

Sicherheitsanforderung

Container

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 3.64	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 Containern verwendet werden, zusätzlich zu den CIS-Benchmarks, die auch implementierungsorientierte 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.	General	5
2.2.	Container Linux Host Konfiguration	7
2.3.	Container Daemon Konfiguration	8
2.4.	Container-Daemon-Konfigurationsdateien	9
2.5.	Container-Images und Build-Datei-Konfiguration	10
2.6.	Container Runtime Konfiguration	17
2.7.	Container Security Operationen	22
2.8.	Registry	22
3.	Technischer Basisschutz von IT-/NT-Systemen	24
3.1.	Protokollierung	24

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

Dieses Anforderungsdokument beschreibt die wesentlichen Sicherheitsmaßnahmen und best practices für das Management und die Bereitstellung von Containern.

Es zielt darauf ab, sicherzustellen, dass Container sicher verwendet werden und behandelt Aspekte wie die Unveränderlichkeit von Containern, Benutzerprivilegien, Betriebssystemkonfigurationen und die Trennung von Produktions- und Entwicklungsumgebungen.

Das Modell der geteilten Verantwortung für die Containersicherheit umfasst sowohl den Anbieter der Container-Laufzeitumgebung als auch den Benutzer (z. B. DevOps-Teams). Der Anbieter der Container-Laufzeitumgebung ist für die Sicherstellung sicherer Standardkonfigurationen, die Implementierung von Sicherheitsmechanismen und die Bereitstellung regelmäßiger Updates verantwortlich.

Der Benutzer ist für den Aufbau und die Wartung sicherer Container-Images, die Konfiguration von Sicherheitseinstellungen für Container, die Verwaltung von Netzwerk- und Ressourcenkonfigurationen sowie die Einhaltung von Sicherheitsrichtlinien verantwortlich.

2.1. General

Req 1 Container müssen als unveränderlich behandelt werden.

Container sollen unveränderlich sein, daher dürfen sie zur Laufzeit nicht verändert werden. Anstatt Container im Livebetrieb zu patchen, muss das Image gepatched und redeployed werden. Container Images müssen über eine vorhandene CI/CD Pipeline geändert werden. Diese Pipeline muss den Sicherheitsanforderungen für CI/CD folgen

Motivation: Container sollen unveränderlich sein, daher dürfen sie zur Laufzeit nicht verändert werden. Anstatt Container im Livebetrieb zu patchen, muss das Image gepatched und redeployed werden. Container Images müssen über eine vorhandene CI/CD Pipeline geändert werden. Diese Pipeline muss den Sicherheitsanforderungen für CI/CD folgen

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigter Zugang zum System
- Unberechtigte Modifikation von Daten

ID: 3.64-1/5.0

Req 2 Produktionssysteme müssen von Test- und Entwicklungssystemen vollständig getrennt sein.

Neben Servern sind auch Infrastrukturelemente wie Netz- und Speichersysteme zu trennen. Eine in Teilen lediglich logische Trennung ist nur dann zulässig, wenn gewährleistet ist, dass diese logische Trennung nicht umgangen werden kann, und dass eine sicherheitsrelevante Beeinträchtigung der Produktion durch Aktivitäten auf den Test- und Entwicklungssystemen ausgeschlossen ist. Um dies zu erreichen wird eine physische Trennung empfohlen. Für Referenzanlagen muss entschieden werden, ob diese einen eher produktiven Charakter haben oder eher einer Testumgebung zuzuordnen sind.

Eine direkte Kommunikation ist nur dann zulässig, wenn sie absolut unumgänglich ist und über Firewalls geführt wird. Ansonsten sind Übergänge zwischen diesen Netzbereichen über Sicherheitssysteme wie Jumphosts herzustellen.

Testaktivitäten mit Sicherheitsbezug wie Penetrationstests und die Durchführung von Netzwerk-Scans müssen jederzeit ohne schädliche Auswirkungen auf die Produktion durchgeführt werden können.

Es wird empfohlen, für diese unterschiedlichen Netzbereiche eindeutig unterscheidbare IP-Adressbereiche zu verwenden.

Motivation: Bei Test- und Entwicklungssystemen kann nicht von einem hinreichend sicheren Systemzustand ausgegangen werden, insbesondere da diese naturgemäß permanenten Veränderungen ausgesetzt sind. Bei einer Verknüpfung der Systemarten untereinander oder der Nutzung gleicher Plattformen bestehen die Risiken, dass Unberechtigte aus der Test-/Entwicklungsumgebung heraus auf Produktionssysteme und Wirkdaten zugreifen können oder die Stabilität der Produktionssysteme und damit die Verfügbarkeit der zugehörigen Dienste gefährdet wird. Eine von der Produktion vollkommen unabhängige Test- und Entwicklungs-Infrastruktur ermöglicht auch während einer

„Frozen Zone“ Änderungen für Test- und Entwicklungssysteme durchzuführen. So können Sicherheitsupdates schneller getestet und für die Produktion bereitgestellt werden.

Umsetzungsbeispiel: Für Abnahmetests, in denen auch Sicherheits-Scans und Penetrationstests durchgeführt werden, wird eine von der Produktion unabhängige Umgebung verwendet, um jedwede Beeinträchtigung zu vermeiden. Ein weiteres Beispiel ist eine Umgebung für Performancetests, die physisch getrennt von der Produktion aufgebaut ist. Bei Nutzung gemeinsamer Komponenten, z. B. gleiches Speichersystem oder gemeinsames Netz, könnten ansonsten Beeinträchtigungen produktiver Systeme nicht ausgeschlossen werden.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigter Zugang zum System
- Unberechtigte Einsichtnahme oder Abgriff von Daten
- Störung der Verfügbarkeit
- Angriffe motiviert und erleichtert durch Informationspreisgabe oder erkennbare Sicherheitsschwachstellen

Für diese Anforderung sind folgende Gewährleistungsziele relevant:

- Verfügbarkeit

ID: 3.14-2/3.3

Req 3	Bekannte Schwachstellen in der Software oder Hardware des Systems müssen behoben oder gegen Missbrauch abgesichert werden.
-------	--

Bekannte Schwachstellen in Software- und Hardware Komponenten müssen mittels Installation von verfügbaren Systemaktualisierungen des Herstellers (z. B. Patches, Updates/Upgrades) behoben werden. Alternativ ist der Einsatz von Workarounds (Akut-Lösungen, welche die Schwachstelle nicht beheben, aber die Ausnutzung wirksam verhindern) zulässig. Workarounds sollten grundlegend nur temporär Verwendung finden und sobald möglich durch eine reguläre Systemaktualisierung ersetzt werden, um die Schwachstellen vollständig zu schließen.

Komponenten, die bekannte, nicht behebbare Schwachstellen enthalten, dürfen in einem System nicht eingesetzt werden.

Die Behandlung von neu bekannt werdenden Schwachstellen muss zudem für die gesamte Einsatzphase des Systems fortlaufend sichergestellt und in den kontinuierlichen Betriebsprozessen eines Security Patch Managements umgesetzt werden.

Motivation: Der Einsatz von Komponenten ohne eine Behebung enthaltener Schwachstellen erhöht deutlich das Risiko einer erfolgreichen Kompromittierung. Der Angreifer wird zusätzlich dadurch begünstigt, dass i. d. R. nicht nur detaillierte Informationen zu bereits bekannt gewordenen Schwachstellen offen verfügbar sind, sondern oft auch bereits angepasste Angriffswerkzeuge, die eine aktive Ausnutzung erleichtern.

Umsetzungsbeispiel: Im Anschluss an die initiale Installation eines Betriebssystems von einem offiziellen Installationsmedium werden alle aktuell verfügbaren Patches und Sicherheits-Updates eingespielt.

Zusätzliche Informationen:

Primäre Quellen für bereits bekannt gewordene Schwachstellen in einer Software/Hardware sind Auflistungen in den Versionshinweisen sowie die Sicherheitshinweise aus den offiziellen Meldekanälen des Herstellers oder unabhängiger CERTs. Insbesondere die Meldekanäle werden sinnvollerweise fest in kontinuierliche Prozesse des Security Patch Managements für ein System eingebunden, damit neu bekanntwerdende Schwachstellen zeitnah registriert und in operative Behebungsmaßnahmen geführt werden können.

Als ergänzende Maßnahme zur Aufdeckung von potenziell noch enthaltenen, grundsätzlich bereits bekannt gewordenen Arten von Schwachstellen können gezielte Schwachstellenuntersuchungen des Systems durchgeführt werden. Zu diesem Zweck eignen sich besonders spezialisierte Werkzeuge wie automatisierte Schwachstellen-Scanner. Beispiele hierfür sind: Tenable Nessus, Qualys Scanner Appliance.

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: 3.01-10/9.0

Req 4	Software und Hardware des Systems müssen von einem Sicherheitsschwachstellen-Support seitens des Herstellers abgedeckt werden.
-------	--

In einem System dürfen nur Software- und Hardware-Produkte eingesetzt werden, für die ein Sicherheitsschwachstellen-Support durch den Hersteller besteht.

Ein solcher Support muss umfassen, dass der Hersteller

- das Produkt kontinuierlich auf eine Betroffenheit von Sicherheitsschwachstellen beobachtet und analysiert,
- über Art, Schweregrad und Ausnutzbarkeit von im Produkt entdeckten Schwachstellen umgehend informiert
- zeitnah Produktaktualisierungen oder wirksame Workarounds zur Behebung der Schwachstellen bereitstellt.

Der Sicherheitsschwachstellen-Support muss für die gesamte Dauer bestehen, in der das betroffene Produkt im Einsatz verbleibt.

Support-Phasen mit eingeschränktem Leistungsumfang

Viele Hersteller bieten für ihre Produkte als Option einen zeitlich ausgedehnten Support an, welcher über die für den allgemeinen Markt vorgesehene Support-Phase hinausgeht, aber oftmals mit Einschränkungen verbunden ist. Einige Hersteller definieren ihren Support grundlegend in Abstufungen, die Einschränkungen schon während der letzten Phase vor dem absoluten Endtermin des regulären Supports enthalten können.

Wird ein Produkt innerhalb von Support-Phasen eingesetzt, die Einschränkungen unterliegen, muss explizit sichergestellt werden, dass diese Einschränkungen die Verfügbarkeit des Sicherheitsschwachstellen-Support nicht beeinträchtigen.

Open Source Software und Hardware

Open Source Produkte werden häufig von freien Organisationen oder Communities entwickelt; entsprechend ist ein vertraglich vereinbarter Sicherheitsschwachstellen-Support unter Umständen nicht verfügbar. Grundsätzlich muss auch hier sichergestellt werden, dass von der Organisation/Community (oder einem offiziell durch diese eingesetzten Dritten) für das betroffene Produkt ein umfassendes Sicherheitsschwachstellen-Management betrieben wird, welches die oben genannten Kriterien erfüllt und als zuverlässig etabliert gilt.

Motivation: Hardware- und Software-Produkte, für die kein vollumfassender Sicherheitsschwachstellen-Support durch den Hersteller besteht, stellen ein Risiko dar. Dies bedeutet, dass ein Produkt nicht ausreichend auf eine Betroffenheit durch weiterentwickelte Angriffsformen oder neu entdeckte Schwächen technischer Umsetzungen geprüft wird. Analog werden bei vorhandenen Sicherheitsschwachstellen in einem Produkt keine Nachbesserungen (z. B. Updates, Patches) bereitgestellt. Hieraus resultiert ein System, dessen Schwachstellen nicht behoben werden können, so dass sie für einen Angreifer ausnutzbar bleiben, um das System zu kompromittieren oder es negativ zu beeinträchtigen.

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: 3.01-1/9.0

2.2. Container Linux Host Konfiguration

Dieser Abschnitt behandelt best practices zur Härtung des Container-Linux-Hosts. Es umfasst Anforderungen, um sicherzustellen, dass der Host sicher konfiguriert ist und eine robuste Grundlage für den Betrieb von Containern bietet.

Sicherheitsempfehlungen für den Linux Host , den Sie für die Ausführung von Container-Workloads verwenden möchten.

Dies sind Sicherheitsempfehlungen, die Sie befolgen sollten, um die Hostmaschine vorzubereiten, die Sie für die Ausführung containerisierter Workloads verwenden möchten.

Um eine solide und sichere Grundlage für die Ausführung zu schaffen, muss der Container-Linux-Host gehärtet werden.

Diese Empfehlungen sind möglicherweise nicht direkt für Container-Benutzer anwendbar, sondern eher für die Anbieter von Container-Runtimes.

Hier sind Beispiele, die berücksichtigt werden müssen, wenn Sie **Docker** verwenden:

- Stellen Sie sicher, dass eine separate Partition für Container erstellt wurde
- Stellen Sie sicher, dass nur vertrauenswürdige Benutzer den Docker-Daemon steuern dürfen
- Stellen Sie sicher, dass das Auditing für den Docker-Daemon konfiguriert ist
- Stellen Sie sicher, dass das Auditing für Docker-Dateien und -Verzeichnisse konfiguriert ist - /run/containerd
- Stellen Sie sicher, dass das Auditing für Docker-Dateien und -Verzeichnisse konfiguriert ist - /var/lib/docker
- Stellen Sie sicher, dass das Auditing für Docker-Dateien und -Verzeichnisse konfiguriert ist - /etc/docker
- Stellen Sie sicher, dass das Auditing für Docker-Dateien und -Verzeichnisse konfiguriert ist - docker.service
- Stellen Sie sicher, dass das Auditing für Docker-Dateien und -Verzeichnisse konfiguriert ist - containerd.sock
- Stellen Sie sicher, dass das Auditing für Docker-Dateien und -Verzeichnisse konfiguriert ist - docker.sock
- Stellen Sie sicher, dass das Auditing für Docker-Dateien und -Verzeichnisse konfiguriert ist - /etc/default/docker
- Stellen Sie sicher, dass das Auditing für Docker-Dateien und -Verzeichnisse konfiguriert ist - /etc/docker/daemon.json
- Stellen Sie sicher, dass das Auditing für Docker-Dateien und -Verzeichnisse konfiguriert ist - /etc/containerd/config.toml
- Stellen Sie sicher, dass das Auditing für Docker-Dateien und -Verzeichnisse konfiguriert ist - /etc/sysconfig/docker
- Stellen Sie sicher, dass das Auditing für Docker-Dateien und -Verzeichnisse konfiguriert ist - /usr/bin/containerd
- Stellen Sie sicher, dass das Auditing für Docker-Dateien und -Verzeichnisse konfiguriert ist - /usr/bin/containerd-shim
- Stellen Sie sicher, dass das Auditing für Docker-Dateien und -Verzeichnisse konfiguriert ist - /usr/bin/containerd-shim-runc-v1
- Stellen Sie sicher, dass das Auditing für Docker-Dateien und -Verzeichnisse konfiguriert ist - /usr/bin/containerd-shim-runc-v2
- Stellen Sie sicher, dass das Auditing für Docker-Dateien und -Verzeichnisse konfiguriert ist - /usr/bin/runc

*Falls Sie **containerd** oder eine andere Runtime verwenden, werden einige Docker-spezifische Dateien nicht verfügbar sein. Ein Härtung kann dennoch für nicht Docker-spezifische Dateien erreicht werden und muss durchgeführt werden.*

Motivation: Ein nicht gehärteter Host kann Schwachstellen aufweisen, die von Angreifern ausgenutzt werden könnten, um unbefugten Zugriff zu erlangen, Schadsoftware einzuschleusen oder Container-Workloads zu kompromittieren.

ID: 3.64-5/5.0

2.3. Container Daemon Konfiguration

Dieser Abschnitt beschreibt die Anforderungen zur Härtung des Container-Daemons.

Die Einstellungen in diesem Abschnitt betreffen ALLE Container-Instanzen.

Req 6 Der Container-Daemon muss gemäß Best Practices gehärtet werden.

Diese Anforderung listet die Empfehlungen auf, die das Verhalten des Container-Daemons ändern und sichern.

Diese Anforderungen sind möglicherweise nicht direkt für Container-Benutzer anwendbar, sondern eher für die Anbieter von Container-Runtimes.

Hier sind Beispiele, die berücksichtigt werden müssen, wenn Sie Docker verwenden:

- Führen Sie den Container-Daemon nach Möglichkeit als Nicht-Root-Benutzer aus
- Stellen Sie sicher, dass der Netzwerkverkehr zwischen Containern auf der Standard-Bridge eingeschränkt ist
- Stellen Sie sicher, dass das Protokollierungslevel auf 'info' gesetzt ist
- Stellen Sie sicher, dass Docker Änderungen an iptables vornehmen darf
- Stellen Sie sicher, dass keine unsicheren Registries verwendet werden
- Stellen Sie sicher, dass der aufs Storage-Treiber nicht verwendet wird
- Stellen Sie sicher, dass die TLS-Authentifizierung für den Container-Daemon konfiguriert ist
- Stellen Sie sicher, dass das Standard-ulimit angemessen konfiguriert ist
- Aktivieren Sie die Unterstützung für Benutzer-Namensräume
- Stellen Sie sicher, dass die Standard-cgroup-Nutzung bestätigt wurde
- Stellen Sie sicher, dass die Basisgerätegröße nicht geändert wird, bis sie benötigt wird
- Stellen Sie sicher, dass die Autorisierung für Docker-Client-Befehle aktiviert ist
- Stellen Sie sicher, dass zentrale und entfernte Protokollierung konfiguriert ist
- Stellen Sie sicher, dass Container daran gehindert werden, neue Privilegien zu erwerben
- Stellen Sie sicher, dass die Live-Wiederherstellung aktiviert ist
- Stellen Sie sicher, dass der Userland-Proxy deaktiviert ist
- Stellen Sie sicher, dass, falls zutreffend, ein benutzerdefiniertes seccomp-Profil auf Daemon-Ebene angewendet wird
- Stellen Sie sicher, dass experimentelle Funktionen nicht in Produktionsumgebungen implementiert werden

*Falls Sie **containerd** oder eine andere Runtime verwenden, werden einige Docker-spezifische Einstellungen nicht verfügbar sein.*

Motivation: Die Härtung des Container-Daemons nach Best Practices reduziert Sicherheitsrisiken und schützt vor Angriffen und unbefugtem Zugriff.

ID: 3.64-6/5.0

2.4. Container-Daemon-Konfigurationsdateien

Dieser Abschnitt beschreibt die Anforderungen für die Absicherung der Konfigurationsdateien des Container Daemon. Er befasst sich mit Dateien und Verzeichnissen, die mit dem Container Daemon zusammenhängen, sowie mit den Zugriffsrechten und dem Besitz.

Für das ordnungsgemäße und sichere Funktionieren des Daemon müssen Dateien und Verzeichnisse, die sensible Parameter enthalten können, geschützt werden.

Req 7 Die Container-Daemon-Konfigurationsdateien müssen gemäß Best Practices gehärtet werden.

Diese Anforderung behandelt die Dateiberechtigungen und den Besitz von Container-Daemon-bezogenen Dateien und Verzeichnissen. Es ist wichtig, die Dateien und Verzeichnisse, die sensible Parameter enthalten können, sicher zu halten, um das korrekte und sichere Funktionieren des Container-Daemons zu gewährleisten.

Diese Empfehlungen sind möglicherweise nicht direkt für Container-Benutzer anwendbar, sondern eher für die Anbieter von Container-Runtimes.

Hier sind Beispiele, die berücksichtigt werden müssen, wenn Sie **Docker** verwenden:

- Stellen Sie sicher, dass der Besitz der docker.service-Datei auf root:root gesetzt ist
- Stellen Sie sicher, dass die Berechtigungen der docker.service-Datei angemessen gesetzt sind
- Stellen Sie sicher, dass der Besitz der docker.socket-Datei auf root:root gesetzt ist
- Stellen Sie sicher, dass die Berechtigungen der docker.socket-Datei auf 644 oder restriktiver gesetzt sind
- Stellen Sie sicher, dass der Besitz des Verzeichnisses /etc/docker auf root:root gesetzt ist
- Stellen Sie sicher, dass die Berechtigungen des Verzeichnisses /etc/docker auf 755 oder restriktiver gesetzt sind
- Stellen Sie sicher, dass der Besitz der Registry-Zertifikatsdatei auf root:root gesetzt ist
- Stellen Sie sicher, dass die Berechtigungen der Registry-Zertifikatsdatei auf 444 oder restriktiver gesetzt sind
- Stellen Sie sicher, dass der Besitz der TLS-CA-Zertifikatsdatei auf root:root gesetzt ist
- Stellen Sie sicher, dass die Berechtigungen der TLS-CA-Zertifikatsdatei auf 444 oder restriktiver gesetzt sind
- Stellen Sie sicher, dass der Besitz der Docker-Serverzertifikatsdatei auf root:root gesetzt ist
- Stellen Sie sicher, dass die Berechtigungen der Docker-Serverzertifikatsdatei auf 444 oder restriktiver gesetzt sind
- Stellen Sie sicher, dass der Besitz der Docker-Serverzertifikatschlüsseldatei auf root:root gesetzt ist
- Stellen Sie sicher, dass die Berechtigungen der Docker-Serverzertifikatschlüsseldatei auf 400 gesetzt sind
- Stellen Sie sicher, dass der Besitz der Docker-Socket-Datei auf root:docker gesetzt ist
- Stellen Sie sicher, dass die Berechtigungen der Docker-Socket-Datei auf 660 oder restriktiver gesetzt sind
- Stellen Sie sicher, dass der Besitz der daemon.json-Datei auf root:root gesetzt ist
- Stellen Sie sicher, dass die Berechtigungen der daemon.json-Datei auf 644 oder restriktiver gesetzt sind
- Stellen Sie sicher, dass der Besitz der Datei /etc/default/docker auf root:root gesetzt ist
- Stellen Sie sicher, dass die Berechtigungen der Datei /etc/default/docker auf 644 oder restriktiver gesetzt sind
- Stellen Sie sicher, dass die Berechtigungen der Datei /etc/sysconfig/docker auf 644 oder restriktiver gesetzt sind
- Stellen Sie sicher, dass der Besitz der Datei /etc/sysconfig/docker auf root:root gesetzt ist
- Stellen Sie sicher, dass der Besitz der Containerd-Socket-Datei auf root:root gesetzt ist
- Stellen Sie sicher, dass die Berechtigungen der Containerd-Socket-Datei auf 660 oder restriktiver gesetzt sind

Falls Sie **containerd** oder eine andere Runtime verwenden, werden einige Docker-spezifische Einstellungen nicht verfügbar sein.

Motivation: Die Härtung der Konfigurationsdateien des Container-Daemons nach Best Practices schützt vor Fehlkonfigurationen und verhindert potenzielle Angriffe.

ID: 3.64-7/5.0

2.5. Container-Images und Build-Datei-Konfiguration

Dieser Abschnitt behandelt die Sicherheit von Container-Images und Build-Dateien.

Req 8	Container-Basisimages müssen gemäß Best Practices gehärtet werden.
-------	--

Empfehlungen für Container-Basis-Images und Build-Dateien.

Empfehlungen, die Sie für Container-Base-Images und Build-Dateien befolgen sollten, um sicherzustellen, dass Ihre containerisierte Infrastruktur sicher ist.

Diese Empfehlungen können gemäß dem Modell der geteilten Verantwortung sowohl für Container-Benutzer als auch für Anbieter von Container-Runtimes gelten.

- Stellen Sie sicher, dass ein Benutzer für den Container erstellt wurde
- Stellen Sie sicher, dass Content Trust für Docker aktiviert ist
- Stellen Sie sicher, dass HEALTHCHECK-Anweisungen zu Container-Images hinzugefügt wurden
- Stellen Sie sicher, dass Update-Anweisungen in Dockerfiles nicht alleine verwendet werden
- Stellen Sie sicher, dass setuid- und setgid-Berechtigungen entfernt werden
- Stellen Sie sicher, dass COPY anstelle von ADD in Dockerfiles verwendet wird

Motivation: Die Härtung von Container-Basisimages nach Best Practices verringert Sicherheitslücken und schützt Deployments vor Angriffen durch kompromittierte Images.

ID: 3.64-8/5.0

Req 9	Es müssen Fixe Tags verwendet werden um die Unveränderlichkeit zu gewährleisten.
-------	--

Verwenden Sie das spezifischste verfügbare Tag. Wenn ein Image mehrere Tags hat, z.B. :8 und :8.0.1 oder: 8.0.1-alpine, müssen sie das letzte verwenden, da es die spezifischste Referenz ist. Vermeiden Sie generische Tags wie "latest". [Bei Verwendung von unspezifischen Tags kann sich der Container bei einem Redeployment verändern und die Funktionsfähigkeit der Software einschränken.]

Da Tags im Container-Ökosystem grundsätzlich nicht unveränderlich (immutable) sind, kann ein Tag wie :8.0.1-alpine – so präzise er auch erscheinen mag – zukünftig durch ein anderes Image überschrieben werden. Eine tatsächliche Unveränderlichkeit und Nachvollziehbarkeit lässt sich daher nur durch die Angabe eines Digests erreichen. Sie können die Image-Version auf einen spezifischen Digest festlegen.

Motivation: Es kann von Vorteil sein, Images selbst zu erstellen, da Sie die Kontrolle über alle Komponenten behalten die mitgeliefert werden.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigte Modifikation von Daten
- Störung der Verfügbarkeit

ID: 3.64-9/5.0

Req 10	Es muss ein Benutzer mit den geringsten Berechtigungen verwendet werden.
--------	--

Wenn in einer Container-Datei kein Benutzer angegeben ist, wird der Container standardmäßig mit dem Root-Benutzer ausgeführt.

Der Container darf nicht mit dem Standard-Root-Benutzer ausgeführt werden.

Es muss ein spezifischer Benutzer verwendet werden.

Motivation: Verhindern Sie, dass Container als Standard-Root-Benutzer ausgeführt werden.

Für diese Anforderung sind folgende Bedrohungen relevant:

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

ID: 3.64-10/5.0

Req 11	Ein Containerspezifisches Host-Betriebssystem muss verwendet werden.
--------	--

Containerspezifische Host-Betriebssysteme müssen anstelle von Allzweck-Betriebssystemen verwendet werden.

Containerspezifische Host-Betriebssysteme verringern die Angriffsfläche, die ein System bietet.

Es gibt weniger Möglichkeiten, ein Container spezifisches Host-Betriebssystem wie z. B. : RedHat Atomic oder Core anzugreifen

Falls es nicht möglich ist ein Container spezifisches OS zu benutzen muss das Betriebssystem mindestens nach Linux/Windows SoCs gehärtet werden.

Motivation: Containerspezifische Betriebssysteme reduzieren die Angriffsfläche aufgrund ihres minimalistischen Ansatzes.

Umsetzungsbeispiel: Verwenden Sie minimale Container-Images.

Es gibt auch minimale Images innerhalb der DT <https://devsecops-tools.pages.devops.telekom.de/user-documentation/docs/mcc/baseimages/>, deren Verwendung bereits die Härtung gemäß SoCs zum Teil übernimmt und dazu beiträgt, mit einer soliden Grundlage zu beginnen.

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: 3.64-11/5.0

Req 12 Unnötige Pakete müssen vermieden werden.

Jedes Paket, das zu einem Container-Image hinzugefügt wird, erhöht die potenzielle Angriffsfläche. Durch die Minimierung der Anzahl der Pakete verringern Sie die Anzahl der Schwachstellen, die von Angreifern ausgenutzt werden könnten.

Motivation: Das Vermeiden unnötiger Pakete reduziert die Angriffsfläche und minimiert potenzielle Sicherheitsrisiken.

ID: 3.64-12/5.0

Req 13 Container müssen auf Schwachstellen untersucht werden.

Images müssen in der Registry auf bekannte Sicherheitslücken gescannt werden.
Container müssen auch zur Laufzeit gescannt werden.

Motivation: Finden Sie bekannte Sicherheitslücken in Containern.

Umsetzungsbeispiel: Kann Beispielsweise mit DTSP erfüllt werden :<https://dtsp.telekom-dienste.de/login?redirect=%2F>

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: 3.64-13/5.0

Req 14 Eine Secret Verwaltung muss zur Verfügung gestellt werden.

Secrets müssen außerhalb von Images gespeichert und bei Bedarf zur Laufzeit dynamisch bereitgestellt werden.

Motivation: Die meisten Orchestratoren, wie Docker Swarm und Kubernetes, verfügen über eine native Verwaltung von Secrets. Diese Orchestratoren sorgen nicht nur für die sichere Speicherung von Secrets und die "just in time"-Injektion in Container, sondern machen auch die Verwaltung einfacher. Der Orchestrator kann z.B. sicherstellen, dass nur der Webanwendungscontainer Zugriff auf ein bestimmtes Secret hat.

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: 3.64-14/5.0

Req 15	Nicht benötigte Dienste müssen deaktiviert werden.
--------	--

Nach der Installation von Systemen und Softwareprodukten sind häufig herstellerseitig voreingestellte, lokale oder aus dem Netz erreichbare Dienste aktiv, die für den Betrieb und die Funktionalität des konkreten Systems im vorgesehenen Einsatzumfeld nicht erforderlich sind.

Auf einem System dürfen jedoch grundsätzlich nur die tatsächlich benötigten Dienste aktiv sein.

Entsprechend müssen unmittelbar nach der Installation sämtliche auf einem System nicht erforderliche Dienste vollständig deaktiviert werden. Es muss sichergestellt werden, dass diese Dienste auch nach einem Neustart des Systems weiterhin deaktiviert bleiben.

Motivation: Aktive Dienste, die nicht benötigt werden, vergrößern unnötig die Angriffsfläche eines Systems und in unmittelbarer Folge das Risiko einer erfolgreichen Kompromittierung. Dieses Risiko kann zusätzlich nochmals erhöht sein, falls - wie bei nicht benötigten Diensten häufig beobachtet - eine gezielte Betrachtung und Optimierung der Konfiguration hinsichtlich Sicherheit nicht ausreichend stattfindet.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigte Nutzung von Diensten oder Ressourcen
- Angriffe motiviert und erleichtert durch Informationspreisgabe oder erkennbare Sicherheitsschwachstellen

ID: 3.01-5/9.0

Req 16	Eingesetzte Software muss aus vertrauenswürdigen Quellen bezogen und auf Integrität geprüft werden.
--------	---

Die auf dem System eingesetzte Software muss aus vertrauenswürdigen Quellen bezogen und vor der Installation auf ihre Integrität geprüft werden.

Diese Anforderung bezieht sich auf sämtliche Arten von Software:

- Firmware sowie Microcode für Hardware-Komponenten
- Betriebssysteme
- Software-Bibliotheken
- Applikationen und Anwendungssoftware
- vorintegrierte Anwendungslösungen, wie z. B. Software Appliances oder Container

sowie weitere Software, die ggf. zum Einsatz kommt.

Vertrauenswürdige Quellen

Als vertrauenswürdige Bezugsquellen gelten allgemein:

- die offiziellen Vertriebs- und Bereitstellungskanäle des Herstellers
- dritte Distributoren, sofern sie durch den Hersteller autorisiert und legitimer Bestandteil von dessen Bereitstellungskanälen sind
- Internet-Downloads, sofern diese von offiziellen Bereitstellungs-Servern des Herstellers oder autorisierter Distributoren erfolgen

(1) Bietet der Bereitstellungs-Server verschiedene Formen von Downloads an, müssen durch Verschlüsselung bzw. kryptografische Signaturen geschützte Downloads jenen ohne einen solchen Schutz vorgezogen wer-

den.

(2) Sichert der Bereitstellungs-Server die Transport-Schicht mittels kryptografischer Protokolle (z. B. HTTPS, sftp), müssen die zugehörig präsentierten Server-Zertifikate oder Server-Keys/Fingerprints bei jedem Download validiert werden, um die Identität des Bereitstellungs-Servers zu bestätigen; schlägt die Validierung fehl, ist der Download abubrechen und der Bereitstellungs-Server als nicht vertrauenswürdige Quelle zu bewerten.

Integritätsprüfung

Die Integritätsprüfung soll sicherstellen, dass die erhaltene Software frei von Manipulationen und Schadsoftware-Befall ist. Sofern vorhanden, müssen zur Prüfung die vom Hersteller implementierten Mechanismen verwendet werden.

Valide Mechanismen sind:

- physikalische Siegel bzw. fest aufgebrachte Echtheitszertifikate (bei Bereitstellung der Software auf physikalischen Datenträgern)
- Abgleich von kryptografischen Hash Werten (z. B. SHA256, SHA512) der erhaltenen Software gegen Soll-Werte, die der Hersteller separat benennt
- Prüfung von kryptografischen Signaturen (z. B. GPG, Zertifikate), mit denen der Hersteller seine Software versieht

Zusätzlich ist eine Überprüfung der Software mittels eines Anti-Virus- bzw. Anti-Malware Scanners empfohlen (falls der Hersteller keinen der zuvor erwähnten Integritätsschutz-Mechanismen für seine Software implementiert hat, ist diese Überprüfung verpflichtend).

Erweiterte Integritätsprüfung bei Bezug von Software aus öffentlichen Registries

Öffentliche Registries ermöglichen es Entwicklern, beliebige eigene Software-Projekte zur Nutzung bereitzustellen. Die Bandbreite umfasst sowohl Projekte von namhaften Anbietern mit kontrollierten Entwicklungsprozessen als auch von kleineren Anbietern oder Amateur-Entwicklern.

Beispiele für solche Registries sind:

- Code Registries (z. B. GitHub, Bitbucket, SourceForge, Python Package Index)
- Container Registries (z. B. Docker Hub)

Software aus öffentlichen Registries muss vor dem Einsatz einer erweiterten Integritätsprüfung unterzogen werden. Zusätzlich zu den im vorherigen Abschnitt beschriebenen Bestandteilen der Integritätsprüfung soll die erweiterte Prüfung explizit sicherstellen, dass die Software ihre Funktion tatsächlich wie beschrieben erfüllt, keine inhärenten Sicherheits-Risiken wie etwa gezielt implementierte Malware-Funktionen enthält und nicht von bekannten Sicherheits-Schwachstellen betroffen ist. Sofern die Software direkte Abhängigkeiten zu dritten Software-Projekten aufweist (sog. "Dependencies", sehr typisch in Open Source Software), die zum Einsatz der Software ebenfalls bezogen und installiert werden müssen, sind diese in die erweiterte Integritätsprüfung mit einzubeziehen.

Methoden für eine erweiterte Integritätsprüfung können beispielsweise sein:

- strikte Validierung der Projekt-/Paket-Namen (Vermeidung der Verwechslung mit maliziösen, gezielt nachgeahmten Software-Projekten)
- dynamische Code Analyse / strukturierte Funktionsprüfung in einer Test-Umgebung
- statische Code Analyse mittels eines Linters (z. B. Splint, JSLint, Pylint)
- Untersuchung mittels eines Sicherheitsschwachstellen-Scanners (z. B. Qualys, Nessus)
- Untersuchung mittels eines Container Security Scanners (z. B. JFrog Xray, Harbor, Clair, Docker Scan)
- Untersuchung mittels eines SCA (Software Composition Analysis) Tools bzw. Dependency-Scanners (z. B. OWASP Dependency Check, Snyk)

Die Prüfungsmethoden sind passend zur exakten Form der Software-Bereitstellung (Quellcode, Binaries/Artefakte, Container) auszuwählen und angemessen zu kombinieren.

Motivation: Softwarelieferketten enthalten diverse Angriffsvektoren. Ein Angreifer kann an verschiedenen Stellen ansetzen, um Software zu manipulieren oder gezielt eigene Routinen einzubringen und die Zielumgebung, in der die Software anschließend zum Einsatz gebracht wird, zu schädigen oder unter seine Kontrolle zu bringen. Der Angriff

kann auf dem Transport- bzw. Übertragungsweg geschehen oder auch auf die Bereitstellungsquelle selbst ausgeübt werden. Entsprechend erleichtert wird ein Angriff, falls Software nicht über offizielle und kontrollierte Quellen bezogen wird oder eine Integritätsprüfung ausgelassen wird.

Besonderes Risiko zeigt sich für Software, die aus öffentlichen Registries bezogen wird, da diese jedermann für die Bereitstellung von Software-Projekten offenstehen. Perfide Angriffsmethoden sind inzwischen bekannt, in denen der Angreifer zunächst eine Zeitlang sicherheitstechnisch völlig unauffällige, funktionale Software bereitstellt und sobald diese sich etabliert und eine gewisse Verbreitung gefunden hat, in zukünftigen Versionen gezielt versteckten Schadcode integriert. Andere Methoden setzen auf ähnlich klingende Projektnamen zu weit verbreiteten Bestandsprojekten oder überstimmende Versionsnummern, um manipulierte Software in darauf aufbauende Lösungen zu injizieren.

Umsetzungsbeispiel: Bezug der Software über die offiziellen Bereitstellungskanäle des Herstellers. Nach Erhalt der Software umgehend Prüfung auf Integrität mittels kryptografischer Checksummen, wie vom Hersteller bereitgestellt, sowie Freiheit von bekannter Schadsoftware mittels Anti-Malware- / Anti-Virus-Scanner. Ablage der geprüften Software auf einer internen, geschützten Dateiablage und weitere Nutzung (z. B. Rollout auf die Zielsysteme) nur noch von dort.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigte Modifikation von Daten
- Unbemerkt durchführbare Angriffe
- Angriffe motiviert und erleichtert durch Informationspreisgabe oder erkennbare Sicherheitsschwachstellen

ID: 3.01-2/9.0

Req 17 Auf dem System darf ausschließlich nur benötigte Software eingesetzt werden.

In den Hersteller-seitig bereitgestellten Installationsroutinen für Software sind oftmals Komponenten oder einzelne Bestandteile der Software als Standard-Installation vorausgewählt, die für den Betrieb und die Funktion eines spezifischen Systems nicht notwendig sind. Hierzu zählen auch Teile einer Software, die als Anwendungsbeispiele (z. B. Default-Web-Seiten, Beispieldatenbanken, Testdaten) mit installiert, aber typischerweise im Anschluss nicht verwendet werden.

Solche Komponenten müssen bei der Installation des Systems gezielt abgewählt (nicht mit installiert) werden oder - falls eine Abwahl während der Installation nicht möglich ist - unmittelbar im Anschluss entfernt werden.

Des Weiteren darf grundsätzlich keine Software auf einem System eingesetzt sein, die für den Betrieb, die Wartung oder die Funktion des Systems nicht erforderlich ist.

Motivation: Schwachstellen in der Software eines Systems sind Einfallstore für Angreifer. Durch die Deinstallation nicht benötigter Komponenten können die potentiellen Angriffsflächen deutlich reduziert werden.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigte Nutzung von Diensten oder Ressourcen
- Angriffe motiviert und erleichtert durch Informationspreisgabe oder erkennbare Sicherheitsschwachstellen

ID: 3.01-3/9.0

Req 18 Die Erreichbarkeit von aktivierten Diensten muss eingeschränkt werden.

Ein bereitgestellter Dienst muss grundsätzlich auf allen Schnittstellen des Systems vollständig deaktiviert werden, über die eine Erreichbarkeit des Dienstes für den ordnungsgemäßen Betrieb des Systems nicht erforderlich ist. Die Deaktivierung ist primär durch eine entsprechende Konfiguration des Dienstes bzw. Betriebssystems umzusetzen. In Fällen, in denen die zu Verfügung stehenden Konfigurationsoptionen eine Deaktivierung auf einzelnen Schnittstellen nicht ermöglichen, darf stattdessen auch ein lokaler Filter ("Host Firewall") auf dem System eingesetzt werden, um Zugriffe auf den Dienst über nicht erforderliche Schnittstellen zu blockieren.

Die Erreichbarkeit eines Dienstes über die erforderlichen Schnittstellen muss zudem auf legitime Kommunikationspartner eingeschränkt werden. Die Einschränkung ist durch eine entsprechende Konfiguration des Dienstes bzw. Betriebssystems oder mittels eines lokalen Filters ("Host Firewall") umzusetzen. Alternativ darf diese Aufgabe an ein netzseitiges Filterelement ausgelagert werden, sofern sich das System in einem separaten Netzwerk-Segment befindet und die Kommunikation mit diesem Segment ausschließlich über das netzseitige Filterelement möglich ist.

Motivation: Mittels Deaktivierung von Diensten auf Schnittstellen, über die eine Erreichbarkeit nicht notwendig ist, sowie durch die Einschränkung möglicher Kommunikationspartner, kann die Angriffsfläche, die ein System bietet, stark reduziert werden.

Umsetzungsbeispiel: Ein SNMP-Dienst, über den ein System gemonitort wird, ist ausschließlich auf der dedizierten Management-Netzwerk-Schnittstelle des Systems aktiviert. Über eine Firewall ist zusätzlich reglementiert, dass einzig das legitime Monitoring System der Infrastruktur-Umgebung diesen Dienst erreichen kann.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigte Nutzung von Diensten oder Ressourcen
- Angriffe motiviert und erleichtert durch Informationspreisgabe oder erkennbare Sicherheitsschwachstellen

ID: 3.01-6/9.0

Req 19 Nicht benötigte Funktionen in der eingesetzten Software und Hardware müssen deaktiviert werden.

Während der initialen Installation von Software können zunächst Funktionen standardmäßig aktiviert worden sein, die für den Betrieb und die Funktionalität des konkreten Systems nicht notwendig sind. Funktionen sind meistens ein fester Bestandteil der Software, der nicht einzeln gelöscht oder deinstalliert werden kann.

Solche Funktionen müssen unmittelbar nach der initialen Installation über die Konfigurationseinstellungen der Software deaktiviert werden, so dass sie auch nach einem Neustart des Systems dauerhaft deaktiviert bleiben.

Bereits vor der Auslieferung oder während der initialen Inbetriebnahme sind möglicherweise auch in der Hardware zunächst Funktionen standardmäßig aktiviert worden, die für den Einsatz des konkreten Systems nicht benötigt werden. Solche Funktionen, wie beispielsweise nicht benötigte Schnittstellen, müssen ebenfalls unmittelbar nach der initialen Inbetriebnahme dauerhaft deaktiviert werden.

Motivation: Die Hardware oder Software eines Systems enthält oft aktivierte Funktionen, die nicht genutzt werden. Solche Funktionen können eine unnötige Angriffsfläche für Manipulationen darstellen. Des Weiteren entsteht darüber ein Potential, dass unautorisiert auf Bereiche oder Daten des Systems zugegriffen werden kann.

Umsetzungsbeispiel: [Beispiel 1]

Deaktivierung von Debugging-Funktionen in einer Software, die im Falle der Entstörung genutzt werden, aber während des normalen Betriebs nicht aktiv sein müssen.

[Beispiel 2]

Deaktivierung von ungenutzten Netzwerk-Schnittstellen eines Servers.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigte Nutzung von Diensten oder Ressourcen
- Angriffe motiviert und erleichtert durch Informationspreisgabe oder erkennbare Sicherheitsschwachstellen

ID: 3.01-4/9.0

Req 20 Die Berechtigungen von Benutzern und Anwendungen müssen auf den zur Erfüllung ihrer Aufgaben benötigten Umfang beschränkt werden.

Die Berechtigungen auf einem System müssen so weit eingeschränkt werden, dass ein Benutzer nur auf Daten zugreifen und Funktionen nutzen kann, die er im Rahmen seiner Arbeit benötigt. Entsprechende Berechtigungen sind auch für den Zugriff auf Dateien zu vergeben, die Bestandteil des Betriebssystems oder von Anwendungen sind oder von diesen erzeugt werden (z. B. Konfigurations- und Protokollierungsdateien).

Neben dem Zugriff auf Daten muss auch die Ausführung von Anwendungen und deren Bestandteilen mit möglichst niedrigen Berechtigungen erfolgen. Anwendungen sollten nicht mit Administrator- oder Systemberechtigungen ausgeführt werden.

Motivation: Werden einem Benutzer zu weitreichende Berechtigungen auf einem System gewährt, kann er in einem Umfang auf Daten und Anwendungen zugreifen, der zur Erfüllung der zugewiesenen Aufgaben nicht erforderlich ist. Dies erzeugt ein unnötig erhöhtes Risiko im Falle eines Missbrauchs, insbesondere auch, falls der Benutzer oder dessen Benutzerkonto durch einen Angreifer kompromittiert werden.

Anwendungen, die über zu weitreichende Berechtigungen verfügen, können von einem Angreifer missbraucht werden, um unbefugte Zugänge zu erhalten oder zu erweitern und damit einen Zugriff auf schutzbedürftige Daten und Systembereiche zu erlangen.

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
- Abstreiten von durchgeführten Aktionen

ID: 3.01-23/9.0

2.6. Container Runtime Konfiguration

Dieser Abschnitt konzentriert sich auf die Sicherheit der Container-Laufzeit-Konfiguration.

Req 21	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-Namespace

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-Namespace

Im Pod-Spec muss hostIPC: false gesetzt werden.

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

Verbot der Nutzung des Host-Netzwerk-Namespace

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-Angriffen.

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 22	Die Container-Laufzeitkonfiguration muss gemäß Best Practices gehärtet werden.
--------	--

Für Container können Laufzeitparameter angegeben werden, die Sicherheitsfolgen auf den Host und die darauf laufenden Container haben könnten. Daher sollte überprüft werden wie Container gestartet werden und welche Parameter ihnen zugeordnet sind.

Container müssen sicherstellen, dass Folgendes gewährleistet ist:

- Stellen Sie sicher, dass der Swarm-Modus nicht aktiviert ist, wenn er nicht benötigt wird
- Stellen Sie sicher, dass, falls zutreffend, ein AppArmor-Profil aktiviert ist
- Stellen Sie sicher, dass, falls zutreffend, SELinux-Sicherheitsoptionen gesetzt sind
- Stellen Sie sicher, dass die Linux-Kernel-Fähigkeiten innerhalb der Container eingeschränkt sind
- Stellen Sie sicher, dass keine privilegierten Container verwendet werden
- Stellen Sie sicher, dass keine sensiblen Hostsystemverzeichnisse auf Container gemountet sind
- Stellen Sie sicher, dass sshd nicht innerhalb von Containern ausgeführt wird
- Stellen Sie sicher, dass keine privilegierten Ports innerhalb von Containern zugeordnet sind
- Stellen Sie sicher, dass nur die benötigten Ports auf dem Container offen sind
- Stellen Sie sicher, dass der Netzwerk-Namensraum des Hosts nicht freigegeben ist
- Stellen Sie sicher, dass die Speichernutzung für Container begrenzt ist
- Stellen Sie sicher, dass die CPU-Priorität auf Containern angemessen gesetzt ist
- Stellen Sie sicher, dass das Root-Dateisystem des Containers als schreibgeschützt eingehängt ist
- Stellen Sie sicher, dass eingehender Containerverkehr an eine spezifische Host-Schnittstelle gebunden ist
- Stellen Sie sicher, dass die Container-Neustart-Policy 'on-failure' auf '5' gesetzt ist
- Stellen Sie sicher, dass der Prozess-Namensraum des Hosts nicht freigegeben ist
- Stellen Sie sicher, dass der IPC-Namensraum des Hosts nicht freigegeben ist
- Stellen Sie sicher, dass Hostgeräte nicht direkt Containern ausgesetzt sind
- Stellen Sie sicher, dass das Standard- ulimit zur Laufzeit überschrieben wird, falls nötig
- Stellen Sie sicher, dass der Mount-Propagation-Modus nicht auf 'shared' gesetzt ist
- Stellen Sie sicher, dass der UTS-Namensraum des Hosts nicht freigegeben ist
- Stellen Sie sicher, dass das Standard-seccomp-Profil nicht deaktiviert ist
- Stellen Sie sicher, dass docker exec-Befehle nicht mit der privilegierten Option verwendet werden
- Stellen Sie sicher, dass docker exec-Befehle nicht mit der Option user=root verwendet werden
- Stellen Sie sicher, dass die cgroup-Nutzung bestätigt ist
- Stellen Sie sicher, dass der Container daran gehindert wird, zusätzliche Privilegien zu erwerben

- Stellen Sie sicher, dass die Container-Gesundheit zur Laufzeit überprüft wird
- Stellen Sie sicher, dass Docker-Befehle immer die neueste Version ihres Images verwenden
- Stellen Sie sicher, dass das PIDs-cgroup-Limit verwendet wird
- Stellen Sie sicher, dass die Docker-Standardbrücke "docker0" nicht verwendet wird
- Stellen Sie sicher, dass die Benutzer-Namensräume des Hosts nicht freigegeben sind
- Stellen Sie sicher, dass der Docker-Socket nicht innerhalb von Containern eingehängt ist
- Stellen Sie sicher, dass Container ein schreibgeschütztes Dateisystem haben

Motivation: Die Härtung der Container-Runtime-Konfiguration nach Best Practices schützt vor Fehlkonfigurationen und reduziert potenzielle Angriffsflächen.

ID: 3.64-22/5.0

Req 23 Die Namespace-Isolation muss gewährleistet sein.

Ressourcen, mit denen ein Container interagieren kann, müssen begrenzt sein. Dies umfasst Dateisysteme, Netzwerkschnittstellen und Interprozess

Kommunikation, Hostnamen, Benutzerinformationen und Prozesse.

Stellen Sie sicher, dass sich Apps und Prozesse, die sich in einem Container befinden auf die ihm zugewiesenen physischen und virtuellen Ressourcen beschränken.

Die Namespace-Isolierung bietet jedem Container einen eigenen Netzwerkstapel mit eindeutigen Schnittstellen und IP-Adressen.

Motivation: Container isolieren

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: 3.64-23/5.0

Req 24 Sicherheitszonen müssen eingerichtet werden.

Container, die verschiedenen Sicherheitszonen zugeordnet sind (z. B. DMZ-Zone, Anwendungs- / Datenbankzone) müssen

auf verschiedenen Hosts / Host-Pools bereitgestellt werden. Ein Pool bezieht sich auf eine Gruppe von mindestens einer oder mehreren VMs. Im Falle der Verwendung von Container as a Service (CaaS) -Plattformen können diese Pools für mehrere Anwendungen gemeinsam genutzt werden. Für kritische Anwendungen müssen dedizierte Hosts / Host-Pools. Wenn der einzige exponierte Dienst ein Ingress-Controller (z. B. Nginx) ist, der hinsichtlich Sicherheit gehärtet ist, können Sie dies implementieren, indem Sie nur Container-spezifische Maßnahmen verwenden, wie z.B. Pod Security Policies / Network Security Policies.

Motivation: Die Unterteilung von Containern nach Zweck und Bedrohungslage bietet eine zusätzliche Separierung. Durch das Gruppieren von Containern auf diese Weise ist es für einen Angreifer, der eine der Gruppen kompromittiert, schwieriger diesen Zugriff auszuweiten.

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: 3.64-24/5.0

Req 25 Linux-Kernel-Funktionen müssen innerhalb von Containern eingeschränkt werden.

Standardmäßig starten Container mit einem eingeschränkten Satz von Linux-Kernel-Funktionen. Das bedeutet, dass jedem Prozess die erforderlichen Fähigkeiten gewährt werden können, anstatt ihm Root zu geben. Unter Verwendung der Linux-Kernel-Funktionen müssen Prozesse im Allgemeinen nicht als Root ausgeführt werden. Alle Fähigkeiten, die für die korrekte Funktion des Behälters nicht erforderlich sind, müssen entfernt werden. Insbesondere sollte in dem von Docker bereitgestellten Standardfunktionssatz die NET_RAW-Funktion entfernt werden, wenn dies nicht ausdrücklich erforderlich ist.

Übersicht der Linux Kernel Funktionen für Docker : <https://docs.docker.com/engine/security/#linux-kernel-capabilities>

Motivation: Der Linux-Kernel ist in der Lage, die Privilegien des Root-Benutzers in verschiedene Einheiten aufzuteilen, die als Capabilities bezeichnet werden.

Minimierung der Angriffsfläche durch das Entfernen von Capabilities, die für den Betrieb des Containers nicht benötigt werden.

Umsetzungsbeispiel: Beispiel für Docker Docker :

```
$ docker run --cap-drop ALL
```

Danach können via **-cap-add** flag neue Capabilities hinzugefügt werden.

ID: 3.64-25/5.0

2.7. Container Security Operationen

Req 26 Image-Sprawl muss vermieden werden

Jedes zusätzliche Image erhöht die potenzielle Angriffsfläche. Durch die Begrenzung der Anzahl der Images minimieren Sie die Anzahl der Schwachstellen, die von Angreifern ausgenutzt werden könnten.

Motivation: Das Vermeiden von Image Sprawl reduziert die Angriffsfläche und erleichtert die Verwaltung sowie Absicherung der Container-Umgebung.

ID: 3.64-26/5.0

Req 27 Container-Sprawl muss vermieden werden.

Jeder zusätzliche Container erhöht die potenzielle Angriffsfläche. Durch die Begrenzung der Anzahl der Container minimieren Sie die Anzahl der Schwachstellen, die von Angreifern ausgenutzt werden könnten.

Motivation: Das Vermeiden von Container Sprawl erleichtert die Verwaltung, reduziert die Angriffsfläche und minimiert Sicherheitsrisiken.

ID: 3.64-27/5.0

2.8. Registry

Dieser Abschnitt beschreibt die Anforderungen an Container-Registries.

Req 28 Registries müssen über eine Benutzerverwaltung verfügen.

Für jede Registry muss eine Mandantenfähigkeit gewährleistet sein, entweder durch die Verwendung dedizierter Umgebungen oder durch eine Benutzerverwaltung auf der gleichen Instanz.

Motivation: Trennen und verwalten des Zugangs zu Registries.

ID: 3.64-28/5.0

Req 29 Images von externen Registries müssen zuerst in eine interne Registry heruntergeladen werden.

Images von externen Registries müssen zuerst in eine interne Registry heruntergeladen werden, damit sie auf Schwachstellen überprüft werden können
Innerhalb der DT können Sie die Magenta Trusted Registry <https://mtr.devops.telekom.de/> verwenden.

Motivation: Fortlaufende, ständig aktualisierte, zentralisierte Berichterstellung von Images. Eine Überprüfung auf Schwachstellen und Compliance.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigter Zugang zum System
- Unberechtigte Modifikation von Daten

ID: 3.64-29/5.0

Req 30 Verbindungen zu Registries müssen verschlüsselt sein.

Registries müssen so konfiguriert werden, dass sie von Entwicklungstools, Orchestratoren und Container-Runtimes nur verschlüsselt erreicht werden.

Motivation: Ziel ist es, sicherzustellen, dass alle Daten zwischen vertrauenswürdigen Endpunkten während des Transits verschlüsselt sind.

Für diese Anforderung sind folgende Bedrohungen relevant:

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

ID: 3.64-30/5.0

Req 31 Es muss eine ausreichende Authentifizierung und Autorisierung verwendet werden.

Jeder Schreibzugriff auf eine Registry erfordert eine Authentifizierung. Entwickler sollten Images nur in für Sie bestimmten Repositories speichern können.
Images müssen von autorisierten Personal signiert und erst danach in eine Registry verschoben werden.

Motivation: Stellen Sie sicher, dass nur Images von vertrauenswürdigen Stellen zur Registry hinzugefügt werden können.

Für diese Anforderung sind folgende Bedrohungen relevant:

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

ID: 3.64-31/5.0

3. Technischer Basisschutz von IT-/NT-Systemen

3.1. Protokollierung

Req 32	Die Systemzeit muss zu einer exakten Referenzzeit (Zeitnormal) synchronisiert werden.
--------	---

Als Zeitnormal ist eine Zeitgeber-Quelle heranzuziehen, die ein Zeitsignal auf Basis der koordinierten Weltzeit ("UTC" = "Universal Time Coordinated") liefert.

Anmerkung: die anhand der Zeitgeber-Quelle zu UTC synchronisierte Systemzeit darf auf dem lokalen System mittels entsprechender Zeitzone-Konfiguration bei der Ausgabe von Zeitinformationen in die Ortszeit transformiert werden, sofern die so stattfindende Zeitzoneanpassung eindeutig erkennbar und nachvollziehbar ist.

Systeme, die zu einer gemeinsamen Sicherheits-Domäne gehören, müssen sich einheitlich an derselben Zeitgeber-Quelle synchronisieren.

Motivation: Zeitsynchronisation kann auf einem System eine technische Voraussetzung sein, um zeitkritische Abhängigkeiten in beispielsweise der Nutzung von Zertifikaten oder Authentisierungsmechanismen korrekt bedienen zu können. Dringend erforderlich sind außerdem exakte Zeitstempel in Protokollierungseinträgen, damit diese bei Sicherheitsvorfällen oder Problemanalysen in eindeutiger zeitlicher Relation auswertbar sind.

Umsetzungsbeispiel: Einige zulässige Zeitgeber-Quellen:

- vertrauenswürdig eingestuftes NTP ("NetworkTimeProtocol") Server im IP-Netzwerk
- DCF77-Funksignal über physisch angebundenes Empfängermodul
- GPS-Funksignal über physisch angebundenes Empfängermodul

Für diese Anforderung sind folgende Bedrohungen relevant:

- Störung der Verfügbarkeit
- Abstreiten von durchgeführten Aktionen
- Unbemerkt durchführbare Angriffe

ID: 3.01-32/9.0

Req 33	Sicherheitsrelevante Ereignisse müssen mit einem exakten Zeitstempel und einer eindeutigen Systembezeichnung protokolliert werden.
--------	--

Sicherheitsrelevant sind jegliche Ereignisse, die Auswirkungen auf die Informationssicherheit (Vertraulichkeit, Integrität, Verfügbarkeit) eines Systems haben können oder dessen mißbräuchliche Verwendung anzeigen. Systeme müssen das Auftreten von sicherheitsrelevanten Ereignissen protokollieren.

Um eine Zuordnung und Auswertung dieser Ereignisse zu ermöglichen, müssen diese zusammen mit einer eindeutigen Systembezeichnung (z. B. Hostname, IP- oder MAC-Adresse) und dem genauen Zeitpunkt des Auftretens ("Zeitstempel") protokolliert werden. Der Zeitstempel eines protokollierten Ereignisses muss mindestens folgende Informationen beinhalten:

- Datum des Ereignisses (Jahr, Monat, Tag)
- Uhrzeit des Ereignisses (Stunden, Minuten, Sekunden)
- Zeitzone, auf die sich diese Zeitangaben beziehen

Die Sicherheitsrelevanz konkreter Ereignisse ergibt sich individuell aus Art und Funktion des Systems sowie den darin verarbeiteten Daten. Daher kann keine einheitliche Liste aus Ereignissen, die Sicherheitsrelevanz besitzen, vorgegeben werden. Relevante Ereignisse müssen passend zum System identifiziert werden. Die folgende Auflistung enthält typische sicherheitsrelevante Ereignisse und ist entsprechend als eine grundlegende Orientierungshilfe bei der Identifikation der sicherheitsrelevanten Ereignisse für ein bestimmtes System zu verstehen.

Sicherheitsrelevantes Ereignis	Informationen, die zum Ereignis zu protokollieren sind
--------------------------------	--

Fehlerhafte Anmeldeversuche	• betroffenes Benutzerkonto, • Anzahl der Fehlversuche • Quelle des Zugriffs (IP-Adresse, Client-ID/Client-Name, lokale Konsole, ...)
Systemzugriffe über Benutzerkonten (einschliesslich System-Konten bzw. technischer Benutzerkonten)	• betroffenes Benutzerkonto • Startzeit/Endzeit des Zugriffs (Anmeldung/Abmeldung) • Quelle des Zugriffs (IP-Adresse, Client-ID/Client-Name, lokale Konsole, ...)
Änderungen an Benutzerkonten (einschliesslich System-Konten bzw. technischer Benutzerkonten)	• Benutzerkonto, mit dem die Änderung vorgenommen wurde • Benutzerkonto, an dem die Änderung vorgenommen wurde • Art der vorgenommenen Änderung (Neueinrichtung, Aktivierung, Deaktivierung, Löschung, Passwort-Rücksetzung, ...)
Änderungen der Rollenzuweisungen/Berechtigungen von Benutzerkonten (einschliesslich System-Konten bzw. technischer Benutzerkonten)	• Benutzerkonto, mit dem die Änderung vorgenommen wurde • Benutzerkonto, an dem die Änderung vorgenommen wurde • Bezeichnung der betroffenen Rolle / Typ der betroffenen Berechtigung • Art der vorgenommenen Änderung (Rolle/Berechtigung hinzugefügt, Rolle/Berechtigung entfernt, ...)
Nutzung/Initiierung von kritischen Systemfunktionen	• Benutzerkonto, mit dem die Nutzung vorgenommen wurde • Bezeichnung/Typ der genutzten Systemfunktion
Zugriff auf sensible Daten ("Sensitive Data"), die im System verarbeitet werden	• Benutzerkonto, mit dem der Zugriff vorgenommen wurde • Bezeichnung des betroffenen Datenfelds (der Inhalt des Datenfelds sollte dabei nicht mit protokolliert werden, um einen Datenabfluss über Log-Dateien zu verhindern) • Art des Zugriffs (Lesen, Ändern, Löschen, ...)
Abnormaler Anstieg in der Auslastung von Systemressourcen, wie etwa Arbeitsspeicher, CPU, Netzwerk oder Festplattenzugriffen über einen längeren Zeitraum	• Schwellwert der betroffenen Systemressource • Dauer der Überschreitung des Schwellwerts • Höhe der Überschreitung des Schwellwerts Es müssen abhängig vom individuellen System sinnvolle Schwellwerte definiert werden.
Sonstige Ereignisse, die für versuchte oder erfolgreiche Angriffe gegen das System Hinweise liefern können *)	• individuell abhängig von der Art des Ereignisses

*) Dies erfordert eine Identifizierung der wichtigsten, für das System typischerweise zu erwartenden Angriffsformen und die Sicherstellung, dass zu deren Erkennung relevante Ereignisse durch das System tatsächlich protokolliert werden. In der Regel kann das nur durch den Entwickler/Hersteller des Systems gewährleistet werden.

Bei der Protokollierung müssen die jeweils geltenden gesetzlichen und betrieblichen Bestimmungen beachtet werden. Letztere umfassen insbesondere auch Vereinbarungen, die mit den betrieblichen Sozialpartnern getroffen wurden. Diese Bestimmungen besagen unter anderem, dass die Protokollierung von Ereignissen nur zweckgebunden erfolgen darf. Die Protokollierung von Ereignissen zum Zweck der Arbeitskontrolle von Mitarbeitern ist nicht zulässig.

Zudem muss - wie für jegliche Daten, die von einem System verarbeitet werden - auch für Protokollierungsdaten ein angemessener Schutzbedarf berücksichtigt und umgesetzt werden; dies gilt für die Speicherung, die Übertragung sowie den Zugriff. Insbesondere falls Wirkdaten in den Protokollierungsdaten enthalten sind, muss derselbe Schutzbedarf berücksichtigt werden, der auch für die reguläre Verarbeitung dieser Wirkdaten innerhalb des Quell-Systems Anwendung findet.

Motivation: Die Protokollierung von sicherheitsrelevanten Ereignissen ist eine Grundvoraussetzung, um laufende Angriffe oder auch Angriffe im Nachhinein erkennen zu können. Nur so ist es möglich, sinnvolle Maßnahmen zur Erhaltung oder auch Wiederherstellung der Systemsicherheit durchzuführen. Des Weiteren dienen die Protokollierungsdaten der Beweissicherung, um rechtlich gegen Angreifer vorgehen zu können.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Abstreiten von durchgeführten Aktionen
- Unbemerkt durchführbare Angriffe

ID: 3.01-33/9.0

Req 34	Für sicherheitsrelevante Protokollierungsdaten, die lokal erfasst werden, müssen geltende Aufbewahrungszeiträume und Löschfristen eingehalten werden.
--------	---

Aus Sicht der IT-Sicherheit wird auf einem System die lokale Erfassung von sicherheitsrelevanten Protokollierungsdaten nicht zwingend gefordert. Da lokal erfasste Protokollierungsdaten bei Systemstörungen beschädigt oder von einem erfolgreichen Angreifer gezielt manipuliert werden können, sind sie für sicherheitstechnische bzw. forensische Analysen nur bedingt belastbar. Entsprechend ist für die IT-Sicherheit relevant, dass Protokollierungsdaten an einen separaten Log Server weitergeleitet werden.

Eine lokale Erfassung kann dennoch erfolgen; beispielsweise falls bei der Erzeugung der Protokollierungsdaten eine lokale Speicherung aufgrund technischer Abläufe zunächst unabdingbar ist oder es begründete betriebliche Interessen gibt, Protokollierungsdaten auch lokal vorzuhalten.

Folgende Grundregelung ist bei einer lokalen Erfassung von Protokollierungsdaten zu berücksichtigen:

- Sicherheitsrelevante Protokollierungsdaten müssen für einen Zeitraum von 90 Tagen aufbewahrt werden.
(Diese Anforderung gilt nur, falls auf dem System keine zusätzliche Weiterleitung an einen separaten Log Server umgesetzt ist und Protokollierungsdaten daher ausschließlich lokal erfasst werden.)
- Nach Ablauf von 90 Tagen muss eine umgehende Löschung von gespeicherten Protokollierungsdaten erfolgen.

Abweichungen

Abweichende Aufbewahrungszeiträume und Löschfristen können durch gesetzliche oder regulatorische Anforderungen (insbesondere im Zusammenhang mit personenbezogenen Daten) bestehen oder durch vertragliche Vereinbarungen definiert sein. In diesen Fällen sind die geltenden Zeiträume individuell mit einem Project Security Manager (PSM) bzw. Datenschutzberater (DSB) abzustimmen oder werden durch diesen vorgegeben.

Motivation: Protokollierungsdaten sind ein immens wichtiges Werkzeug der IT-Sicherheit zur Abwendung, Detektierung und Aufklärung von Systemstörungen, Sicherheits- und Datenschutzvorfällen. Auf der anderen Seite unterliegt die Erfassung von Protokollierungsdaten wie jede andere Datenverarbeitung auch gesetzlichen und regulatorischen Auflagen. Entsprechend müssen Richtlinien eingehalten werden, die beides in Einklang bringen.

Umsetzungsbeispiel: Unter Berücksichtigung der gültigen Gesetzeslage und zutreffenden Datenschutz-Regularien werden folgende Löschfristen für lokal erfasste sicherheitsrelevante Protokollierungsdaten auf einem beispielhaften TK-System umgesetzt:

- Standard System Logs: Löschung nach spätestens 90 Tagen
- Protokollierungen von öffentlichen IP-Adressen: Löschung (oder Anonymisierung) nach spätestens 7 Tagen
- Protokollierungen der Vergabe von dynamischen öffentlichen IP-Adressen durch die TK-Lösung: Löschung nach spätestens 7 Tagen
- Protokollierungen nicht abrechnungsrelevanter Call Detail Records: Löschung nach spätestens 7 Tagen
- Protokollierungen der Kommunikations-Inhalte von E-Mail und SMS: Löschung nach spätestens 24 Stunden
- Protokollierungen der behandelten Domain-Abfragen des DNS-Servers der TK-Lösung: Löschung nach spätestens 24 Stunden

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigte Einsichtnahme oder Abgriff von Daten
- Abstreiten von durchgeführten Aktionen
- Unbemerkt durchführbare Angriffe

ID: 3.01-34/9.0

Req 35 Sicherheitsrelevante Protokollierungsdaten müssen umgehend nach deren Erzeugung an einen Log-Server weitergeleitet werden.

Die sicherheitsrelevanten Protokollierungsdaten des Systems müssen umgehend nach deren Erzeugung an einen Log-Server weitergeleitet werden.

Im Sinne der Anforderung muss der dafür genutzte Log-Server die folgenden Kriterien erfüllen:

- Der Log-Server muss dediziert für die Vorhaltung von Protokollierungsdaten eingesetzt sein. Er darf keine weiteren Funktionen oder Dienste abbilden, die nicht in Relation zu dieser Aufgabe stehen. Eine zulässige Ausnahme ist die Kombination eines Log Servers mit einer SIEM (Security Information and Event Management) Lösung.
- Der Log-Server muss vom System separiert sein. Er darf nicht als zusätzliches Modul unmittelbar mit auf dem System betrieben werden.
- Der Log-Server muss zentral für die Einsatzumgebung bereitstehen, zu der auch das System gehört.

Zur Übertragung der Protokollierungsdaten an den Log-Server sollten bevorzugt auf diesen Zweck spezialisierte Standard-Protokolle (z. B. Syslog oder SNMPv3) zum Einsatz kommen.

Motivation: Werden Protokollierungsdaten nur lokal gespeichert, können diese im Fall einer erfolgreichen Kompromittierung des Systems durch einen Angreifer manipuliert werden, um dessen Angriff und auf dem System durchgeführte Aktionen zu verschleiern. Aus diesem Grund muss die Weiterleitung von Protokollierungsdaten an einen Log-Server sehr zeitnah nach dem Auftreten eines sicherheitsrelevanten Ereignisses im Quell-System erfolgen. So stehen die angefallenen Protokollierungsdaten separiert vom Quell-System und unabhängig von dessen Zustand jederzeit für Analysen bereit. Die Verwendung eines zentralen Log-Servers für eine Einsatzumgebung stellt sicher, dass auf die Logs sämtlicher IT/NT-Systeme dieser Einsatzumgebung schnell und einheitlich zugegriffen und der Zugriff selbst optimal in die Incident-Prozesse der Umgebung integriert werden kann.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigte Modifikation von Daten
- Störung der Verfügbarkeit
- Abstreiten von durchgeführten Aktionen
- Unbemerkt durchführbare Angriffe

ID: 3.01-35/9.0

Req 36	Für sicherheitsrelevante Protokollierungsdaten, die an den Log Server weitergeleitet werden, muss die Einhaltung geltender Aufbewahrungszeiträume und Löschfristen sichergestellt werden.
--------	---

Folgende Grundregelung ist zu berücksichtigen:

- sicherheitsrelevante Protokollierungsdaten müssen auf dem Log Server für einen Zeitraum von 90 Tagen aufbewahrt werden.
- nach Ablauf von 90 Tagen muss eine umgehende Löschung der auf dem Log Server gespeicherten Protokollierungsdaten erfolgen.

Abweichungen

Abweichende Aufbewahrungszeiträume und Löschfristen können durch gesetzliche oder regulatorische Anforderungen bestehen (insbesondere im Zusammenhang mit personenbezogenen Daten) oder durch vertragliche Vereinbarungen definiert sein. In diesen Fällen sind die geltenden Zeiträume individuell mit einem Project Security Manager (PSM) bzw. Datenschutzberater (DSB) abzustimmen oder werden durch diesen vorgegeben.

Log Server in Verantwortung eines Dritten

Sollte der ausgewählte Log Server nicht innerhalb derselben operativen Verantwortung liegen wie das Quellsystem der Protokollierungsdaten, muss sichergestellt werden, dass dem verantwortlichen Betreiber des Log Servers die gültigen Parameter für die zu empfangenden Protokollierungsdaten bekannt sind und durch diesen im Sinne der hier genannten Regelungen eingehalten werden.

Motivation: Protokollierungsdaten sind ein immens wichtiges Werkzeug der IT-Sicherheit zur Abwendung, Detektierung und Aufklärung von Systemstörungen, Sicherheits- und Datenschutzvorfällen. Auf der anderen Seite unterliegt die Erfassung von Protokollierungsdaten wie jede andere Datenverarbeitung auch gesetzlichen und regulatorischen Auflagen. Entsprechend müssen Richtlinien eingehalten werden, die beides in Einklang bringen.

Umsetzungsbeispiel: Unter Berücksichtigung der gültigen Gesetzeslage und zutreffenden Datenschutz-Regularien werden folgende Löschfristen für ausgeleitete sicherheitsrelevante Protokollierungsdaten eines beispielhaften TK-Systems auf dem separaten Log Server umgesetzt:

- Standard System Logs: Löschung nach spätestens 90 Tagen
- Protokollierungen von öffentlichen IP-Adressen: Löschung (oder Anonymisierung) nach spätestens 7 Tagen
- Protokollierungen der Vergabe von dynamischen öffentlichen IP-Adressen durch die TK-Lösung: Löschung nach spätestens 7 Tagen
- Protokollierungen nicht abrechnungsrelevanter Call Detail Records: Löschung nach spätestens 7 Tagen
- Protokollierungen der Kommunikations-Inhalte von E-Mail und SMS: Löschung nach spätestens 24 Stunden
- Protokollierungen der behandelten Domain-Abfragen des DNS-Servers der TK-Lösung: Löschung spätestens nach 24 Stunden

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigte Einsichtnahme oder Abgriff von Daten
- Abstreiten von durchgeführten Aktionen
- Unbemerkt durchführbare Angriffe

ID: 3.01-36/9.0

Req 37	Sicherheitsrelevante Protokollierungsdaten müssen echtzeitnah an eine Erkennungs- und Reaktionslösung für Sicherheitsvorfälle bereitgestellt werden.
--------	--

Eine zulässige Erkennungs- und Reaktionslösung im Sinne dieser Anforderung muss die folgenden Kriterien erfüllen:

- Es erfolgt eine kontinuierliche Analyse bzw. Korrelation der aus angebundenen IT/NT-Systemen bereitgestellten Protokollierungsdaten.
- Die Identifizierung bedrohlicher Aktivitäten und laufender Angriffe findet auf Basis einer fortlaufend aktuell ge-

haltenen Erkennungsmuster-Datenbank statt.

- Integrierte Alarmierungsprozesse lösen im Fall eines Fundes die Einleitung von Gegenmaßnahmen aus.
- Sicherheitsexperten sind aktiv involviert in den Betrieb der Lösung sowie die Auswertung von Alarmen.

Eine solche Lösung ist in ihrer Funktion nicht mit einem Log Server gleichzusetzen. Entsprechend muss die Bereitstellung an eine Erkennungs- und Reaktionslösung stets zusätzlich zu einer vorhandenen Weiterleitung an einen Log Server erfolgen. Eine Ausnahme besteht, falls der genutzte Log Server seinerseits die Protokollierungsdaten umgehend nach Erhalt zusätzlich an eine Erkennungs- und Reaktionslösung bereitstellt. In diesem Fall ist es ausreichend, dass das System die Protokollierungsdaten nur an den Log Server weiterleitet.

Die Protokollierungsdaten müssen ab dem Zeitpunkt ihrer Erzeugung im System innerhalb von 5 Minuten an die Erkennungs- und Reaktionslösung bereitgestellt werden ("echtzeitnah").

Die konkret zu unterstützenden technischen Schnittstellen und Protokolle für die Bereitstellung werden durch die im Einsatzumfeld des Systems verfügbare Erkennungs- und Reaktionslösung sowie die technische Basis des Systems bestimmt.

Motivation: IT/NT-Systeme sind permanent sicherheitstechnischen Bedrohungen und Angriffen ausgesetzt. Mittels einer zentralen Analyse und der Korrelation erfasster System-Ereignisse ganzer IT-Landschaften in nahezu Echtzeit können aktuelle Bedrohungen und laufende Angriffe auch über Systemgrenzen hinaus erkannt und zeitnah passende Gegenmaßnahmen eingeleitet werden. Die unverzügerte Bereitstellung von Protokollierungsdaten an die Erkennungs- und Reaktionslösung ist essenziell für deren Effektivität.

Umsetzungsbeispiel: **Beispiel 1:** Ein System auf Basis eines klassischen Servers erfasst sicherheitsrelevante Protokollierungsdaten im lokalen Syslog oder Windows Event Log des Betriebssystems. Die Protokollierungsdaten werden kontinuierlich zur weiteren Analyse, Angriffserkennung und Alarmbehandlung an ein SIEM (Security Information and Event Management) weitergeleitet.

Beispiel 2: Auf einem System wird ein EDR ("Endpoint Detection and Response") Agent ausgeführt. Der EDR-Agent leitet die von ihm auf dem System erfassten sicherheitsrelevanten Protokollierungsdaten kontinuierlich an das zentrale Backend der EDR-Lösung aus, auf dem die weitere Analyse, Angriffserkennung und Alarmbehandlung erfolgt.

Beispiel 3: Netzwerk-Komponenten stellen Ereignisse an ein SIEM bereit, die es ermöglichen, verdächtiges Kommunikationsverhalten wie Netzwerk-Scans, unzulässige Datenflüsse zwischen Netz-Segmenten oder aktive Verbindungen bzw. Verbindungsversuche zu verdächtigen Zielen im Internet zu entdecken. Im speziellen liefern Gateway Router aus dem CNDTAG "Netflow" Logs, während Forward Proxies Zugriffe auf Domänen und/oder IP-Adressen melden und Firewalls blockierte Verbindungsversuche loggen.

Beispiel 4: Auf Basis von Cloud-nativen Funktionen entwickelte Systeme erzeugen Protokollierungsdaten mittels der Cloud-eigenen Protokollierungs- und Überwachungsinfrastruktur direkt aus diesen Funktionen heraus. In einem solchen System wird die Erzeugung sicherheitsrelevanter Protokollierungsdaten in den entsprechenden Konfigurationsoptionen der eingesetzten Cloud-nativen Funktionen gezielt aktiviert. Die entstehenden Protokollierungsdaten werden kontinuierlich zur weiteren Analyse, Angriffserkennung und Alarmbehandlung an ein SIEM (Security Information and Event Management) übertragen. Einige Cloud-Anbieter leisten eine gewisse Angriffsmustererkennung bereits in der Cloud-eigenen Überwachungsinfrastruktur und erzeugen daraus spezifische Cloud-Alarme. Diese werden zusätzlich an das SIEM übertragen.

Beispiel 5: Ein System basiert auf einer SaaS ("Software as a Service") Applikation eines Cloud-Anbieters. Typischerweise verantwortet im SaaS Modell der Cloud-Anbieter alle operativen Aspekte der Applikation, inkl. des Sicherheitsmanagements. Der Cloud-Kunde selbst hat keinen Einfluss auf die zugrunde liegende technische Infrastruktur oder den Betrieb der Applikation. Anhand von Vertragsvereinbarungen, anerkannten Zertifikaten und dokumentierten Selbstauskünften des Cloud-Anbieters wird formell überprüft, dass der Cloud-Anbieter eine Erkennungs- und Reaktionslösung für Sicherheitsvorfälle implementiert hat. Einige Cloud-Anbieter erzeugen zusätzlich dedizierte Protokollierungsdaten über sicherheitsrelevante Aktivitäten innerhalb der separaten Kunden-Mandanten ihrer SaaS Applikation (beispielsweise zu Änderungen von Konfigurationseinstellungen, Berechtigungsvergaben oder Benutzer-Konten innerhalb der Mandanten) und stellen diese den Cloud-Kunden zu Verfügung. Diese sogenannten Audit-Logs werden kontinuierlich zur weiteren Analyse, Angriffserkennung und Alarmbehandlung an ein SIEM (Security Information and Event Management) übertragen.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unbemerkt durchführbare Angriffe

