und aller anderen Systeme nur wegen eines Mandanten zu gefährden.

Umsetzungsbeispiel: Bei bestimmten Technologien und Arbeitslasten kann dies durch eine Live-Migration erreicht werden, und bei solchen, für die es keine technischen Möglichkeiten gibt, kann dies durch eine Klausel in den SLA-Bedingungen abgedeckt werden.

ID: 3.34-33/5.0

3. Private Clouds

3.1. Cloud-Infrastruktur-Netzwerk und Cloud-Management

Das Cloud-Infrastrukturnetz (auch Underlay-Netz genannt) der Cloud ist das interne Cloud-Netz, das für die IP-Konnektivität zwischen allen Hardware-Komponenten und der Cloud-Software erforderlich ist, einschließlich:

- Rechenknoten
- Storage-Knoten
- Netzwerkknoten
- Serververwaltung (BMC - Basedboard Management Controller, z. B. IPMI oder Redfish)
- Netzwerkgeräte (Switches, Router, Firewalls, ...)
- Cloud-Management-Funktionen

Dies gilt nicht für Mandanten-Netzwerke.

3.1.1. Rund um das Cloud Management

Eine Cloud-Umgebung erfordert eine Vielzahl anderer Systeme, die für den sicheren Betrieb der Cloud-Umgebung vorhanden sein müssen. Die meisten dieser Systeme (z. B. Protokollierung, Überwachung, Authentifizierung usw.) sind gut bekannt und werden in anderen Sicherheitsdokumenten ausführlich behandelt.

Bei der Vorbereitung, der Konfiguration, der Installation und dem Betrieb von Cloud-Infrastrukturen gilt es, die grundlegenden Prinzipien zu wiederholen:

- die Angriffsfläche von Systemen und Diensten muss begrenzt werden
- die Systeme müssen dem Prinzip der minimalen Installation folgen
- alle sensiblen Daten müssen während der Übertragung und Speicherung (und wenn möglich auch während der Nutzung) verschlüsselt werden
- die Berechtigungen der Benutzer (Menschen und M2M) müssen dem Least-Privilege Prinzip folgen
- die Kommunikationspartner müssen angemessen authentifiziert und autorisiert sein
- die Systeme müssen entsprechend ihrer Vertraulichkeitsstufe getrennt werden
- die eingehende, ausgehende und interne Kommunikation muss kontrolliert und gefiltert werden
- Infrastruktur, Systeme und Dienste müssen überwacht und regelmäßig getestet werden
- die Systeme müssen regelmäßig aktualisiert und gepatcht werden
- ein Verfahren für das Management von Zwischenfällen muss vorhanden sein

Req 34	Die Überprüfung der Einhaltung der Sicherheitsvorschriften muss automatisch erfolgen.
--------	---

Aufgrund der Vielfalt der Systeme, die in der Cloud gehostet werden, und der Komplexität der Cloud-Plattform ist es erforderlich, auf allen Servern und Geräten die Einhaltung der einschlägigen Sicherheitsanforderungen zu überprüfen.

Die Implementierung muss als Teil der CI/CD-Kette vor der Bereitstellung (wenn CI/CD und Automatisierung verwendet werden) und während der Lebensdauer regelmäßig erfolgen. Wie bei der Härtung muss auch hier die Automatisierung so weit wie möglich genutzt werden, um das Risiko zu vermeiden, dass Konformitätsprüfungen und/oder Systeme übersehen werden.

Motivation: Regelmäßige Überprüfungen der Einhaltung der Vorschriften validieren den aktuellen Sicherheitsstatus von Systemen und können Schwachstellen in Sicherheitskonfigurationen oder ungepatchte Systeme mit bekannten Sicherheitslücken aufdecken.

Umsetzungsbeispiel: Einige Beispiele für Tools zur Überprüfung der Einhaltung von Vorschriften sind:

- CIS-Benchmarks
- Ansible

- InSpec
- Orpheus (Telekom intern)
- TASTE-OS (Telekom-intern, eine Liste aller TOS-Instanzen finden Sie unter <https://yam.telekom.de/docs/DOC-685659>)

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigter Zugang zum System
- Unberechtigte Einsichtnahme oder Abgriff von Daten
- Unberechtigte Modifikation von Daten
- Unberechtigte Nutzung von Diensten oder Ressourcen
- Störung der Verfügbarkeit
- Abstreiten von durchgeführten Aktionen
- Unbemerkt durchführbare Angriffe
- Angriffe motiviert und erleichtert durch Informationspreisgabe oder erkennbare Sicherheitsschwachstellen

ID: 3.86-28/2.0

3.2. Cloud-Nutzung

Clouds werden in der Regel eingesetzt, um die Automatisierung zu ermöglichen und die Kosten zu minimieren. Häufig wird dieselbe Cloud-Umgebung auch für die Bereitstellung von Entwicklungs-, Test-, Referenz- und Produktionssystemen/-anwendungen verwendet. Die Vermischung von Entwicklungs-, Test-, Referenz- und Produktionssystemen birgt ein zusätzliches Risiko. Bei ordnungsgemäßer Anwendung aller Sicherheitsmaßnahmen in diesem Dokument und der besten Sicherheitspraktiken im Allgemeinen ist es möglich, das Risiko auf ein akzeptables Niveau für den Betrieb von Entwicklungs-, Test-, Referenz- und Produktionssystemen in derselben Cloud zu senken.

In diesem Teil des Dokuments werden zusätzlich verbindliche Sicherheitsanforderungen beschrieben, die eine Vermischung von Entwicklungs-, Test-, Referenz- und Produktions-Cloud-Infrastrukturen sowie eine Vermischung von Entwicklungs-, Test-, Referenz- und Produktionssystemen in derselben Cloud-Infrastruktur ermöglichen.

3.2.1. Zweckgebundene Trennung der Cloud-Infrastruktur

Req 35	Die Entwicklungs-, Test- und Referenz-Cloud-Infrastrukturen selbst müssen auf einer von der Produktions-Cloud-Infrastruktur vollständig getrennten Hardware betrieben werden, einschließlich des Cloud-Management- und Orchestrierungssystems.
--------	--

Rechenknoten, Speicherknoten, Netzwerkgeräte und andere unterstützende Systeme, die zur Entwicklungs-, Test- oder Referenz-Cloud-Infrastruktur gehören, müssen auf einer von der Produktions-Cloud-Umgebung getrennten Hardware betrieben werden. Dies gilt auch für einen separaten DCR (Data Center Router).

Selbst wenn die Sicherheitsanforderungen für Test- und Referenzsysteme mit den Sicherheitsanforderungen für Produktionssysteme übereinstimmen, müssen auch betriebliche Aspekte berücksichtigt werden. Das Testen einer neuen Änderung oder Funktion auf gemeinsam genutzter Hardware zwischen Test- und Produktions-Cloud-Umgebungen könnte die gesamte Produktions-Cloud-Umgebung zum Absturz bringen.

Motivation: Entwicklungs-, Test- und Referenzsysteme haben nicht immer dieselben Sicherheitsanforderungen wie Produktionssysteme, und eine Vermischung dieser Systeme würde unnötige Risiken für die Cloud-Produktionsumgebung mit sich bringen.

Umsetzungsbeispiel: Separate (getrennte) Hardware (oder Software oder extern genutzte Dienste) für jede der Cloud-Infrastrukturplattformen für Entwicklung, Tests und Produktion.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigter Zugang zum System
- Unberechtigte Einsichtnahme oder Abgriff von Daten
- Unberechtigte Modifikation von Daten
- Unberechtigte Nutzung von Diensten oder Ressourcen
- Störung der Verfügbarkeit

- Abstreiten von durchgeführten Aktionen
- Angriffe motiviert und erleichtert durch Informationspreisgabe oder erkennbare Sicherheitsschwachstellen

ID: 3.86-63/2.0