OPS.1.1.8 Infrastructure as Code, Pipelines und DevOps-Prozesse

1. Beschreibung

1.1. Einleitung

Moderne IT-Betriebs- und Entwicklungsmodelle setzen zunehmend auf DevOps-Praktiken, um die Bereitstellung von Anwendungen und Diensten zu beschleunigen und zu automatisieren. Kernelemente dieser Vorgehensweise sind der Einsatz von CI/CD-Pipelines (Continuous Integration/Continuous Delivery), die Verwaltung von Infrastruktur als Code (Infrastructure as Code, IaC) und die konsequente Nutzung von Versionsverwaltungssystemen als zentrale Quelle der Wahrheit ("Single Source of Truth"). Durch diese Automatisierung und Kodifizierung von Betriebs- und Entwicklungsprozessen entstehen neue, komplexe Angriffsvektoren. Eine Kompromittierung der CI/CD-Pipeline kann Angreifern ermöglichen, Schadcode direkt in produktive Anwendungen einzuschleusen, Zugangsdaten zu entwenden oder die Infrastruktur zu manipulieren. Fehlerhafter oder unsicherer IaC kann zur automatisierten Bereitstellung verwundbarer Infrastrukturen in großem Maßstab führen. Dieser Baustein adressiert diese Risiken und stellt Anforderungen, um die Integrität, Vertraulichkeit und Verfügbarkeit in hochautomatisierten DevOps-Umgebungen sicherzustellen.

1.2. Zielsetzung

Dieser Baustein zeigt einen systematischen Weg auf, wie DevOps-Prozesse und die zugehörigen Werkzeugketten (Versionsverwaltung, CI/CD-Tools, IaC-Frameworks) abgesichert werden können. Ziel ist es, Sicherheitsmechanismen tief in den automatisierten Lebenszyklus von Software und Infrastruktur zu integrieren ("DevSecOps"), um Risiken frühzeitig zu erkennen, Missbrauch der Pipelines zu verhindern und die Konformität mit dem IT-Grundschutz zu gewährleisten, ohne die Agilität der Prozesse zu beeinträchtigen.

1.3. Abgrenzung und Modellierung

Der Baustein OPS.1.1.8 ist auf den Informationsverbund einmal anzuwenden, sofern DevOps-Prozesse, IaC oder CI/CD-Pipelines eingesetzt werden.
Dieser Baustein behandelt nicht die spezifischen Anforderungen an die sichere Softwareentwicklung, die im Baustein CON.1 Softwareentwicklung beschrieben sind. Er fokussiert auf die Absicherung der Prozesse und Werkzeuge, die den Code von der Entwicklung bis in den produktiven Betrieb überführen. Ebenso werden grundlegende Anforderungen an die Absicherung von Servern oder Cloud-Diensten in den jeweiligen Bausteinen (z. B. OPS.1.1.1 Server unter Linux/Unix oder OPS.2.3 Nutzung von Cloud-Diensten) behandelt.

2. Gefährdungslage

Für den Baustein OPS.1.1.8 sind folgende spezifische Bedrohungen und Schwachstellen von besonderer Bedeutung:

Kompromittierung der CI/CD-Pipeline

Angreifer verschaffen sich Zugriff auf die Konfiguration oder die Laufzeitumgebung der CI/CD-Pipeline (z. B. Jenkins, GitLab CI, GitHub Actions). Dadurch können sie Build-Prozesse manipulieren, um Schadcode einzuschleusen, Artefakte zu verändern oder Zugangsdaten, die in der Pipeline hinterlegt sind, zu stehlen.

Unsicherer oder fehlerhafter Infrastructure as Code (IaC)

IaC-Skripte (z. B. Terraform, Ansible, Bicep) enthalten Konfigurationsfehler, die zur Bereitstellung unsicherer Infrastruktur führen. Beispiele sind offene Firewall-Regeln, öffentlich zugängliche Speicherdienste oder die Verwendung schwacher kryptografischer Einstellungen. Da IaC skalierbar ist, können diese Fehler systemweit ausgerollt werden.

Kompromittierung von Zugangsdaten in Versionsverwaltung und Pipelines

Entwickler hinterlegen sensible Zugangsdaten wie Passwörter, API-Schlüssel oder private Zertifikate direkt im Code in Versionsverwaltungssystemen (z. B. Git). Diese sind dann für alle lesbar, die Zugriff auf das Repository haben, und können bei einer Kompromittierung leicht missbraucht werden.

Einschleusung von Schadcode über kompromittierte Abhängigkeiten

Die CI/CD-Pipeline lädt automatisch externe Softwarebibliotheken, Basis-Images für Container oder andere Abhängigkeiten aus öffentlichen oder internen Quellen. Sind diese Quellen kompromittiert oder enthalten die Abhängigkeiten selbst Schwachstellen (Supply Chain Attack), wird der Schadcode unbemerkt in die eigenen Anwendungen und Systeme integriert.

Unzureichende Kontrolle und Protokollierung automatisierter Prozesse

Aufgrund der hohen Geschwindigkeit und Automatisierung von DevOps-Prozessen fehlen ausreichende Kontroll- und Genehmigungsschritte. Änderungen an der Infrastruktur oder an Anwendungen werden ohne manuelle Prüfung produktiv gesetzt. Eine unzureichende Protokollierung erschwert die Nachvollziehbarkeit von Aktionen und die Aufklärung von Sicherheitsvorfällen.

Missbrauch durch unzureichende Funktionstrennung in DevOps-Prozessen

Mitarbeitende verfügen über weitreichende Berechtigungen, die es ihnen erlauben, Code zu schreiben, zu genehmigen und produktiv zu setzen. Das Fehlen des Vier-Augen-Prinzips oder einer klaren Funktionstrennung zwischen Entwicklung, Test und Betrieb erhöht das Risiko von unbeabsichtigten Fehlern oder vorsätzlichem Missbrauch.

3. Anforderungen

Im Folgenden sind die spezifischen Anforderungen des Bausteins OPS.1.1.8 aufgeführt.

3.1. Basis-Anforderungen

Die folgenden Anforderungen MÜSSEN für diesen Baustein vorrangig erfüllt werden.

OPS.1.1.8.A1 Richtlinie für sichere DevOps-Prozesse (B)

Es MUSS eine Richtlinie für die sichere Gestaltung und den sicheren Betrieb von CI/CD-Pipelines und DevOps-Prozessen existieren. Diese MUSS grundlegende Sicherheitsvorgaben, Verantwortlichkeiten und die zu verwendenden Werkzeuge und Verfahren festlegen.

OPS.1.1.8.A2 Absicherung der Versionsverwaltung (B)

Das zentrale Versionsverwaltungssystem (z. B. Git-Server) MUSS abgesichert sein. Es MUSS ein Berechtigungskonzept umgesetzt sein, das den Zugriff auf Code-Repositories regelt. Alle Änderungen MÜSSEN einer Person zugeordnet werden können.

OPS.1.1.8.A3 Grundlegender Schutz von Zugangsdaten (Secrets) (B)

Es MUSS verboten sein, Zugangsdaten (wie Passwörter oder API-Schlüssel) unverschlüsselt und in Klartext in Repositories der Versionsverwaltung zu speichern. Entwickler MÜSSEN dahingehend sensibilisiert werden.

OPS.1.1.8.A4 Rudimentäre Zugriffskontrolle für CI/CD-Werkzeuge (B)

Der Zugriff auf die Administrations- und Konfigurationsoberflächen der CI/CD-Werkzeuge (z. B. Jenkins) MUSS auf einen definierten Personenkreis beschränkt sein. Es MUSS geregelt sein, wer Pipelines ausführen und deren Konfiguration ändern darf.

OPS.1.1.8.A5 Protokollierung der Pipeline-Aktivitäten (B)

Alle sicherheitsrelevanten Aktivitäten innerhalb der CI/CD-Pipeline, wie das Starten eines Builds, Änderungen an der Pipeline-Konfiguration und Deployments, MÜSSEN protokolliert werden.

3.2. Standard-Anforderungen

Gemeinsam mit den Basis-Anforderungen entsprechen die folgenden Anforderungen dem Stand der Technik. Sie SOLLTEN grundsätzlich erfüllt werden.

OPS.1.1.8.A6 Einsatz eines zentralen Secrets Managements (S)

Zugangsdaten, die von der CI/CD-Pipeline benötigt werden, SOLLTEN zentral und sicher über ein dediziertes Secrets-Management-System (z. B. HashiCorp Vault, Azure Key Vault, AWS Secrets Manager) verwaltet werden. Die Pipeline SOLLTE sich zur Laufzeit bei diesem System authentisieren, um die benötigten Secrets abzurufen.

OPS.1.1.8.A7 Sicherung von Code-Änderungen durch Branch-Protection und Reviews (S)

Für produktiv relevante Branches (z. B. main, release) im Versionsverwaltungssystem SOLLTEN Schutzmechanismen (Branch Protection Rules) aktiviert werden. Änderungen SOLLTEN nur über Pull- oder Merge-Requests möglich sein, für die eine Genehmigung durch mindestens eine weitere Person (Vier-Augen-Prinzip) erforderlich ist.

OPS.1.1.8.A8 Integration von automatisierten Sicherheitsprüfungen (SAST/SCA) (S)

In die CI/CD-Pipeline SOLLTEN automatisierte Werkzeuge zur Sicherheitsanalyse integriert werden. Dies SOLLTE mindestens eine statische Code-Analyse (Static Application Security Testing, SAST) sowie eine Analyse der Abhängigkeiten auf bekannte Schwachstellen (Software Composition Analysis, SCA) umfassen. Builds SOLLTEN bei Funden von kritischen Schwachstellen fehlschlagen.

OPS.1.1.8.A9 Absicherung von Build-Artefakten und Container-Images (S)

Alle erzeugten Build-Artefakte (z. B. Binärdateien, Container-Images) SOLLTEN in einem gesicherten Artefakt-Repository gespeichert werden. Der Zugriff auf dieses Repository SOLLTE geregelt sein. Container-Images SOLLTEN regelmäßig auf bekannte Schwachstellen gescannt werden.

OPS.1.1.8.A10 Statische Analyse von Infrastructure as Code (IaC) (S)

Infrastructure as Code (IaC) SOLLTE vor der Anwendung durch spezialisierte Werkzeuge statisch auf Sicherheitsfehlkonfigurationen und die Einhaltung von internen Richtlinien (Policy as Code) überprüft werden.

OPS.1.1.8.A11 Detailliertes Rollen- und Rechtekonzept für die CI/CD-Pipeline (S)

Es SOLLTE ein detailliertes Rollen- und Rechtekonzept für die CI/CD-Pipeline nach dem Prinzip der geringsten Rechte (Least Privilege) umgesetzt werden. Es SOLLTE klar getrennt sein, wer Code schreiben, Builds genehmigen und Deployments in verschiedene Umgebungen (Test, Produktion) durchführen darf.

3.3. Anforderungen bei erhöhtem Schutzbedarf

Die folgenden Anforderungen sind exemplarische Vorschläge für ein Schutzniveau, das über den Stand der Technik hinausgeht. Sie SOLLTEN bei erhöhtem Schutzbedarf in Betracht gezogen werden.

OPS.1.1.8.A12 Härtung und Isolation der Pipeline-Umgebung (H)

Die Laufzeitumgebung der CI/CD-Pipeline (Build-Agents) SOLLTE gehärtet und isoliert sein. Build-Jobs SOLLTEN in ephemeren, containerisierten Umgebungen ausgeführt werden, die nach jeder Ausführung zerstört werden, um Persistenz von Angreifern zu verhindern. Der Netzwerkzugriff der Build-Agents auf interne Ressourcen und das Internet SOLLTE restriktiv geregelt sein.

OPS.1.1.8.A13 Erstellung und Verifizierung von Software-Stücklisten (SBOM) (H)

Für jedes Build-Artefakt SOLLTE automatisch eine maschinenlesbare Software-Stückliste (Software Bill of Materials, SBOM) in einem Standardformat (z. B. CycloneDX, SPDX) erstellt werden. Diese SOLLTE archiviert werden, um die Transparenz über alle verwendeten Komponenten zu gewährleisten und bei neuen Schwachstellenmeldungen schnell reagieren zu können.

OPS.1.1.8.A14 Integration fortgeschrittener Sicherheitsanalysen (DAST/IAST) (H)

Zusätzlich zu SAST und SCA SOLLTEN fortgeschrittene dynamische Sicherheitsanalysen (Dynamic Application Security Testing, DAST) und interaktive Analysen (Interactive Application Security Testing, IAST) in automatisierten Testumgebungen als Teil der Pipeline durchgeführt werden.

OPS.1.1.8.A15 Kryptografische Signierung von Code-Commits und Artefakten (H)

Alle Commits im Versionsverwaltungssystem SOLLTEN von den Entwicklern kryptografisch signiert werden (z. B. mit GPG), um ihre Authentizität und Integrität sicherzustellen. Alle erzeugten Build-Artefakte SOLLTEN ebenfalls signiert werden, und die Signatur SOLLTE vor dem Deployment verifiziert werden.

OPS.1.1.8.A16 Umsetzung des Prinzips der unveränderlichen Infrastruktur (Immutable Infrastructure) (H)

Die durch IaC bereitgestellte Infrastruktur SOLLTE als unveränderlich behandelt werden. Anstatt manuelle Änderungen an laufenden Systemen vorzunehmen, SOLLTEN alle Anpassungen im IaC-Code erfolgen. Anschließend wird die bestehende Infrastruktur durch eine neue, aus dem aktualisierten Code generierte Version ersetzt. Dies verhindert Konfigurationsdrift und stellt die Nachvollziehbarkeit sicher.