

Sicherheitsanforderung

CICD Ketten

Deutsche Telekom Gruppe

Version	1.3
Datum	01.12.2025
Status	Freigegeben

Impressum

Herausgeber
Deutsche Telekom AG
Vorstandsbereich Technology & Innovation
Chief Security Officer

Reuterstrasse 65, 53315 Bonn
Deutschland

Dateiname	Dokumentennummer 3.68	Dokumententyp Sicherheitsanforderung
Version 1.3	Stand 01.12.2025	Status Freigegeben
Fachlicher Ansprechpartner Telekom Security psa.telekom.de	Gültigkeitsdauer 01.12.2025 - 30.11.2030	Freigegeben von Stefan Pütz, Leiter SEC-T-TST

Zusammenfassung

This document covers security of CI/CD chains and secure usage of CI/CD chains for development, testing and production of systems which use CI/CD chains.

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.	CICD-Kettenanforderungen	5
2.1.	Allgemeine Anforderungen an die CICD-Kette	5
2.2.	Härtung	13
2.3.	Authentifizierung und Autorisierung	15
3.	Anforderungen an die Nutzer der CICD-Kette	18
3.1.	Allgemeine Anforderungen an die Nutzer der CICD-Kette	18
3.2.	Authentication and Authorization	23
4.	Nutzung der CICD-Kette durch Dritte	27
5.	Referenzen	28
6.	Glossar	29

1. Einleitung

Dieses Dokument behandelt die Sicherheitsanforderungen für:

- die CICD-Kette selbst (z. B. GitLab, Runner, Sicherheitsüberprüfung, Code Commits), in den meisten Fällen als "CICD-Kette" bezeichnet
- Systeme, die die CICD-Kette nutzen (Tenants der CICD-Kette und Nutzer der CICD-Kette; z. B. Anwendungen, die in GitLab gehostet werden, einschließlich ihrer jeweiligen Entwickler, Maintainer, usw.) für Entwicklungs-, Test- und Produktionszwecke, meist nur als Systeme bezeichnet, wobei der Systemverantwortliche die Person ist, die diesen Tenant innerhalb der CICD-Kette pflegt oder der Hauptansprechpartner für ihn ist

Der Begriff CICD (manchmal auch als CI/CD oder CI-CD geschrieben) hat die folgenden Bedeutungen:

- CI - fortlaufende Integration (Continuous Integration)
- CD - fortlaufende Bereitstellung und fortlaufende Verteilung (Continuous Delivery/Continuous Deployment)

CI/CD ist eine Kombination aus fortlaufender Integration und fortlaufender Bereitstellung, obwohl der Begriff in einigen Fällen auch die Verteilung einschließt, da die automatische Verteilung auf die fortlaufende Integration und Bereitstellung folgt. CICD wird in diesem Dokument als Abkürzung verwendet.

Die Abdeckung von Sicherheitsaspekten erstreckt sich über alle Schritte von CICD:

- Planung (und Dokumentation)
- **Code-Speicherung ***
- **Code-Qualitätskontrolle und Code-Bewertung ***
- **Erstellung ***
- **Test ***
- **Freigabe ***
- Bereitstellung
- Betrieb (und Überwachung)

Die ausgewählten (*) Schritte der CICD stehen in diesem Dokument besonders im Mittelpunkt, die übrigen werden zumindest teilweise abgedeckt, da für diese Themen bereits Sicherheitsanforderungen bestehen (z.B. gelten die Anforderungen an die Betriebssicherheit weiterhin).

Sichere Entwicklung ist eine wichtige Vorgehensweise. Schließlich sind Sicherheitsprobleme das Ergebnis unvorhergesehener Umstände und ganz einfach - Fehler, wie nicht initialisierter Speicher, unzulässige Speicheroperationen (Pufferunterlauf/-überlauf), Nicht-Bereinigung von Eingaben, fest kodierte Anmeldeinformationen, Code-/Befehlsinjektion, Konfigurationsfehler, usw. Die CICD-Kette in Kombination mit DevOps-Praktiken muss nicht nur eine schnelle Entwicklung, Bereitstellung und Verteilung ermöglichen, sie kann auch für eine sichere Entwicklung, Bereitstellung und Verteilung genutzt werden.

Terminologie

Die Begriffe CICD-Kette, CD-Werkzeugkette und CD-System werden austauschbar verwendet.

Auch die Begriffe Software, Systeme und Anwendungen werden in den meisten Fällen synonym verwendet. Sie sind die Kunden/Benutzer/Tenants der CICD-Kette.

Eine detaillierte Beschreibung der einzelnen Begriffe befindet sich im Glossar.

2. CICD-Kettenanforderungen

Dieses Kapitel enthält Sicherheitsanforderungen, die direkt auf die CICD-Kette angewendet werden; ergo diejenigen, für die die Betreiber der CICD-Kette verantwortlich sind.

2.1. Allgemeine Anforderungen an die CICD-Kette

Dieses Kapitel enthält Sicherheitsanforderungen, die sowohl für CICD-Ketten als auch für Systeme gelten, die CICD-Ketten verwenden. Die CICD-Kette muss die Funktionalität und vernünftige Standardwerte bereitstellen, aber die Systemverantwortlichen sind letztendlich für die korrekte und angemessene Verwendung der CICD-Kette verantwortlich.

Req 1	Die CICD-Kette muss völlig getrennt von Software und Systemen behandelt werden, die die CICD-Kette nutzen.
-------	--

Ein Stück Software oder ein System, das die CICD-Kette nutzt, kann nicht gleichzeitig Teil der CICD-Kette sein. Es ist wichtig, die Hosts, auf denen die CICD-Kette läuft, von der gesamten Software und dem System zu trennen, die in der CICD-Kette betrieben werden. Außerdem können interne Komponenten (und Netze) der CICD-Kette nicht von Software und Systemen genutzt werden, die in der CICD-Kette betrieben werden.

Die einzige Ausnahme kann die Verwendung der CICD-Kette zum Speichern und Testen bevorstehender Implementierungen der CICD-Kette selbst sein (zu beachten ist, dass die Implementierung der CICD-Kette durch die CICD-Kette ausgeschlossen werden muss, z. B. Denial-of-Service).

Motivation: Ein System mit privilegiertem Zugang zu den Interna der CICD-Kette kann sich leicht Zugang zu allen anderen in der CICD-Kette untergebrachten Systemen verschaffen.

Umsetzungsbeispiel:

- Die GitLab-Instanz kann zum Hosten der Software für die Bereitstellung (z. B. Ansible-Playbooks) von GitLab verwendet werden, aber ihre Pipelines/Runner können nicht für die (erneute) Bereitstellung verwendet werden. Dies muss vollständig separat erfolgen. Zusätzlich zu den typischen betrieblichen Problemen könnte dies zu einem Denial-of-Service führen.
- Auf die Interna der CICD-Kette (z. B. PostgreSQL von GitLab, Verwaltungsschnittstellen der GitLab-Runner, ...) können Anwendungen nicht zugreifen oder sie nutzen (auch nicht von ihren Aufträgen, die auf den Runnern ausgeführt 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
- Abstreiten von durchgeführten Aktionen
- Unbemerkt durchführbare Angriffe

ID: 3.68-1/1.3

Req 2	Die CICD-Kette muss mandantenfähig sein.
-------	--

Damit die CICD-Kette mehrere Software und Systeme (und natürlich auch Benutzer) hosten kann, muss sie die Mandantenfähigkeit in der gesamten Kette implementieren (nicht nur in einigen Komponenten wie dem Code-Repository). Die CICD-Kette muss eine vollständige Trennung zwischen einem gehosteten System und allen anderen gehosteten Systemen ermöglichen.

Motivation: Ohne eine ordnungsgemäße Trennung zwischen den in der CICD-Kette gehosteten Systemen kann ein gehostetes System Zugang zu einem anderen System erhalten, das in derselben CICD-Kette gehostet wird.

Umsetzungsbeispiel:

- GitLab bietet diese Möglichkeit, da jede Gruppe und jedes Projekt (Repository) vollständig von allen anderen

getrennt werden kann.

- Zusätzliche Software als Teil der CI/CD-Kette erfordert möglicherweise zusätzliche Integration und spezielle Konfiguration, um eine ordnungsgemäße Mandantenfähigkeit (Trennung) in der gesamten CI/CD-Kette zu erreichen.
- Einfache Git-Software, die auf Linux mit SSH-Zugang und einer einfachen webbasierten Benutzeroberfläche gehostet wird, erfüllt diese Anforderung nicht und kann im Allgemeinen nicht für das Hosting mehrerer Projekte verwendet werden
- Jenkins (+Runner) erlaubt standardmäßig den Zugriff auf andere Projekte - eine Trennung der Mandanten ist in diesem Fall nicht gegeben

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: 3.68-2/1.3

Req 3 Die CI/CD-Kette muss über ein Werkzeug zur Verwaltung von Geheimnissen verfügen.

Um den Einsatz von Produktionssystemen (und in vielen Fällen auch von Entwicklungs- und Testsystemen) zu unterstützen, enthält die Software, die entwickelt und eingesetzt wird, Geheimnisse (z. B. Zertifikate, Token, Schlüssel, Passwörter), die es ermöglichen, verschiedene Teile des Systems miteinander zu verbinden und in einigen Fällen auch den Einsatz in der Zielumgebung zu ermöglichen (z. B. Zugangsschlüssel für den Einsatz in der Cloud).

Da all diese Geheimnisse nicht im Klartext in den Repositories der CI/CD-Kette gespeichert werden dürfen, müssen sie entsprechend geschützt werden. Die CI/CD-Kette muss den Systemen, die sie nutzen, ausdrücklich mitteilen, welche Art von Geheimnisspeicher zur Verfügung steht und wofür er verwendet werden kann.

Motivation: Die Speicherung von Geheimnissen im Klartext zusammen mit dem übrigen Code (ohne Werkzeug zur Verwaltung von Geheimnissen) würde anderen Systemen einen extrem leichten Zugang ermöglichen.

Umsetzungsbeispiel:

- Geschützte Variablen von GitLab (allein) - ausreichend für die meisten Anwendungsfälle in der Entwicklung und in einigen Fällen für Testsysteme
- Geschützte GitLab-Variablen in Kombination mit einem geheimen Speicher (wie Vault) - ausreichend für Entwicklungs- und Testsysteme und in den meisten Fällen für Produktionssysteme (bevorzugte Lösung)
 - Das Szenario, das die Bereitstellung von Produktionssystemen ermöglichen würde, besteht in der Verwendung von geschützten GitLab-Variablen zur Speicherung der Anmeldeinformationen für den Vault-Zugang. Dann greifen die Bereitstellungssysteme (z. B. Runner), die die Anmeldeinformationen aus der Pipeline erhalten, auf den Vault zu (der über einen spezifischen Anmeldeinformations-Datenspeicher verfügt, der in Bezug auf IP-Adressen oder auf andere Weise eingeschränkt ist) und erhalten systemspezifische Anmeldeinformationen wie HTTPS-Zertifikate und Token oder Kennwörter für die Herstellung der Verbindung zu anderen Systemen.

Vault muss ordnungsgemäß gehärtet und konfiguriert sein, aber die Härtung von Vault ist nicht Gegenstand dieses Dokuments.

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
- Unbemerkt durchführbare Angriffe

- Angriffe motiviert und erleichtert durch Informationspreisgabe oder erkennbare Sicherheitsschwachstellen

ID: 3.68-3/1.3

Req 4 Die CI/CD-Kette muss auf Sicherheitsereignisse überwacht werden.

Da die CI/CD-Kette in der Produktion anderer Systeme verwendet wird, hängt die Sicherheit anderer Systeme ebenfalls stark von der Produktion der CI/CD-Kette ab. Daher ist es von äußerster Wichtigkeit, dass die CI/CD-Kette ordnungsgemäß überwacht wird und dass sicherheitsrelevante Ereignisse auf der gleichen Ebene wie die Ereignisse der Produktionssysteme verarbeitet werden.

Einige Beispiele für sicherheitsrelevante Ereignisse in einer CI/CD-Umgebung, die über die typischen Ereignisse hinausgehen, sind: Code Commit, Binär-Upload, Pipeline-Ausführung.

Motivation: Wenn ein CI/CD-System nicht auf Sicherheitsereignisse überwacht würde, könnte ein Angreifer die CI/CD-Kette als Ausgangspunkt für einen Angriff auf andere Produktionssysteme nutzen, da der Angriff nicht rechtzeitig oder überhaupt nicht erkannt würde.

Umsetzungsbeispiel: Die Sicherheitsereignisse sollten mit Zeitstempel und Verursacher protokolliert werden. Die Protokollierungsmöglichkeiten von GitLab sind unter <https://docs.gitlab.com/ee/administration/logs/> dokumentiert.

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
- Unbemerkt durchführbare Angriffe
- Angriffe motiviert und erleichtert durch Informationspreisgabe oder erkennbare Sicherheitsschwachstellen

ID: 3.68-4/1.3

Req 5 Die CI/CD-Kette muss die Speicherung von Artefakten getrennt vom Code ermöglichen.

Um Probleme zu vermeiden, die Binärdateien verursachen können, wenn sie in Code-Repositories gespeichert werden, muss die CI/CD-Kette über ein Artefakt-Repository verfügen (gehärtet und mandantenfähig entsprechend dem Rest der Umgebung), damit die Benutzer der CI/CD-Kette Artefakte auf angemessene Weise speichern können.

Eine Ausnahme von dieser Regel kann die Speicherung einiger Softwarekomponenten sein, die eng in die Kette integriert sind, wie z. B. Icons und Bilder, nicht aber Bibliotheken und andere Softwarepakete.

Motivation: Wenn Binärdateien im Code-Repository gespeichert werden, ist es weniger wahrscheinlich, dass sie als bösartig erkannt werden. Außerdem eignen sich Code-Repositorys nicht besonders für die Handhabung großer Dateien und haben möglicherweise noch keine Sicherheitslücken entdeckt und behoben.

Umsetzungsbeispiel: Beispiele für Artefakt-Repositorys:

- JFrog Artifactory
- GitLab Git Large File Storage (LFS) - zwar nicht sehr funktionsreich, aber für einige Anwendungsfälle ausreichend

Für diese Anforderung sind folgende Bedrohungen relevant:

- Störung der Verfügbarkeit
- Unbemerkt durchführbare Angriffe

ID: 3.68-5/1.3

Req 6	Die CICD-Kette muss Code und Artefakte regelmäßig auf Sicherheitsprobleme, im Klartext offengelegte Geheimnisse, bösartigen Code und Binärdateien überprüfen.
-------	---

Obwohl die beste Praxis natürlich darin besteht, beim Hochladen oder Übertragen zu scannen, ist dies nicht ausreichend, da Malware oder bösartiger Code in den ersten Tagen unentdeckt bleiben kann. Daher ist ein zusätzlicher regelmäßiger Scan erforderlich.

Zusätzlich zum herkömmlichen Scannen von Artefakten und Code auf Malware müssen Code und Artefakte auf Geheimnisse (Passwörter, Schlüssel, Token, etc.) gescannt werden, die versehentlich im Klartext hinterlassen wurden. Obwohl dieser Ansatz eine Menge falsch positiver Ergebnisse liefern kann, allein schon wegen der standardmäßigen Vorlagenkonfigurationsdateien, ist es nicht notwendig, dass die Ergebnisse zentral verarbeitet werden, sondern sie können an die Benutzer der CICD-Kette zurückgegeben werden, die die Verantwortung für die Behebung des Problems tragen würden.

Motivation: Bösartiger Code und Binärdateien können von böswilligen Akteuren in die CICD-Kette eingeschleust werden, um noch mehr Systeme zu infizieren. Um das Risiko der Infiltration anderer Systeme über die CICD-Kette zu minimieren, muss die CICD-Kette die Artefakte und den Code regelmäßig scannen.

Umsetzungsbeispiel: Als Beispiel für das Scannen von Artefakten kann jeder Antivirus- und Anti-Malware-Scanner verwendet werden, und wenn Artifactory verwendet wird, kann JFrog Xray eine gute Lösung sein. Für das Scannen von Code gibt es viele Tools für SAST und DAST (statisches und dynamisches Testen der Anwendungssicherheit) und sie hängen stark von den unterstützten Sprachen und APIs ab. SonarQube wird in der Regel häufig für SAST verwendet.

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
- Unbemerkt durchführbare Angriffe

ID: 3.68-6/1.3

Req 7	Die CICD-Kette muss eine angemessene, unveränderliche Protokollierung der Artefakte für die Nachverfolgung und Prüfung bereitstellen.
-------	---

Um Artefakte zu ihren Quellen und den Komponenten, aus denen sie bestehen, zurückverfolgen zu können, muss die CICD-Kette ausreichende Informationen über jedes Artefakt bereitstellen. Protokolle/Ausgaben von Build-Jobs, die neben dem Zeitstempel zumindest die entsprechende Commit-Kennung und Prüfsummen der abgerufenen Artefakte enthalten, können als absolutes Minimum angesehen werden.

Motivation: Ein Angreifer, der andere Systeme infiltrieren möchte, kann die CICD-Kette nutzen und Artefakte anschließend unbemerkt ändern.

Umsetzungsbeispiel: Einige Beispiele hierfür sind:

- Artefakte, die direkt in die Artefaktfabrik eingestellt wurden (Download/Upload), müssen bis zur Quelle zurückverfolgt werden können
- Artefakte, die sich auf Code-Commits beziehen, müssen direkt zurückverfolgt werden können zu:
 - dem Quellcode-Commit, der zu seiner Erstellung führte
 - andere Artefakte, die bei der Erstellung des endgültigen Artefakts verwendet wurden (z. B. Pakete, die aus anderen Quellen heruntergeladen wurden)

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

- Unbemerkt durchführbare Angriffe

ID: 3.68-7/1.3

Req 8	Die CICD-Kette muss den Code und die Artefakte gemäß den DTAG-Kryptographiestandards signieren.
-------	---

Die Code-Commits und die aus dem Code erstellten Artefakte müssen signiert werden, um die Integrität zu gewährleisten und die Authentizität zu verbessern oder zumindest das Risiko zu minimieren, dass Angreifer die CICD-Kette missbrauchen, um andere Systeme anzugreifen. Das CICD-System muss die aktuellen DTAG-Kryptographiestandards (3.50 - "Kryptographische Algorithmen und Sicherheitsprotokolle") für das Signieren von Code und Artefakten verwenden.

Motivation: Ein Angreifer, der in andere Systeme eindringen möchte, kann die CICD-Kette verwenden und anschließend unbemerkt Artefakte ändern.

Umsetzungsbeispiel: Eine Anleitung, wie man Commits in Gitlab signiert, findet sich unter https://docs.gitlab.com/ee/user/project/repository/gpg_signed_commits/.

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
- Unbemerkt durchführbare Angriffe

ID: 3.68-8/1.3

Req 9	Der Deployment-Pfad für ein Artefakt oder den Code muss zwingend die Überprüfung einer digitalen Signatur beinhalten, bevor das Artefakt oder der Code ausgeführt oder geöffnet wird.
-------	---

Die CICD-Kette ist in den meisten Fällen nicht direkt in den Bereitstellungspfad der in ihr gehosteten Systeme involviert, aber in den Fällen, in denen die CICD-Kette involviert ist, muss sie die digitale Signatur des Codes und der Artefakte verifizieren.

Systeme, die die CICD-Kette nutzen, müssen die Überprüfung der digitalen Signatur von Code und Artefakten, die Teil des Bereitstellungspfads (z. B. der Pipeline) sind, einschließen.

In einigen Fällen sind weder Code noch Artefakte ursprünglich signiert, müssen aber verwendet werden, weil sie z. B. aus einem Open-Source-Repository stammen. Dennoch ist es möglich, die folgenden Maßnahmen zu ergreifen:

- Überprüfung der Integrität des Hosts, der den Code oder das Artefakt hostet, durch Überprüfung des HTTPS-Zertifikats
- doppelte Überprüfung der URL auf Tippfehler (sowohl des Servers als auch des Repositorys selbst)
- Verwendung nur von Code-Commits, die im stabilen Zweig gelandet sind oder zumindest eine Art von Überprüfungsprozess durchlaufen haben
- Überprüfung der Prüfsummen der heruntergeladenen Artefakte

Motivation: Ein Angreifer, der andere Systeme infiltrieren möchte, kann die CICD-Kette verwenden und Artefakte anschließend unbemerkt ändern.

Umsetzungsbeispiel: Ein lokales Artefaktwerk kann verwendet werden, um Artefakte und Bilder von öffentlichen Repositories zu erhalten und zu speichern und während des Prozesses regelmäßige Scans durchzuführen.

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
- Unbemerkt durchführbare Angriffe

ID: 3.68-9/1.3

Req 10 Die CI/CD-Kette muss den Code vor Änderungen schützen, um einen Build aus dem Quellcode wiederherstellen zu können.

Normalerweise enthält die CI/CD-Kette ein Versionskontrollsystem. Jeder Code-Commit erhält in der Regel einen bestimmten Identifikator. Sobald ein Code mit einer bestimmten Kennung übertragen wird, muss das CI/CD-System Änderungen an der Übertragung verhindern, damit derselbe Code anschließend nicht zu einem anderen Build führt (externe Einflüsse ausgeschlossen, aber dafür trägt das System, das CI/CD verwendet, die Verantwortung, z. B. die Versionen von Paketen festzulegen).

Motivation: Ein Angreifer, der andere Systeme infiltrieren möchte, kann die CI/CD-Kette nutzen und den Code unbemerkt ändern.

Umsetzungsbeispiel: Normale Git-Software beinhaltet diese Funktionalität, z.B. beschreibt <https://git-scm.com/book/en/v2/Git-Basics-Tagging> eine mögliche Option.

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
- Unbemerkt durchführbare Angriffe

ID: 3.68-10/1.3

Req 11 Die CI/CD-Kette muss in der Lage sein, eine gegenseitige Überprüfung des Codes zu erzwingen, um die Codequalität sicherzustellen.

Da die CI/CD-Kette verwendet wird, um Systeme in die Produktion zu überführen, bei denen alle Änderungen von Kollegen geprüft werden müssen (auch bekannt als 4-Augen-Prinzip), muss die CI/CD-Kette in der Lage sein, solche Anwendungsfälle zu unterstützen. Dies kann durch die Implementierung einer Gating-Funktion als obligatorischer Bestandteil der CI/CD-Kette oder durch eine Kombination von Merge-Richtlinien und Rollen für die Benutzer der CI/CD-Kette erreicht werden.

Eine Ausnahme kann als Teil eines Notfallkontos/einer Notfallrolle vorkonfiguriert werden, wobei nur bestimmte Personen (also nicht das gesamte DevOps-Team) diese Funktion umgehen dürfen, um Notfallkorrekturen in der Produktionsumgebung durchzuführen. Solche Notfall-Commits müssen anschließend so schnell wie möglich überprüft werden.

Motivation: Eine einzelne Person, z. B. ein Entwickler in einer DevOps-Umgebung, kann sich versehentlich oder absichtlich Zugang zum Produktionssystem verschaffen und die Produktionsdaten offenlegen oder verändern.

Umsetzungsbeispiel: GitLab unterstützt diese Funktion durch Merge Requests und Freigaben.

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
- Unbemerkt durchführbare Angriffe

ID: 3.68-11/1.3

Req 12	Die CI/CD-Kette muss über Build-Server verfügen, die streng vom Code-Repository, Artefakt-Repository und anderen internen Teilen der CI/CD-Kette getrennt sind.
--------	---

Um zu verhindern, dass bösartige Software und Akteure die gesamte CI/CD-Kette kompromittieren, müssen die Build-Server (Runner) vollständig von allen internen Teilen der CI/CD-Kette (Datenbank, Messaging-Bus, ...) getrennt sein und dürfen auch keinen privilegierten Zugriff auf das Code-Repository oder das Artefakt-Repository haben. Insbesondere sollten sie nicht einmal in der Lage sein, interne Netzwerke der CI/CD-Kette zu erreichen.

Motivation: Build-Server können bösartigen Code oder Binärdateien herunterladen und ausführen.

Umsetzungsbeispiel: Build-Server müssen von allen anderen internen Komponenten der CI/CD-Kette getrennt sein und dürfen keinen privilegierten Zugang zu den Internen der CI/CD-Kette haben.

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
- Unbemerkt durchführbare Angriffe
- Angriffe motiviert und erleichtert durch Informationspreisgabe oder erkennbare Sicherheitsschwachstellen

ID: 3.68-12/1.3

Req 13	Die CI/CD-Kette muss über eine Möglichkeit verfügen, den Zugang zu Code- und Artefakt-Repositories zu beschränken.
--------	--

Die CI/CD-Kette muss ihren Nutzern die Möglichkeit geben, die Durchsetzung dieses Prinzips zu gewährleisten, indem sie dem Grundsatz "Need-to-know" folgt. In Bezug auf GitLab kann dies durch eine geeignete Kombination von Gruppen- und Projektsichtbarkeit sowie die Zuweisung von Benutzern zu bestimmten Rollen in Gruppen und Projekten erreicht werden. Die Standardeinstellung für neue Gruppen und Projekte muss, sofern dies möglich ist, auf "privat" gesetzt werden, um zu vermeiden, dass sensible Gruppen und Projekte versehentlich offengelegt werden.

Es muss jedoch betont werden, dass es sich hierbei nicht um eine Sicherheitsmaßnahme handelt, die sich gegen Open-Source-Software oder die gemeinsame Nutzung des Codes durch verschiedene Abteilungen/Projekte innerhalb der DTAG richtet. Die Idee ist nach wie vor, Open-Source-Software und die Zusammenarbeit zwischen verschiedenen Abteilungen und Projekten zu fördern, aber sicherzustellen, dass, wenn nötig, sensible Details immer noch vor einem breiteren Publikum verborgen werden können.

Motivation: Systemverantwortliche, die neue Gruppen und Projekte in der CI/CD-Kette erstellen, achten bei der Erstellung von Ressourcen möglicherweise nicht genau darauf und können versehentlich Geheimnisse preisgeben oder den Zugriff auf ihren Code oder ihre Systeme zu einem späteren Zeitpunkt ermöglichen.

Umsetzungsbeispiel: In GitLab muss die Standardeinstellung für Gruppen und Projekte auf privat gesetzt 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: 3.68-13/1.3

Req 14	Die CI/CD-Kette muss über eine Möglichkeit verfügen, bestimmte Zweige vor direkten Commits zu schützen.
--------	---

Einige Zweige, wie z. B. stabile Zweige, müssen vor der Durchführung von Tests vor direkten Commits geschützt werden. Außerdem muss der Hauptzweig vor direkten Commits geschützt werden, wenn die Praxis angewandt wird, je-

den Commit zwingend zu testen und zu prüfen und dann nach erfolgreichem Test und positiver Prüfung direkt in den Hauptzweig zu überführen. Daher muss die CI/CD-Kette es ermöglichen, dass die Systeme, die sie verwenden, konfigurieren können, in welche Zweige direkt übertragen werden darf und in welche nicht.

Motivation: Die CI/CD-Kette wird üblicherweise auch für die Produktionsbereitstellung verwendet, oder zumindest für die Erzeugung von Artefakten, die im Produktionssystem eingesetzt werden sollen. Direkte Übertragungen ohne Schutz könnten es einer einzelnen Person ermöglichen, unbeabsichtigt oder absichtlich bösartige Software auf dem Produktionssystem einzusetzen oder sich Zugang zu den Produktionssystemen zu verschaffen.

Umsetzungsbeispiel: In GitLab müssen der Hauptzweig und andere Zweige, die für die Produktionsbereitstellung verwendet werden, geschützt 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
- Störung der Verfügbarkeit
- Unbemerkt durchführbare Angriffe

ID: 3.68-14/1.3

Req 15	Die CI/CD-Kette muss die Ausgabe von Geheimnissen und anderen sensiblen Daten in Protokollen unterlassen.
--------	---

In Build- und Deployment-Protokollen (Pipeline-Protokollen) (oder Berichten) werden viele Informationen, einschließlich Debugging, gespeichert. Solche Protokolle enthalten Details über die Bereitstellung und sehr oft die Ausgabe von Umgebungsvariablen, die Geheimnisse enthalten (z. B. geschützte Variablen in GitLab). Um zu verhindern, dass sensible Daten und vor allem Geheimnisse nach außen dringen, dürfen die Protokolle und Ergebnisse von Builds (sowohl Artefakt- als auch Nicht-Artefakt-Ergebnisse, z. B. Softwarepakete und Metadaten über das Paket) keine Geheimnisse oder andere sensible Daten enthalten. GitLab tut dies in der Regel standardmäßig für geschützte Variablen, nicht aber für andere Geheimnisse. Daher müssen die Benutzer der CI/CD-Kette zumindest gewarnt werden, welche Geheimnisse und sensiblen Daten durch die CI/CD-Kette sicher (verschleiert/zerstört) sind und worauf die Benutzer der CI/CD-Kette achten müssen.

Motivation: Bereitstellungsprotokolle können Geheimnisse (z. B. Passwörter, Schlüssel, Zertifikate) für den Zugang zu den Entwicklungs-, Test- oder Produktionssystemen enthalten.

Umsetzungsbeispiel: Zusätzlich zu der obligatorischen Transparenz, die den Nutzern der CI/CD-Kette geboten wird, kann die CI/CD-Kette ein Scanning-Tool für Pipeline-Protokolle und Artefakte einführen (obwohl solche Lösungen nicht immer umfassend sind).

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigter Zugang zum System
- Unberechtigte Einsichtnahme oder Abgriff von Daten
- Unberechtigte Nutzung von Diensten oder Ressourcen
- Angriffe motiviert und erleichtert durch Informationspreisgabe oder erkennbare Sicherheitsschwachstellen

ID: 3.68-15/1.3

Req 16	Die CI/CD-Kette muss eine Möglichkeit zum Abrufen von Artefakten haben.
--------	---

Das Abrufprinzip (bei dem z. B. der Client der CI/CD-Kette die Verbindung initiiert und den Download durchführt) gilt gemeinhin als sicherer, daher muss die CI/CD-Kette über eine Funktion zum Abruf von Artefakten (z. B. Bilder, Binärpakete, Konfiguration usw.) verfügen. Das absolute Minimum ist die Möglichkeit, Artefakte über Pipelines und Build-Definitionen abzurufen. Darüber hinaus muss die CI/CD-Kette die Möglichkeit bieten, dass die Artefakte aus dem Artefakt-Repository von Zielsystemen abgerufen werden können (z. B. während der Verteilungsphase).

Motivation: Die Zielsysteme können die Kontrolle über die Verteilung behalten, und eine Kompromittierung der CI/CD-

Kette würde nicht automatisch eine Kompromittierung der Zielsysteme bedeuten.

Umsetzungsbeispiel: Typische Software wie GitLab bietet die Möglichkeit, dass die Zielsysteme Artefakte daraus abrufen (z. B. mit einfachen Tools wie `curl` oder `wget`).

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
- Unbemerkt durchführbare Angriffe

ID: 3.68-16/1.3

Req 17	Alle sensiblen Daten müssen über sichere Kanäle (verschlüsselt) übertragen werden. Bei Verwendung der Cloud muss die gesamte Kommunikation verschlüsselt sein.
--------	--

Die üblichen Anforderungen an das Datengeheimnis gelten für CI/CD-Ketten und Systeme, die CI/CD-Ketten verwenden, aber es ist notwendig, auf neue Bedrohungen hinzuweisen, die bei der Verwendung von CI/CD-Ketten auftreten. Die CI/CD-Kette enthält Daten zur Verwaltung von Produktionssystemen, daher muss ihr Sicherheitsniveau dem der Produktionssysteme entsprechen. Da die CI/CD-Kette aus mehreren Komponenten besteht, muss das Risiko des Abhörens und des Diebstahls von Geheimnissen minimiert werden, weshalb die interne Kommunikation innerhalb der CI/CD-Kette verschlüsselt werden muss (als absolutes Minimum muss die gesamte Kommunikation, die Daten für die Benutzer der CI/CD-Kette enthält oder mit ihnen in Verbindung steht, verschlüsselt werden). Alle externen Schnittstellen der CI/CD-Kette müssen im Allgemeinen eine Authentifizierung beinhalten, was wiederum eine verschlüsselte Kommunikation an ihren äußeren Grenzen erfordert.

Wenn die CI/CD-Kette in einer Cloud-Umgebung gehostet wird, ist es sehr wahrscheinlich, dass verschiedene Komponenten der CI/CD-Ketten auf verschiedenen Hosts, Datenzentren, Standorten/Regionen innerhalb der Cloud landen. Je nach ausgewähltem Cloud-Anbieter ist es möglich, dass die Kommunikation zwischen den Hosts oder den Datenzentren nicht zusätzlich geschützt ist. Daher ist es notwendig, alle Verbindungen zu verschlüsseln, die zwischen verschiedenen virtuellen Maschinen, verschiedenen Containern (sofern sie nicht immer auf demselben Host gehostet werden) oder zwischen verschiedenen Cloud-Diensten bestehen.

Die Systeme, die die CI/CD-Kette nutzen, müssen eine verschlüsselte Kommunikation mit der CI/CD-Kette verwenden, da die Verschlüsselung in den meisten Fällen eine Verifizierung des Hosts (HTTPS-Zertifikat) beinhaltet.

Motivation: Da es sich bei einer CI/CD-Umgebung um eine mandantenfähige Umgebung handelt, sollte man davon ausgehen, dass an einigen Stellen der Kette, insbesondere bei gemeinsam genutzten Runnern, ein höheres Risiko für böswillige Akteure besteht. Durch die Verwendung von Verschlüsselung (und Authentifizierung) kann ein System, das eine CI/CD-Kette verwendet, auch das Risiko verringern, Opfer von böswilligen Akteuren zu werden (wenn es wirklich nichts anderes ist, kann es in einigen Fällen etwas schwieriger sein, Geheimnisse zu stehlen).

Umsetzungsbeispiel: Verwendung ausschließlich sicherer Protokolle wie TLS (HTTPS) und SSH.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigte Einsichtnahme oder Abgriff von Daten
- Angriffe motiviert und erleichtert durch Informationspreisgabe oder erkennbare Sicherheitsschwachstellen

ID: 3.68-17/1.3

2.2. Härtung

Req 18	Die CI/CD-Kette (alle ihre Komponenten) muss gehärtet sein.
--------	---

Zusätzlich zu den regulären Sicherheitsanforderungen und der Härtung, die durch andere PSA-Sicherheitsanforderungsdokumente bekannt sind (z. B. Linux, SSH, Datenbank, Webserver/Anwendungen/Dienste, Docker, Kubernetes), müssen auch die anderen Komponenten der CI/CD-Kette gehärtet werden, die nicht ausdrücklich durch diese Anforderungen abgedeckt sind.

Motivation: Ohne eine ordnungsgemäße Konfiguration, die eine Härtung beinhaltet, können die Komponenten der

CICD-Kette missbraucht werden.

Umsetzungsbeispiel: Dabei sollte folgender Ansatz verfolgt werden:

- Wenn es ein spezifisches PSA-Dokument gibt, das die Komponente der CICD-Kette abdeckt (z. B. PostgreSQL), sollte die Härtung unter Verwendung des spezifischen PSA-Anforderungsdokuments vorgenommen werden
- Wenn es kein spezifisches PSA-Dokument gibt, das den Typ der Komponente der CICD-Kette abdeckt, sollte die Härtung entsprechend erfolgen:
 - bewährten Sicherheitsmaßnahmen (z. B. CIS-Benchmarks)
 - Sicherheits-/Härtungsleitfaden des Herstellers/der Community der spezifischen Komponente der CICD-Kette

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: 3.68-18/1.3

Req 19	Die Build-Server (Runner, Executor, ...) müssen gehärtet werden, um echte Mandantenfähigkeit der CICD-Kette zu unterstützen.
--------	--

Zusätzlich zu den allgemeinen Härtungsanforderungen ist es wichtig, die Notwendigkeit der Härtung von Runnern zu betonen. Runner werden manchmal vergessen, aber sehr oft sind sie das schwächste Glied, weil ihre Härtung bestimmte Anwendungsfälle unterbrechen kann. Die Verwendung von Runnern, die nicht ordnungsgemäß gehärtet sind, unterbricht die Mandantenfähigkeit, bei der ein Benutzer der CICD-Kette die Kontrolle über den gesamten Runner übernehmen und entweder Geheimnisse von anderen Benutzern stehlen oder sie verwenden kann, um in deren Produktionssysteme einzudringen.

Motivation: Build-Server können böartigen Code oder Binärdateien herunterladen und ausführen.

Umsetzungsbeispiel: Einige Beispiele für Runner:

- GitLab-Shell-Executors - nicht für Multitenancy geeignet
- GitLab-Docker-Runner - in der Regel nicht für die Mandantenfähigkeit geeignet
- GitLab mit Kubernetes - im Allgemeinen für Mandantenfähigkeit geeignet, aber abhängig von der Kubernetes-Konfiguration und dem verwendeten SDN (Software-defined Networking)

Mögliche Abhilfemaßnahmen für Fälle, in denen noch nicht vollständig gesicherte Runner vorhanden sind:

- Verwendung von Single-Use-Runnern - ein Runner wird erstellt, der Job (die Pipeline) wird ausgeführt und der Runner wird anschließend zerstört (außerdem muss für den Runner eine maximale Lebensdauer definiert werden)
- Benutzer der CICD-Kette können ihre eigenen Runner mitbringen (in der GitLab-Terminologie auch als gruppen- oder projektspezifische Runner bezeichnet)
 - Die Benutzer der CICD-Kette können weiterhin gemeinsam genutzte Runner für nicht sensible Aufgaben verwenden.
 - Die Benutzer der CICD-Kette müssen ihre eigenen Runner für sensible Aufgaben verwenden, z. B. für die Bereitstellung in der 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
- Unbemerkt durchführbare Angriffe
- Angriffe motiviert und erleichtert durch Informationspreisgabe oder erkennbare Sicherheitsschwachstellen

ID: 3.68-19/1.3

Req 20	Die der CICD-Kette hinzugefügten Build-Server (Runner, Executors, ...) müssen gehärtet sein.
--------	--

Bestimmte Benutzer der CICD-Kette bevorzugen oder verlangen die Verwendung von gruppen- oder projektspezifischen Runnern (auch bekannt als "bring your own runner"). In solchen Fällen müssen die Runner gemäß den regulären Anforderungen für CICD-Ketten-Runner gehärtet werden (siehe die vorherige Anforderung zu Build-Servern).

Motivation: Build-Server können böartigen Code oder Binärdateien herunterladen und ausführen.

Umsetzungsbeispiel: Einige Beispiele für Runner:

- GitLab-Shell-Executors - nicht für Mandantenfähigkeit geeignet
- GitLab-Docker-Runner - generell nicht für Mandantenfähigkeit geeignet
- GitLab mit Kubernetes - im Allgemeinen für Mandantenfähigkeit geeignet, aber abhängig von der Kubernetes-Konfiguration und dem verwendeten SDN (Software-defined Networking)

Mögliche Abhilfe für Fälle, in denen es noch nicht vollständig gesicherte Runner gibt (und sie nicht projekt- und umgebungsspezifisch sind, d. h. sie müssen mandantenfähig sein):

- Verwendung von Single-Use-Runnern - ein Runner wird erstellt, der Job (die Pipeline) wird ausgeführt und der Runner wird anschließend zerstört (außerdem muss für den Runner eine maximale Lebensdauer definiert 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
- Störung der Verfügbarkeit
- Unbemerkt durchführbare Angriffe
- Angriffe motiviert und erleichtert durch Informationspreisgabe oder erkennbare Sicherheitsschwachstellen

ID: 3.68-20/1.3

2.3. Authentifizierung und Autorisierung

Req 21	Die CICD-Kette muss eine Multifaktor-Authentifizierung (MFA) für Benutzer verwenden, zumindest für Benutzer mit der Rolle Approver oder höher.
--------	--

Die übliche Methode für die Übermittlung von Änderungen über SSH-Schlüssel kann zugelassen werden, aber die Commits müssen von einem MFA-authentifizierten Benutzer überprüft/genehmigt werden.

MFA ist zumindest für Benutzer mit der Rolle "Approver" oder höher erforderlich. Ein genehmigender Benutzer ist berechtigt, Artefakte in der Produktion zu verteilen. Daher muss die Berechtigung zum Genehmigen den Benutzern mit Sorgfalt erteilt werden. In Gitlab entspricht die Approver-Rolle zumindest in der kostenlosen Version der Entwicklerrolle (während die kostenpflichtigen Versionen mehr Flexibilität bieten, um zu definieren, wer genehmigen darf).

Motivation: Der Zugang zur Produktion anderer Systeme erfordert in der Regel eine 2-Faktor-Authentifizierung (2FA) oder MFA im Allgemeinen. Da die CICD-Kette auch für die Bereitstellung von Produktionssystemen verwendet wird, erfordert die Authentifizierung für die CICD-Kette die gleiche Art der Authentifizierung. Darüber hinaus trägt die obligatorische Aktivierung von MFA für alle Nutzer der CICD-Kette auch dazu bei, bestimmte Arten von Angriffen auf die Lieferkette zu entschärfen.

Umsetzungsbeispiel: Verwendung von Benutzernamen und Passwort in Kombination mit TOTP (zeitbasiertes Einmalpasswort).

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigter Zugang zum System
- Unbemerkt durchführbare Angriffe

ID: 3.68-21/1.3

Req 22	Die CICD-Kette muss für den M2M-Zugang eine starke Authentifizierung verwenden (einschließlich einer maximalen Lebensdauer des Schlüsselsatzes).
--------	--

MFA für (menschliche) Nutzer wird als starke Authentifizierung bezeichnet, und für den M2M-Zugang ist eine gleichwertige Authentifizierung erforderlich. Einfache Token ohne bestimmte Gültigkeitsdauer werden unter keinen Umständen als ausreichend angesehen. Es wird eine Gültigkeitsdauer von 90 Tagen für M2M-Tokens empfohlen.

Motivation: Einfache Token sind nicht sicherer als komplexe Kennwörter, und ein durchgesickertes Token würde es einem Angreifer ermöglichen, unbemerkt Zugang zur CICD-Kette zu erhalten.

Umsetzungsbeispiel: Ein Beispiel für eine starke Authentifizierung für den M2M-Zugang könnte eine Kombination aus (allen 3 Punkten) sein:

- Client-Zertifikate oder Token mit relativ kurzer Gültigkeitsdauer (z. B. 90 Tage)
- Beschränkung auf Basis des Netzwerkzugangs wie IP-Adressen
- Verhinderung von Brute-Force-Angriffen

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
- Unbemerkt durchführbare Angriffe

ID: 3.68-22/1.3

Req 23	Die CICD-Kette muss Mittel bereitstellen, um die Bereitstellung ungeprüfter Artefakte auf dem Produktionssystem zu verhindern.
--------	--

Die CICD-Kette muss entweder über eine eingebaute Gating-Funktionalität verfügen oder zumindest den Aufbau geeigneter Pipelines ermöglichen, so dass die Bereitstellung auf dem Produktionssystem erst dann erfolgen kann, wenn ein Test bestanden wurde (natürlich einschließlich Sicherheitstests).

Wenn die Verantwortung nicht direkt von der CICD-Kette wahrgenommen wird, müssen die Nutzer der CICD-Kette angemessen darüber informiert werden, welche Schritte zu unternehmen sind, um diese Anforderung zu erfüllen.

Motivation: Die Umgehung der Tests würde die einfache Einführung von böartigem Code oder Software in die Produktionssysteme ermöglichen.

Umsetzungsbeispiel: Gerrit ist ein Beispiel für eine Software, die als Gating-Funktion dienen kann, und andererseits können auch GitLab-Pipelines entsprechend konfiguriert 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
- Unbemerkt durchführbare Angriffe

ID: 3.68-23/1.3

Req 24 Die CI/CD-Kette muss in der Lage sein, Build-Definitionen zu schützen.

Build-Definitionen, Pipelines für die Bereitstellung und ihre Abhängigkeiten werden häufig verwendet, um sicherzustellen, dass Artefakte ordnungsgemäß getestet werden, einschließlich Sicherheitstests wie Codeanalyse, Scannen auf Malware usw. In solchen Fällen müssen Build-Definitionen, Pipelines und ihre Abhängigkeiten geschützt werden, damit nicht jeder Entwickler sie außer Kraft setzen und die Sicherheitstests deaktivieren kann. In der Regel wird diese Aufgabe ausgewählten Repository-Maintainern (oder -Eigentümern) übertragen, die als einzige die Berechtigung haben, Build-Definitionen zu ändern. Die CI/CD-Kette muss die Einrichtung einer solchen Struktur ermöglichen, um sicherzustellen, dass Build-Definitionen, Pipeline-Definitionen und ihre Abhängigkeiten vor regulären Entwickler-Commits geschützt werden können.

Motivation: Die Änderung von Build-Definitionen wäre ein einfacher Weg, Sicherheits- und andere Tests zu deaktivieren und böartigen Code oder Software auf den Produktionssystemen einzusetzen.

Umsetzungsbeispiel: Die kostenlose Version von GitLab ermöglicht die Konfiguration geeigneter Rollen und Zugriffsrechte, wenn auch nicht umfassend. In den kostenpflichtigen Versionen können die Dateien, die Build-Definitionen enthalten, vor regulären Commits geschützt werden (d. h. Repository-Maintainer- oder Eigentümerrechte wären erforderlich).

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
- Unbemerkt durchführbare Angriffe

ID: 3.68-24/1.3

Req 25 Die CI/CD-Kette muss die Möglichkeit bieten, den Zugriff auf sensible Informationen für externe Entwickler/Reporter zu beschränken.

Wird eine CI/CD-Kette auch für das Deployment der Produktion einiger Systeme verwendet, können Informationen aus Deployment-Protokollen für die am Entwicklungsprozess beteiligten Personen sichtbar sein, da diese in der Regel Zugriff auf das spezifische Repository haben. In einigen Fällen dürfen die Deployment-Informationen jedoch nicht mit externen Entwicklern oder Reportern geteilt werden (typische Rolle in GitLab), daher ist es notwendig, dass die CI/CD-Kette eine Funktion bietet, die nur Personen den Zugriff auf bestimmte Deployment-Protokolle und -Artefakte erlaubt, die tatsächlich Zugriff haben.

Motivation: Die Verwendung bestimmter Deployment-Pipelines für die Produktionsbereitstellung kann sensible Informationen in den Protokollen offenbaren, die nicht für alle Entwickler sichtbar sein sollen.

Umsetzungsbeispiel: In GitLab gibt es in der Regel nicht viele Möglichkeiten, einen solchen Zugriff einzuschränken, aber zumindest kann der Zugriff auf Projekte/Gruppen teilweise eingeschränkt werden.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigte Einsichtnahme oder Abgriff von Daten
- Angriffe motiviert und erleichtert durch Informationspreisgabe oder erkennbare Sicherheitsschwachstellen

ID: 3.68-25/1.3

3. Anforderungen an die Nutzer der CI/CD-Kette

Dieses Kapitel behandelt die Sicherheitsanforderungen, die für die Benutzer der CI/CD-Kette gelten, die in den meisten Fällen als "Systeme" oder "Anwendungen" bezeichnet werden. Von solchen Systemen wird erwartet, dass sie Systemverantwortliche haben, d. h. Personen mit höheren Rechten als gewöhnliche Entwickler (in der GitLab-Sprache wären dies meist Repository-Besitzer oder Maintainer).

3.1. Allgemeine Anforderungen an die Nutzer der CI/CD-Kette

Dieses Kapitel enthält Sicherheitsanforderungen, die für Systeme gelten, die CI/CD-Ketten verwenden. Die CI/CD-Kette muss Funktionalität und vernünftige Standardwerte bereitstellen, aber die Systemverantwortlichen sind letztendlich für die korrekte und angemessene Verwendung der CI/CD-Kette verantwortlich.

Req 26	Das System darf Geheimnisse nicht in Code-Repositories oder Artefakten speichern. Das System muss für die Speicherung von Geheimnissen das von der CI/CD-Kette bereitgestellte Tool zur Verwaltung von Geheimnissen verwenden.
--------	--

Um den Einsatz von Produktionssystemen (und in vielen Fällen auch von Entwicklungs- und Testsystemen) zu unterstützen, enthält die Software, die entwickelt und eingesetzt wird, Geheimnisse (z. B. Zertifikate, Token, Schlüssel, Passwörter), die es ermöglichen, verschiedene Teile des Systems miteinander zu verbinden und in einigen Fällen auch den Einsatz in der Zielumgebung zu ermöglichen (z. B. Zugangsschlüssel für den Einsatz in der Cloud).

Die Speicherung von Geheimnissen im Klartext im Code-Repository ist eines der größten Probleme, das letztendlich zu unberechtigtem Zugriff führt. Daher verfügt das CI/CD-System in der Regel über ein Tool zur Verwaltung von Geheimnissen. Unabhängig davon, wie rudimentär das Tool zur Verwaltung von Geheimnissen ist, ist es immer besser als die Speicherung der Geheimnisse im Klartext. Der Systemverantwortliche muss prüfen, welches Geheimhaltungswerkzeug der CI/CD-Kette für welchen Zweck geeignet ist.

Motivation: Geheimnisse (z. B. Passwörter, Schlüssel, Zertifikate) könnten jedem, der Zugang zum Code-Repository hat, den Zugang zu den Entwicklungs-, Test- oder Produktionssystemen ermöglichen.

Umsetzungsbeispiel: Einige CI/CD-Ketten können sogar mehrere Optionen für die Verwaltung von Geheimnissen anbieten, z. B.:

- GitLab-geschützte Variablen (nur) - ausreichend für die meisten Anwendungsfälle in der Entwicklung und in einigen Fällen für Testsysteme
- Geschützte GitLab-Variablen in Kombination mit einem geheimen Speicher (wie HashiCorp Vault) - ausreichend für Entwicklungs- und Testsysteme und die meisten Fälle von Produktionssystemen
 - Das Szenario, das die Bereitstellung von Produktionssystemen ermöglichen würde, besteht in der Verwendung von geschützten GitLab-Variablen zur Speicherung der Anmeldeinformationen für den Vault-Zugang. Dann greifen die Bereitstellungssysteme (z. B. Runner), die die Anmeldeinformationen aus der Pipeline erhalten, auf den Vault zu (der über einen spezifischen Anmeldeinformations-Datenspeicher verfügt, der in Bezug auf IP-Adressen oder auf andere Weise eingeschränkt ist) und erhalten systemspezifische Anmeldeinformationen wie HTTPS-Zertifikate und Token oder Kennwörter für die Herstellung der Verbindung zu anderen Systemen.

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
- Unbemerkt durchführbare Angriffe
- Angriffe motiviert und erleichtert durch Informationspreisgabe oder erkennbare Sicherheitsschwachstellen

ID: 3.68-26/1.3

Req 27	Das System muss für verschiedene Umgebungen unterschiedliche Anmeldeinformationen verwenden.
--------	--

Das System, das die CI/CD-Kette verwendet, muss eine saubere Trennung zwischen Entwicklungs-, Test-, Vorproduktions- und Produktionsumgebung aufweisen. Um die korrekte Trennung der Umgebungen zu erreichen, müssen für jede Umgebung unterschiedliche Anmeldeinformationen verwendet werden, damit:

- versehentliche Einsätze in falschen Umgebungen nicht möglich sind
- ein Angreifer, der sich die Anmeldeinformationen für die Entwicklungs- oder Testumgebung verschafft hat, nicht in die Produktionsumgebung gelangen kann

Die traditionelle Forderung nach einer Trennung zwischen Entwicklungs-, Test- und Produktionsumgebungen erstreckt sich nun auch auf Pipelines oder in einigen sehr speziellen Fällen auf die Notwendigkeit, getrennte Code- oder Artefakt-Repositories zu haben.

Motivation: Die Verwendung der gleichen Anmeldedaten für z.B. das Entwicklungs- und das Produktionssystem würde einen unautorisierten Zugriff auf das Produktionssystem ermöglichen, wenn nicht alle Entwickler Zugriff auf das Produktionssystem haben dürfen.

Umsetzungsbeispiel: Unterschiedliche Anmeldedaten in diesem Sinne sind:

- Mieter/Projektname (gilt für Cloud-Umgebungen)
- Schlüssel
- Geheimnisse
- Passwörter
- Zertifikate

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
- Unbemerkt durchführbare Angriffe
- Angriffe motiviert und erleichtert durch Informationspreisgabe oder erkennbare Sicherheitsschwachstellen

ID: 3.68-27/1.3

Req 28	In den Fällen, in denen die Speicherung von Geheimnissen in Artefakten oder Konfigurationsdateien (in Repositories) nicht vermieden werden kann, müssen die Geheimnisse in Code- und Artefakt-Repositories der CI/CD-Kette verschlüsselt werden.
--------	--

Für einige sehr ungewöhnliche Anwendungsfälle, in denen die Verwendung des CI/CD-Ketten-Geheimnismanagement-Tools nicht möglich ist (z. B. muss eine bestimmte Konfigurationsdatei ein bestimmtes Feld enthalten, um den Test zu bestehen, oder ein Image muss eine bestimmte Datei enthalten), ist es zulässig, ein verschlüsseltes Geheimnis im Code- oder Artefakt-Repository zu speichern, während der Entschlüsselungsschlüssel weiterhin im Geheimnismanagement-Tool gespeichert werden muss.

Darüber hinaus müssen alle auf diese Weise verwendeten Geheimnisse regelmäßig ausgetauscht werden (z. B. 90 Tage für allgemeine Anwendungsfälle).

Motivation: Die Speicherung von Anmeldeinformationen im Klartext würde einen unbefugten Zugriff auf Produktionssysteme ermöglichen.

Umsetzungsbeispiel: Zu diesem Zweck könnten Ansible-Vault oder OpenSSL verwendet 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
- Unbemerkt durchführbare Angriffe

ID: 3.68-28/1.3

Req 29 Die Artefakte (Binärdateien) müssen im Artefakt-Repository und nicht im Code-Repository gespeichert werden.

Um Probleme zu vermeiden, die Binärdateien verursachen können, wenn sie in Code-Repositories gespeichert werden, müssen die Artefakte in einem geeigneten Artefakt-Repository gespeichert werden.

Eine Ausnahme von dieser Regel kann die Speicherung einiger Software-Komponenten sein, die eng in die Software integriert sind, wie z.B. Icons und Bilder, jedoch nicht Bibliotheken und andere Software-Pakete. Beispiele hierfür sind Python-Libs, Docker-Images, ISO-Images usw.

Motivation: Wenn Binärdateien im Code-Repository gespeichert sind, ist es weniger wahrscheinlich, dass sie als böse erkannt werden. Außerdem eignen sich Code-Repositorys nicht besonders für die Handhabung großer Dateien und haben möglicherweise noch keine Sicherheitslücken entdeckt und behoben.

Umsetzungsbeispiel: Beispiele für Artefakt-Repositories:

- JFrog Artifactory
- GitLab Git Large File Storage (LFS) - zwar nicht reich an Funktionen, aber für einige Anwendungsfälle ausreichend

Für diese Anforderung sind folgende Bedrohungen relevant:

- Störung der Verfügbarkeit
- Unbemerkt durchführbare Angriffe

ID: 3.68-29/1.3

Req 30 Die Systemverantwortlichen müssen bestimmte Zweige vor direkten Commits schützen.

Einige Zweige wie stabile Zweige müssen vor direkten Commits geschützt werden, bevor die Tests durchgeführt wurden. Außerdem muss der Hauptzweig vor direkten Commits geschützt werden, wenn alle Commits zwingend getestet und geprüft werden müssen und dann nach erfolgreichem Test und positiver Prüfung direkt in den Hauptzweig überführt werden. Es ist Sache des Systemverantwortlichen, den Schutz bestimmter Zweige zu implementieren (zu konfigurieren), je nach gewähltem Entwicklungsansatz, oder allgemeiner - der Verwendung der CICD-Kette.

Motivation: Die CICD-Kette wird üblicherweise auch für den Produktionseinsatz verwendet, oder zumindest für die Erzeugung von Artefakten, die im Produktionssystem eingesetzt werden sollen. Direkte Commits ohne Schutz könnten es einer einzelnen Person ermöglichen, unbeabsichtigt oder absichtlich böartige Software auf dem Produktionssystem einzusetzen oder sich Zugang zu den Produktionssystemen zu verschaffen.

Umsetzungsbeispiel: In GitLab müssen der Hauptzweig und andere Zweige, die für die Produktionsbereitstellung verwendet werden, geschützt 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
- Störung der Verfügbarkeit
- Unbemerkt durchführbare Angriffe

ID: 3.68-30/1.3

Req 31	Das System, das die CICD-Kette verwendet, muss einen Prozess für den Code-Peer-Review einrichten/konfigurieren, um die Codequalität sicherzustellen.
--------	--

Um die Codequalität und die Abwesenheit von Sicherheitsproblemen zu gewährleisten, muss das System, das die CICD-Kette verwendet, die verfügbaren Funktionen der CICD-Kette nutzen und das Peer-Review ordnungsgemäß konfigurieren. Je nach CICD-Kette können die folgenden Funktionen verwendet werden: Gating-Funktion, Merge-Richtlinien und Rollen für CICD-Ketten-Benutzer. Für die meisten (wenn nicht für alle) Systeme, die das CICD-System auch für die Bereitstellung in der Produktion verwenden, wäre ein Peer-Review (4-Augen-Prinzip) ohnehin obligatorisch gewesen.

Eine Ausnahme könnte im Rahmen eines Notfallkontos/einer Notfallrolle vorkonfiguriert werden, bei dem/der nur bestimmte Personen (d. h. nicht das gesamte DevOps-Team) dies umgehen dürfen, um Notfallkorrekturen an der Produktionsumgebung vorzunehmen. Solche Notfall-Commits müssen anschließend so schnell wie möglich überprüft werden.

Motivation: Eine einzelne Person, z. B. ein Entwickler in einer DevOps-Umgebung, kann sich versehentlich oder absichtlich Zugang zum Produktionssystem verschaffen und die Produktionsdaten offenlegen oder verändern.

Umsetzungsbeispiel: GitLab unterstützt diese Funktion durch Merge Requests und Genehmigungen.

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
- Unbemerkt durchführbare Angriffe

ID: 3.68-31/1.3

Req 32	Die Systemverantwortlichen müssen den Zugriff auf Code- und Artefakt-Repositories angemessen einschränken.
--------	--

Die Systemverantwortlichen müssen das Prinzip "need to know" befolgen und sicherstellen, dass die Repositories (mit Code oder Artefakten) nur für die erforderliche Zielgruppe zugänglich sind. In Bezug auf GitLab kann dies durch eine geeignete Kombination von Gruppen- und Projektsichtbarkeit sowie durch die Zuweisung von Benutzern zu bestimmten Rollen in Gruppen und Projekten erreicht werden.

Es muss jedoch betont werden, dass es sich hierbei nicht um eine Sicherheitsmaßnahme handelt, die sich gegen Open-Source-Software oder die gemeinsame Nutzung des Codes durch verschiedene Abteilungen/Projekte innerhalb der DTAG richtet. Die Idee ist nach wie vor, Open-Source-Software und die Zusammenarbeit zwischen verschiedenen Abteilungen und Projekten zu fördern, aber sicherzustellen, dass, wenn nötig, sensible Details immer noch vor einem breiteren Publikum verborgen werden können.

Motivation: Systemverantwortliche, die neue Gruppen und Projekte im CICD erstellen, können Geheimnisse preisgeben oder den Zugriff auf ihren Code oder ihre Systeme ermöglichen.

Umsetzungsbeispiel: Im Sinne von GitLab müssen alle Repositories, die sensible Informationen über ihre Systeme enthalten, privat sein, während die Einstellung für Kooperationsprojekte oder Open-Source-Projekte intern bzw. öffentlich sein 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

ID: 3.68-32/1.3

Req 33	Die Build-Definition des Systems muss so aufgebaut sein, dass die Ausgabe von Geheimnissen und anderen sensiblen Daten in Protokollen verhindert wird.
--------	--

Build- und Deployment-Protokolle (oder Berichte) sind ein Ort, an dem eine Vielzahl von Informationen, einschließlich Debugging, gespeichert werden. Solche Protokolle enthalten Details über die Bereitstellung und sehr oft die Ausgabe von Umgebungsvariablen, die Geheimnisse enthalten (z. B. geschützte Variablen in GitLab). Um zu verhindern, dass sensible Daten und vor allem Geheimnisse nach außen dringen, dürfen die Protokolle und Ergebnisse von Builds (sowohl Artefakte als auch Nicht-Artefakte, z. B. Softwarepakete und Metadaten über das Paket) keine Geheimnisse oder andere sensible Daten enthalten. Die Verantwortung liegt nicht nur auf der Seite der CI/CD-Kette, sondern auch die Entwickler und Systembesitzer müssen sicherstellen, dass ihre Build- (und Pipeline-) Definitionen keine Geheimnisse ausgeben.

Motivation: Bereitstellungsprotokolle können Geheimnisse (z. B. Passwörter, Schlüssel, Zertifikate) für den Zugang zu den Entwicklungs-, Test- oder Produktionssystemen enthalten.

Umsetzungsbeispiel: GitLab tut dies im Allgemeinen standardmäßig für geschützte Variablen für Standardanwendungsfälle, kümmert sich aber nicht um zusätzlichen Code (für Pipeline-Definitionen), der in der Verantwortung des im GitLab-Repository gehosteten Systems liegt.

Die zusätzlichen Werkzeuge, die helfen können, durchgesickerte Geheimnisse in Protokollen zu erkennen, sollten verwendet werden, wenn sie von der CI/CD-Kette bereitgestellt werden.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigter Zugang zum System
- Unberechtigte Einsichtnahme oder Abgriff von Daten
- Unberechtigte Nutzung von Diensten oder Ressourcen
- Angriffe motiviert und erleichtert durch Informationspreisgabe oder erkennbare Sicherheitsschwachstellen

ID: 3.68-33/1.3

Req 34	Das System, das die CI/CD-Kette verwendet, muss so konfiguriert werden, dass Qualitätsmaßnahmen zwingend Teil des Build-Prozesses sind und dass der Build-Prozess fehlschlägt, wenn eine oder mehrere der zwingend vorgeschriebenen Kontrollen versagen.
--------	--

Sicherheitsmaßnahmen (die in vielen Fällen an andere Qualitätsmaßnahmen gebunden sind) werden in den meisten Fällen als Tests in der Pipeline oder im Gating-System implementiert. Auch wenn einige Tests für die Funktionalität und Sicherheit des Systems nicht entscheidend sind, werden die (obligatorischen) Sicherheitsmaßnahmen als entscheidend angesehen. Wenn eine oder mehrere (obligatorische) Sicherheitsmaßnahmen versagen, muss die Erstellung oder Verteilung der Pipeline fehlschlagen.

Motivation: Die Umgehung von Sicherheitstests würde die einfache Bereitstellung von böartigem Code oder Software auf den Produktionssystemen ermöglichen.

Umsetzungsbeispiel: GitLab-Pipelines können auch so konfiguriert werden, dass keine Build-Artefakte erzeugt werden, wenn der Sicherheitstest fehlschlägt.

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
- Unbemerkt durchführbare Angriffe

ID: 3.68-34/1.3

Req 35	Das System, das die CI/CD-Kette verwendet, muss den Workflow so einrichten, dass alle Artefakte, die in der Produktion eingesetzt werden sollen, von einer autorisierten Person erstellt werden.
--------	--

Die CI/CD-Kette bietet endlose Automatisierungsmöglichkeiten, birgt aber auch ein großes Risiko. Um das Risiko eines Angriffs zu minimieren, muss das System, das in der CI/CD-Kette gehostet wird, einen Workflow einrichten, der nicht alles automatisch für die Produktion bereitstellt, auch nicht nach automatisierten Tests, sondern vielmehr eine autorisierte Person hat, die die endgültige Genehmigung und Bestätigung vornehmen kann, um die frisch erstellten Artefakte in die Produktion zu schieben.

Motivation: Ein Angreifer könnte eine kleine Schwachstelle ausnutzen, um automatisch Änderungen an der Produktionsumgebung vorzunehmen und einen DoS-Angriff zu starten, die Produktion zu infiltrieren usw.

Umsetzungsbeispiel: In GitLab kann dies durch die Kombination von Zugriffsrechten, Zusammenführungsrichtlinien und dem Schutz von Build-Definitionen erreicht 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
- Störung der Verfügbarkeit
- Unbemerkt durchführbare Angriffe
- Angriffe motiviert und erleichtert durch Informationspreisgabe oder erkennbare Sicherheitsschwachstellen

ID: 3.68-35/1.3

3.2. Authentication and Authorization

Req 36	Das System, das die CI/CD-Kette verwendet, muss eine starke Authentifizierung für den M2M-Zugang konfigurieren.
--------	---

Systeme, die die CI/CD-Kette verwenden, müssen eine starke M2M-Authentifizierung ordnungsgemäß verwenden. Einfache Token ohne spezifizierte Gültigkeitsdauer werden unter keinen Umständen als ausreichend angesehen.

Motivation: Einfache Token sind nicht sicherer als komplexe Kennwörter, und ein durchgesickertes Token würde es einem Angreifer ermöglichen, unbemerkt Zugang zum System zu erhalten.

Umsetzungsbeispiel: Das System muss alle folgenden Details entsprechend konfigurieren (Details hängen von den Fähigkeiten des CI/CD-Systems ab):

- Lebensdauer des Client-Zertifikats oder -Tokens (z. B. empfohlene Gültigkeitsdauer von 90 Tagen)
- Umfang und Berechtigungen für das M2M-Konto
- Netze, von denen aus der Zugriff erfolgen kann (z. B. durch Begrenzung von IP-Adressbereichen)

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
- Unbemerkt durchführbare Angriffe

ID: 3.68-36/1.3

Req 37	Das System darf keine ungetesteten Artefakte direkt in der Produktionsumgebung verteilen.
--------	---

Alle Artefakte, die in der Produktionsumgebung eingesetzt werden sollen, müssen vollständig getestet werden, nicht nur in Bezug auf die Funktionalität, sondern auch auf die Sicherheit. Die Tests sollten dem natürlichen Fluss der Bereitstellung folgen, z. B. Entwicklung -> Test -> Produktion. Es ist auch wichtig zu betonen, dass das System die Sicherheitstests (und Funktionstests) eines Artefakts, das als Teil der CI/CD-Kette konfiguriert ist (z. B. Anti-Malware-Scanning), nicht umgehen darf.

Motivation: Die Umgehung der Tests würde die einfache Bereitstellung von böartigem Code oder Software auf den Produktionssystemen ermöglichen.

Umsetzungsbeispiel: GitLab-Pipelines können auch so konfiguriert werden, dass sie keine Build-Artefakte erzeugen, wenn der Sicherheitstest des Artefakts fehlgeschlagen ist oder nicht durchgeführt wurde.

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
- Unbemerkt durchführbare Angriffe

ID: 3.68-37/1.3

Req 38	Das System darf nur vertrauenswürdige Build- und Test-Abhängigkeiten (Artefakte und Code) verwenden, um die Sicherheit des Builds zu gewährleisten.
--------	---

Der Build-Prozess sowie viele Testprozesse bestehen nicht nur aus dem Code und den Artefakten in den spezifischen Code- und Artefakt-Repositories, sondern auch aus Abhängigkeiten. Abhängigkeiten, die verwendet werden, müssen verifiziert werden.

Motivation: Die Umgehung der Tests von eingeschlossenem Code oder Software würde die einfache Bereitstellung von böartigem Code oder Software auf den Produktionssystemen ermöglichen.

Umsetzungsbeispiel: Die Vertrauenswürdigkeit/Glaubwürdigkeit der Abhängigkeiten muss durch einige der folgenden Mittel überprüft werden:

- Bevorzugung von Artefakten, die digital signiert sind (durch Überprüfung der digitalen Signatur)
- Bezug von Artefakten aus vertrauenswürdigen Repositories (Überprüfung des HTTPS-Zertifikats und doppelte Überprüfung der URL)
- Bevorzugung von signiertem Code
- Bevorzugung stabiler Zweige des Quellcodes anderer Komponenten

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
- Unbemerkt durchführbare Angriffe

ID: 3.68-38/1.3

Req 39	Das System muss sicherstellen, dass extern bereitgestellte Artefakte und Code stets auf Sicherheitsprobleme hin gescannt und vor der weiteren Verwendung überprüft werden.
--------	--

Zusätzlich zur Überprüfung von Artefakten und Abhängigkeiten (wie bereits erwähnt - Verwendung vertrauenswürdiger Abhängigkeiten) muss das System extern bereitgestellte Artefakte, wie z. B. solche, die von öffentlichen Internet-Repositories heruntergeladen oder von einem Anbieter bereitgestellt werden, besonders sorgfältig prüfen. Solche extern bezogenen Artefakte müssen auf Sicherheitsprobleme gescannt und überprüft werden, bevor sie verwendet werden. Die Überprüfung bedeutet NICHT, dass eine Person auf die Hexdump-Ausgabe des Artefakts starren sollte, sondern vielmehr, um:

- die digitale Signatur oder zumindest die Prüfsumme zu überprüfen, wenn die Signatur nicht verfügbar ist
- die Integrität des Repositories, in dem das Artefakt gehostet wird, zu überprüfen (beim Herunterladen)

Motivation: Die Umgehung der Prüfung des Codes oder der Software aus externen Quellen würde die einfache Bereitstellung von böartigem Code oder böartiger Software auf den Produktionssystemen ermöglichen.

Umsetzungsbeispiel: Die extern bereitgestellten Artefakte müssen mindestens auf bekannte Malware gescannt werden (Antimalware/Antivirens Scanner), und die digitale Signatur muss überprüft 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
- Unbemerkt durchführbare Angriffe

ID: 3.68-39/1.3

Req 40	Der Systemverantwortliche muss den Schreibzugriff auf die Build-Definitionen auf die Repository-Maintainer beschränken.
--------	---

Build-Definitionen, Pipelines für die Bereitstellung und ihre Abhängigkeiten werden häufig verwendet, um sicherzustellen, dass Artefakte ordnungsgemäß getestet werden, einschließlich Sicherheitstests wie Codeanalyse, Scannen auf Malware usw. In solchen Fällen müssen Build-Definitionen, Pipelines und ihre Abhängigkeiten geschützt werden, damit nicht jeder Entwickler sie außer Kraft setzen und die Sicherheitstests deaktivieren kann. In der Regel wird diese Aufgabe ausgewählten Repository-Maintainern (oder -Eigentümern) übertragen, die die einzigen Personen sind, die die Berechtigung haben, Build-Definitionen zu ändern. Der Eigentümer des Systems ist dafür verantwortlich, die CI/CD-Kette ordnungsgemäß zu konfigurieren, um den Schreibzugriff auf die Build-Definitionen und ihre Abhängigkeiten einzuschränken, damit sie nicht umgangen werden können.

Motivation: Die Änderung von Build-Definitionen wäre ein einfacher Weg, um Sicherheits- und andere Tests zu deaktivieren und böartige Code oder Software auf den Produktionssystemen einzusetzen.

Umsetzungsbeispiel: Die Einschränkung des Zugriffs auf Dateien im Repository für typische GitLab-Pipelines ist ein weit verbreiteter Ansatz, der jedoch nicht in allen Editionen von GitLab verfügbar ist.

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
- Unbemerkt durchführbare Angriffe

ID: 3.68-40/1.3

Req 41	Der Systemverantwortliche muss den Zugang zu sensiblen Informationen für externe Entwickler/Berichtersteller beschränken.
--------	---

Wenn eine CI/CD-Kette auch für das Deployment der Produktion einiger Systeme verwendet wird, können die Informationen aus den Deployment-Protokollen für die am Entwicklungsprozess beteiligten Personen sichtbar sein, da diese in der Regel Zugriff auf das spezifische Repository haben. In einigen Fällen dürfen die Bereitstellungsinformationen jedoch nicht an externe Entwickler oder Reporter (eine Rolle, die typischerweise in GitLab vorkommt) weitergegeben werden. Daher ist es notwendig, dass der Systemverantwortliche den Zugriff auf bestimmte Bereitstellungsprotokolle und Artefakte ordnungsgemäß konfiguriert, so dass sie nur denjenigen zur Verfügung stehen, die den Zugriff tatsächlich benötigen.

Zusätzlicher Hinweis: Wenn GitLab verwendet wird, gibt es in der Regel nicht viele Möglichkeiten, den Zugriff zu beschränken, aber zumindest kann der Projekt-/Gruppenzugriff teilweise eingeschränkt werden.

Motivation: Die Verwendung bestimmter Deployment-Pipelines für die Produktionsbereitstellung kann sensible Informationen in Protokollen offenlegen, die nicht für alle Entwickler oder Reporter sichtbar sein sollen.

Umsetzungsbeispiel: In GitLab gibt es in der Regel nicht viele Möglichkeiten, einen solchen Zugriff einzuschränken, aber zumindest kann der Projekt-/Gruppenzugriff teilweise eingeschränkt werden.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigte Einsichtnahme oder Abgriff von Daten
- Angriffe motiviert und erleichtert durch Informationspreisgabe oder erkennbare Sicherheitsschwachstellen

ID: 3.68-41/1.3

4. Nutzung der CICD-Kette durch Dritte

Die CICD-Kette kann zusätzlichen direkten oder indirekten Zugang zu den internen Entwicklungs-, Test- und Produktionssystemen der DTAG bieten. Dieses Kapitel befasst sich mit Aspekten des Zugriffs Dritter (z. B. externer Entwickler oder externer Betreiber) in Verbindung mit der CICD-Kette.

Req 42	Externe Entwickler, die Quellcode zur Verfügung stellen und Artefakte bereitstellen, sollten gemäß den DTAG-Sicherheitsanforderungen als externes Personal behandelt werden und dürfen nicht die Berechtigung haben, Pipelines für die Produktion auszulösen.
--------	---

Externe Entwickler, die zur CICD-Kette beitragen, werden als externes Personal betrachtet. Daher müssen alle geltenden DTAG-Sicherheitsanforderungen und -verfahren berücksichtigt werden. Ordnungsgemäße Genehmigungen und Audits in der CICD-Kette gewährleisten die Sichtbarkeit der Handlungen des externen Personals.

Wenn der Zugang von Dritten zum DTAG-Netz für externe Auftragnehmer erforderlich ist, sollten bestehende Zugangsplattformen und -lösungen für Dritte verwendet werden und der Anforderungskatalog für Fremdfirmen (3.11) muss erfüllt werden.

Die Erlaubnis, Pipelines zu Produktionssystemen auszulösen, darf nur internen Mitarbeitern erteilt werden.

Motivation: Die Einschränkung und Überwachung der Aktivitäten externer Entwickler in der CICD-Kette verhindert den unbefugten Zugriff auf interne Systeme.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigter Zugang zum System
- Unberechtigte Modifikation von Daten
- Abstreiten von durchgeführten Aktionen

ID: 3.68-42/1.3

Req 43	Extern bereitgestellter Quellcode und Artefakte müssen auf Schwachstellen gescannt und wie jede externe Ressource behandelt werden.
--------	---

Externe Artefakte können unerwünschte oder veraltete Abhängigkeiten enthalten. Aus diesem Grund ist eine Überprüfung zwingend erforderlich, um eine angemessene Qualität sicherzustellen.

ID: 3.68-43/1.3

5. Referenzen

CIS Benchmarks: <https://www.cisecurity.org/cis-benchmarks/>

6. Glossar

CI - Fortlaufende Integration (Continuous Integration)

Praxis der Zusammenführung aller Entwicklerarbeiten in einem gemeinsamen Code-Repository mit automatisierten und schnellen Builds, Selbsttests der Builds, um Integrationsfehler frühzeitig zu erkennen, Chaos in letzter Minute zu vermeiden, Disziplin durchzusetzen und sofortiges Feedback zu fördern. Es erfordert einen höheren Vorlaufaufwand für die Erstellung automatisierter Testsuiten und lässt sich nicht gut mit älterem Code/Software integrieren. Es kann durch Abruf, in regelmäßigen Abständen oder durch Schieben (zum Code-Repository) erfolgen.

CD - Fortlaufende Bereitstellung (Continuous Delivery)

Ansatz zur Bereitstellung von Software in kurzen Zyklen und zur Gewährleistung, dass die Software jederzeit zuverlässig freigegeben werden kann. Es wird durch eine Pipeline ermöglicht, die die Sichtbarkeit, das Feedback und die Fähigkeit zur kontinuierlichen Bereitstellung verbessert. Es ermöglicht eine schnellere Markteinführung, verbesserte Produktivität und Effizienz, erhöhte Zuverlässigkeit und Produktqualität. Es passt nicht zu allen Branchen (z. B. Medizin), kann aber auch problematisch sein, wenn keine Testautomatisierung vorhanden ist oder wenn es Unterschiede zwischen Entwicklungs-, Test- und Produktionsumgebungen gibt.

Nicht zu verwechseln mit DevOps - trotz der Überschneidungen hat DevOps einen breiteren Anwendungsbereich.

* CD - Fortlaufende Auslieferung (Continuous Deployment)

Ansatz zur häufigen Bereitstellung von Softwarefunktionen durch automatisierte Deployments. Es handelt sich NICHT um eine kontinuierliche Bereitstellung - es geht darum, Software, die [CI+]CD bestanden hat, in die Produktion zu überführen, d. h. sie bereitzustellen oder für Endbenutzer verfügbar zu machen.

* = alternative Verwendung von CD

CI/CD steht für Continuous Integration / Continuous Delivery (obwohl manchmal auch Deployment gemeint ist)

CI/CD ist eine Kombination aus fortlaufender Integration und fortlaufender Bereitstellung, obwohl der Begriff in einigen Fällen auch die Auslieferung einschließt, da die automatische Auslieferung auf die fortlaufende Integration und Bereitstellung folgt.

Eine CI/CD-Pipeline, oder in den meisten Fällen einfach nur eine Pipeline, stellt eine Reihe von Schritten dar, die zusammengeführt werden, um eine fortlaufende Integration und eine fortlaufende Bereitstellung und/oder Auslieferung zu ermöglichen. Pipelines enthalten in der Regel Aufträge, die definieren, was zu tun ist, und Phasen, die festlegen, wann die Aufträge ausgeführt werden sollen. Pipelines in GitLab werden als `".gitlab-ci.yml"`-Datei im Repository definiert.

Jobs sind elementare Einheiten, die von einer Ausführungs-Engine, meist Runnern genannt, ausgeführt werden. In der Praxis handelt es sich dabei in der Regel um (Shell-)Skripte, die in die Pipeline-Definition (`".gitlab-ci.yml"`) eingebettet sind, oder um eigenständige Skriptdateien, die in der Pipeline-Definition referenziert/aufgerufen werden.

Runner sind Systeme (oder Ausführungsmaschinen), die Aufträge in CI/CD-Pipelines ausführen. Sie sind in der Regel Headless-Systeme, die den Repository-Inhalt zusammen mit den Abhängigkeiten abrufen, eine Ausführungsumgebung erstellen (meist ein Container, kann aber auch eine VM oder eine andere virtuelle Shell-Umgebung sein) und die Jobs in der Reihenfolge ausführen, die in der Pipeline definiert wurde.

Artefakte sind das Ergebnis der (nicht) erfolgreichen Ausführung der Jobs in der Pipeline zusammen mit den Ausführungsprotokollen. Bei einem typischen Entwicklungsprojekt ist das Hauptartefakt eine Anwendung/Software/Container/VM, die Dokumentation wird jedoch in der Regel automatisch erstellt oder aktualisiert. Erfolgreiche Aufträge liefern in den meisten Fällen keine verwertbaren Artefakte, sondern Protokolle (und vielleicht Dumps), die den Entwicklern genügend Rückmeldung geben, damit Probleme behoben werden können.