/informatik & security /fh/// st.pölten

Sicherheitsrisiken und Gegenmaßnahmen für DevOps Umgebungen

Dominik Sramec | IT-SECX 2018

dominik.sramec@outlook.com

Agenda

/informatik & security /fh/// st.pölten

- Ziel
- DevOps
- DevSecOps
- Analyse einer Deployment Pipeline
  - Tools und umgebungsunabhängige Gegenmaßnahmen

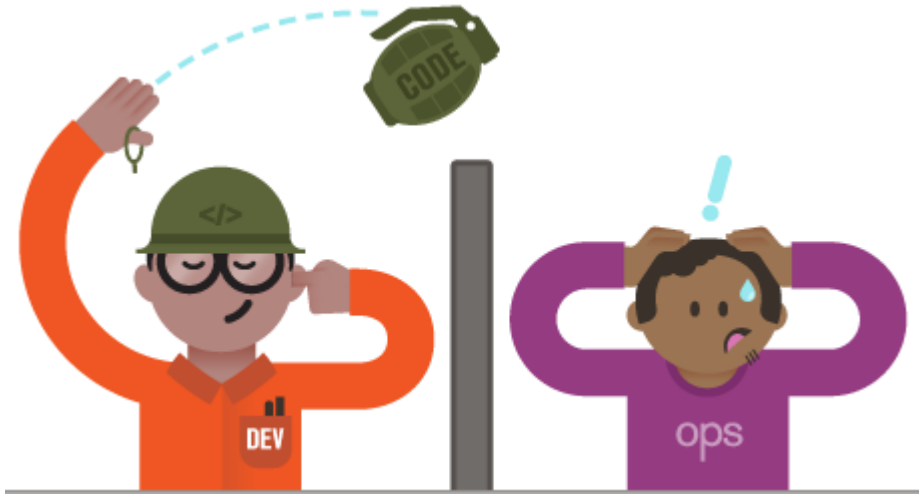
Ziel

- Welche Bedrohungen gibt es in einer DevOps-Umgebung?
- Welche Angriffsvektoren und -möglichkeiten ergeben sich für einen Angreifer in einer DevOps Umgebung?
- Welche Gegenmaßnahmen müssen vorgenommen werden um diese Bedrohungen zu entkräften?

DevOps – Definition (1)

- Kombination aus Denkweisen, Praktiken und Tools um Services und Applikationen schneller ausliefern zu können
  - Dev vs Ops
    - Features regelmäßig und schnell deployen

    vs.
    - Services stabil und zuverlässig halten



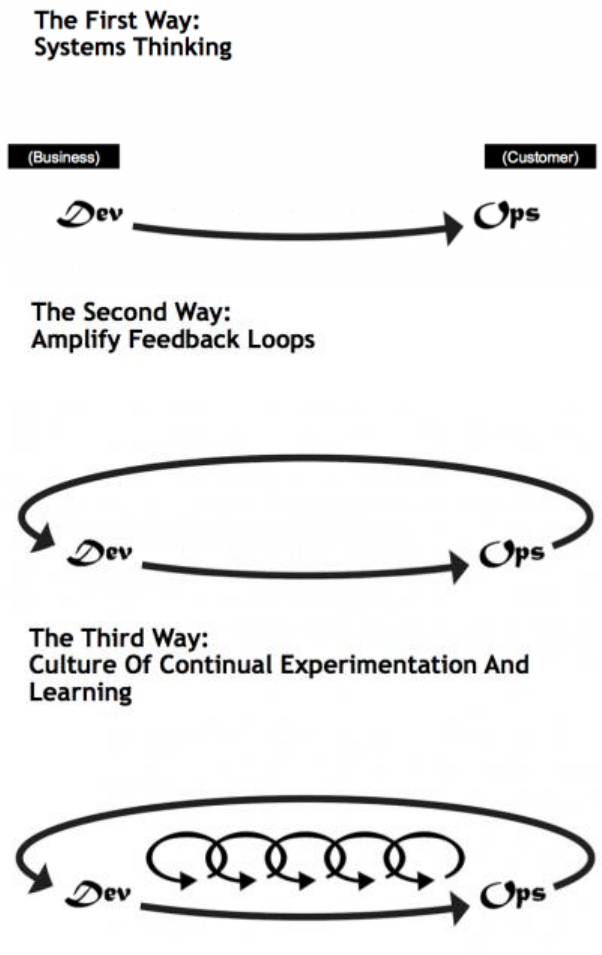
https://insights.thirdrepublic.com/wp-content/uploads/2018/09/devops-success.png

DevOps – Definition (2)

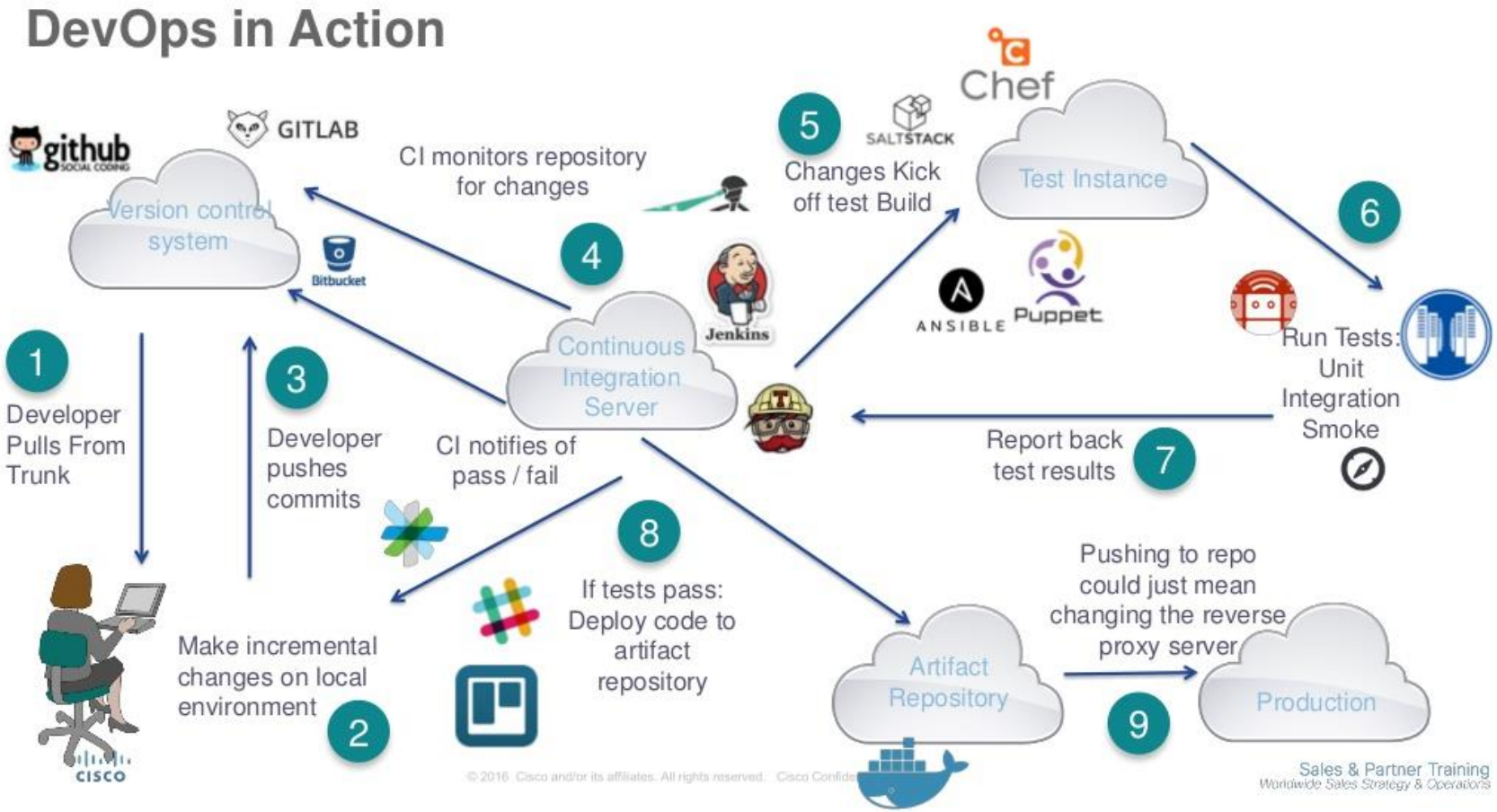
- Weitere Ziele
  - Erhöhung und Verbesserung der Zusammenarbeit von „Dev“ und „Ops“?
    - Agile Manifesto
    - Continuous Integration und Continuous Delivery
    - Toyota Kata
    - Teilen von gemeinsamen Tools
    - …
  - Reduzierung von möglichen Fehlkonfigurationen (Immutable Infrastructure)
  - Prozess von der Entwicklung bis zur Produktion beschleunigen

DevOps – The 3 Ways

- 1st Way
  - Erkennen von Business-Needs, die als Service angeboten werden sollen
- 2nd Way
  - Verkürzung und Verstärkung von Feedback-Schleifen
  - Schnelles und konstantes Feedback aus der gesamten Wertschöpfungskette
- 3rd Way
  - Etablierung einer Kultur des kontinuierlichen Experimentierens und Lernen aus Fehlern



DevOps – Architektur einer Deployment Pipeline



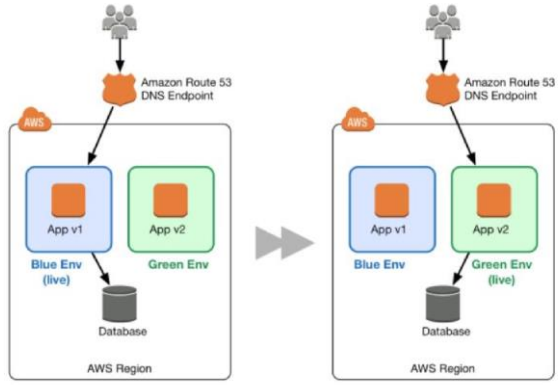
https://de.slideshare.net/CiscoCanada/expanding-your-impact-withprogrammability-in-the-data-center

DevOps – CI vs. CD vs. CD

- Continuous Integration
  - Regelmäßige Code Commits, die in den Master gemerged werden
    - Builden und Ausführen von Tests (gegen den Build)
- Continuous Delivery
  - Deploy merged Code, mit User Interaktion – z.B. Umgebung
    - CI + Release Management
- Continuous Deployment
  - Automatisch bis in Produktion deployen, falls alle Tests erfolgreich ausgeführt werden

DevOps – Releases

- Umgebungsbasierte Release
  - Stellen mehrere Umgebungen für Deployments zur Verfügung
- Applikationsbasierte Releases
  - Features anhand von Konfigurationen freischalten (Feature Toggles)
  - Applikationen müssen dies unterstützen



https://d1.awsstatic.com/whitepapers/AWS_Blue_Green_Deployments.pdf

DevOps – Infrastructure as Code (1)

- Bisherige Probleme ohne Infrastructure as Code
  - Fehlende wiederholbare Prozesse „Snowflakes“
  - Eingeführte Inkonsistenzen aufgrund von menschlichen Fehlern
- Löst Problem der abweichenden Konfigurationen mithilfe von Automatisierung und erhöht gleichzeitig die Agilität
  - Ziel: Infrastruktur automatisch mithilfe von Konfigurationsdateien erstellen
  - Konfigurationen wie Source Code behandeln und versionieren (VCS)

DevOps – Infrastructure as Code (2)

- Immutable Infrastructure
  - Ziel: Stabile und abweichungsfreie Infrastrukturen für Applikationen
  - Immutable: Einmal instanziiert, kann es nicht mehr verändert werden

Microservices und Container

/informatik & security /fh/// st.pölten

- Microservices
  - Deployment Zyklen beschleunigen
  - Ownership und Innovation fördern
  - Wartbarkeit und Skalierbarkeit einer Applikation verbessern

- Container
  - Bündeln Code und deren Abhängigkeiten in Packages
  - Platzsparender als virtuelle Maschinen
  - Orchestrierung ermöglicht automatisierte Konfiguration, Koordination und Management

DevOps aus Sicht von …

Security

https://1c7qp243xy9g1qeffp1k1nvo-wpengine.netdna-ssl.com/wp-content/uploads/2017/08/jb.png

DevOps

https://i.chzbgr.com/full/6134060544/hB0C7D478/

Gründe für DevSecOps

Sabotage: Tesla verklagt Ex-Mitarbeiter

Kurzmeldungen 21.06.2018 07:35 Uhr

Tesla will einen ehemaligen Mitarbeiter vor Gericht bringen, der das Unternehmen angeblich gezielt sabotiert hat. Der Beschuldigte soll bereits zugegeben haben, Tes Produktionssystem gehackt und mehrere Gigabyte an internen Daten an Dritte weitergegeben zu haben. Das geht aus der am Mittwoch (20. Juni 2018) bei einem Gericht in Las Vegas eingereichten Anklageschrift hervor. Das volle Ausmaß der „illegalen Aktivitäten" werde noch ermittelt.

Tesla wirft dem Ex-Angestellten auch vo falsche Angaben gegenüber der Presse gemacht zu haben. Der Mitarbeiter soll Oktober 2017 als Techniker in Teslas Batteriefabrik „Gigafactory" im US-Bundesstaat Nevada angeheuert und d Zugang zu hochsensiblen internen Informationen gehabt haben. Danach h es rasch Ärger mit dem Mann gegeben,

5 Millionen Mal heruntergeladen: Bösartige Docker-Container schürfen Monero

15.06.2018 14:50 Uhr – Fabian A. Scherschel

(Bild: Pixabay)

Zehn Monate lang waren Docker-Images mit Hintertür über Docker Hub verfügbar, obwohl die Verantwortlichen längst über den Schadcode informiert waren.

Unbekannte haben diverse Docker-Images in das offizielle Repository der Containerverwaltungs-Software geladen, die mit versteckten Hintertüren versehen sind. Einmal ausgeführt, übernahmen die Angreifer die Kontrolle über den Container und installierten Krypto-Mi
ten Angrei
verfügbar
damit Mo

Drupalgeddon 2: Angreifer attackieren ungepatchte Drupal-Webseiten

Alert! 16.04.2018 10:48 Uhr – Dennis Schirrmacher

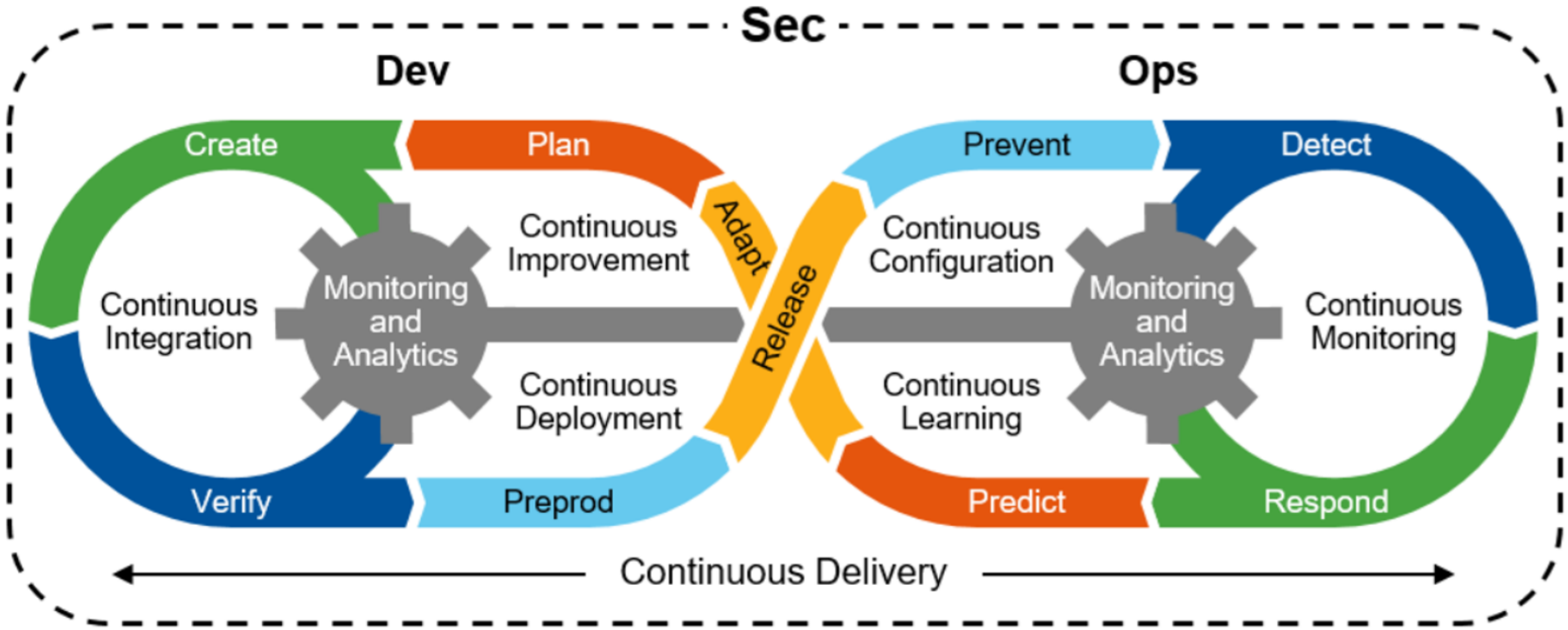
50.000 Euro Strafe: DailyMotion-Hacker fanden Admin-Passwort bei GitHub

03.08.2018 11:32 Uhr – Fabian A. Scherschel

DevSecOps – Definition

- Synonyme
  - DevOpsSec, Continuous Security, Rugged DevOps, SecDevOps
- DevSecOps
  - Ziel: Automatische und durchgehende Einbindung von Sicherheitskontrollen ohne manueller Interaktion
  - Verankerung von Sicherheitsprinzipien in den Entwicklungsprozess
    - Übertragung der Zuständigkeit auf das gesamte Team
  - Security wird Teil des Continuous Delivery, Continuous Learning und Continuous Improvement Prozesses
    - Security wird nicht mehr als Blocker/Temposchwelle angesehen

DevSecOps – Überblick



10 Things to Get Right for Successful DevSecOps," Gartner, Inc., 2017

Vorteile von DevSecOps

- Kostenreduktion durch frühes Erkennen und Lösen von Sicherheitsproblemen in der Entwicklungsphase
- Security Auditing-, Monitoring- und Benachrichtigungssysteme helfen bei der Erkennung von Angreifern
- Secure by Design und Secure Coding
  - Entwickler wenden u.a. „sichere“ Design Patterns an
    - Schulungen und automatisiertes Application Security Tests (AST) nötig
- Security betrifft alle

Mögliche Sicherheitsrisiken bei diversen Praktiken und Tools (1)

/informatik & security /fh/// st.pölten

- Container
  - Kernel Exploit
    - Kernel Schwachstelle gefährdet das gesamte System und die darauf laufenden Services
  - Denial of Service
    - Einzelner Container könnte das ganze System lahmlegen (falls Ressourcen nicht eingeschränkt werden)
  - Container Breakouts
    - Schlechtere Isolation als bei klassischen Virtualisierungen
  - Poisoned Images
    - Abhängigkeiten werden einfach und schnell definiert
    - Risiko mithilfe von Trusted Registries und Image-Scanning verringern
  - Compromising Secrets
    - Schlecht geschützte Secrets, wie Zugangsdaten zu Datenbanken oder Services

Mögliche Sicherheitsrisiken bei diversen Praktiken und Tools (2)

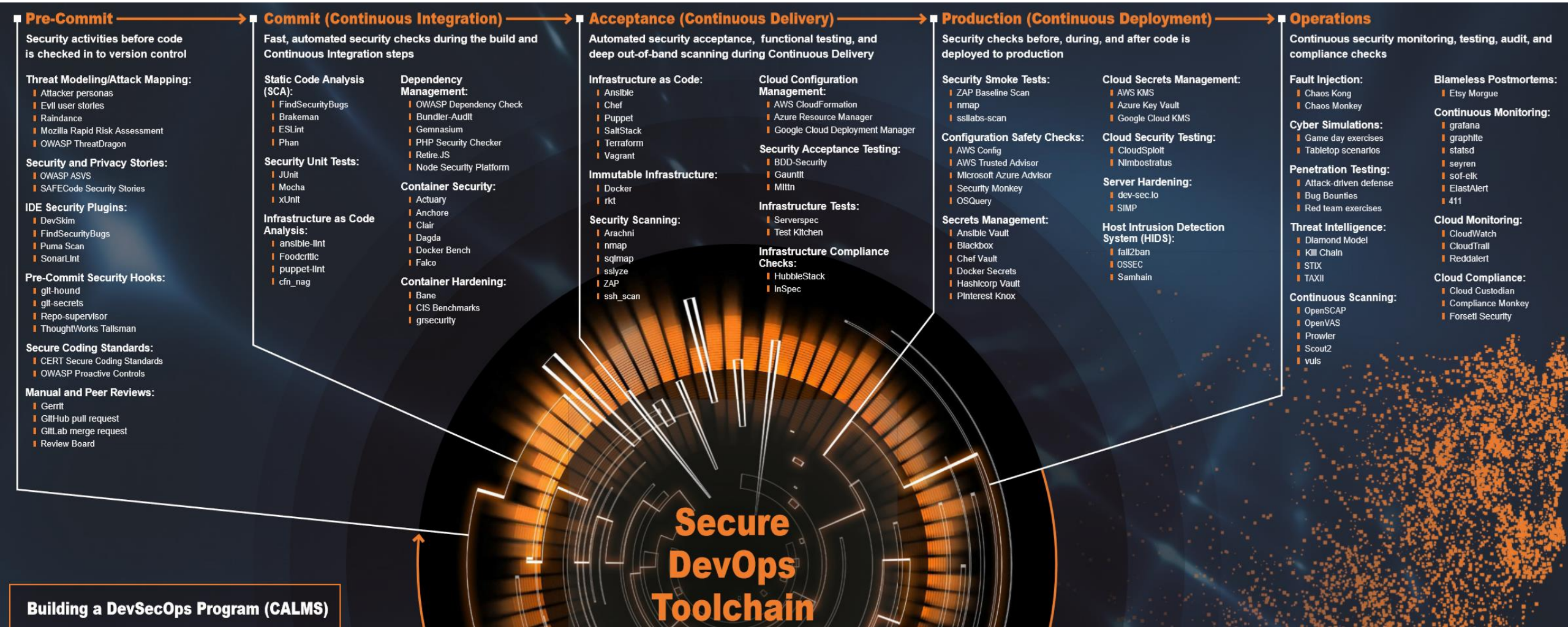
- Polyglott
  - Ermöglicht Entwicklern eine beliebige Programmiersprache, Framework oder Runtime zu wählen
  - Probleme bei Static Application Security Testing (SAST) und Software Component Analysis (SCA) Lösungen
    - Bieten nur wenig oder keine Unterstützung
    - Unternehmen muss Secure Coding Guidelines entwickeln und Frameworks selbst auf Security Fähigkeiten und Risiken überprüfen

Mögliche Sicherheitsrisiken bei diversen Praktiken und Tools (3)

- SAST/DAST
  - Hohe False Positives Rate → muss mithilfe von Feintuning verringert werden (falls möglich)
    - False Negatives werden in Kauf genommern
  - Problem bei nicht unterstützten Programmiersprachen
  - Tools, besonders im Fall von Open-Source-Tools, sind auf Checks zur Verbesserung des Codes, wie Linting und Basis-Checks auf diverse Good/Bad-Practices, beschränkt

DevSecOps – Tools (Entwicklung/PipeLine)

/informatik & security /fh/// st.pölten



Absicherung einer Deployment Pipeline in 5 Stufen

/informatik & security /fh/// st.pölten

- Pre-Commit
  - Sicherheitsbezogene Aktivitäten, wie Threat Modeling, Peer Reviews und Security Trainings, vor Code Check-in
- Commit (Continuous Integration)
  - Schnelle automatisierte Security Checks, wie SAST, Dependency Management und/oder Container Hardening während des Build-Prozesses und der Continuous Integration
- Acceptance (Continuous Delivery)
  - Automatisierte sicherheitsbezogene Akzeptanztests, Funktionstests und umfassende Out-of-Band Scans während des Continuous Delivery Prozesses
- Production (Continuous Deployment)
  - Security Checks vor, während und nach dem Deployment auf Produktion
- Operations
  - Continuous Monitoring, Penetration Tests, Continuous (Vulnerability) Scanning, Audits und Compliance Checks

Disclaimer

- Kritischer Aspekt jedes Tools ist die Automatisierung und der damit verbundenen Anbindungsmöglichkeiten durch z.B. APIs

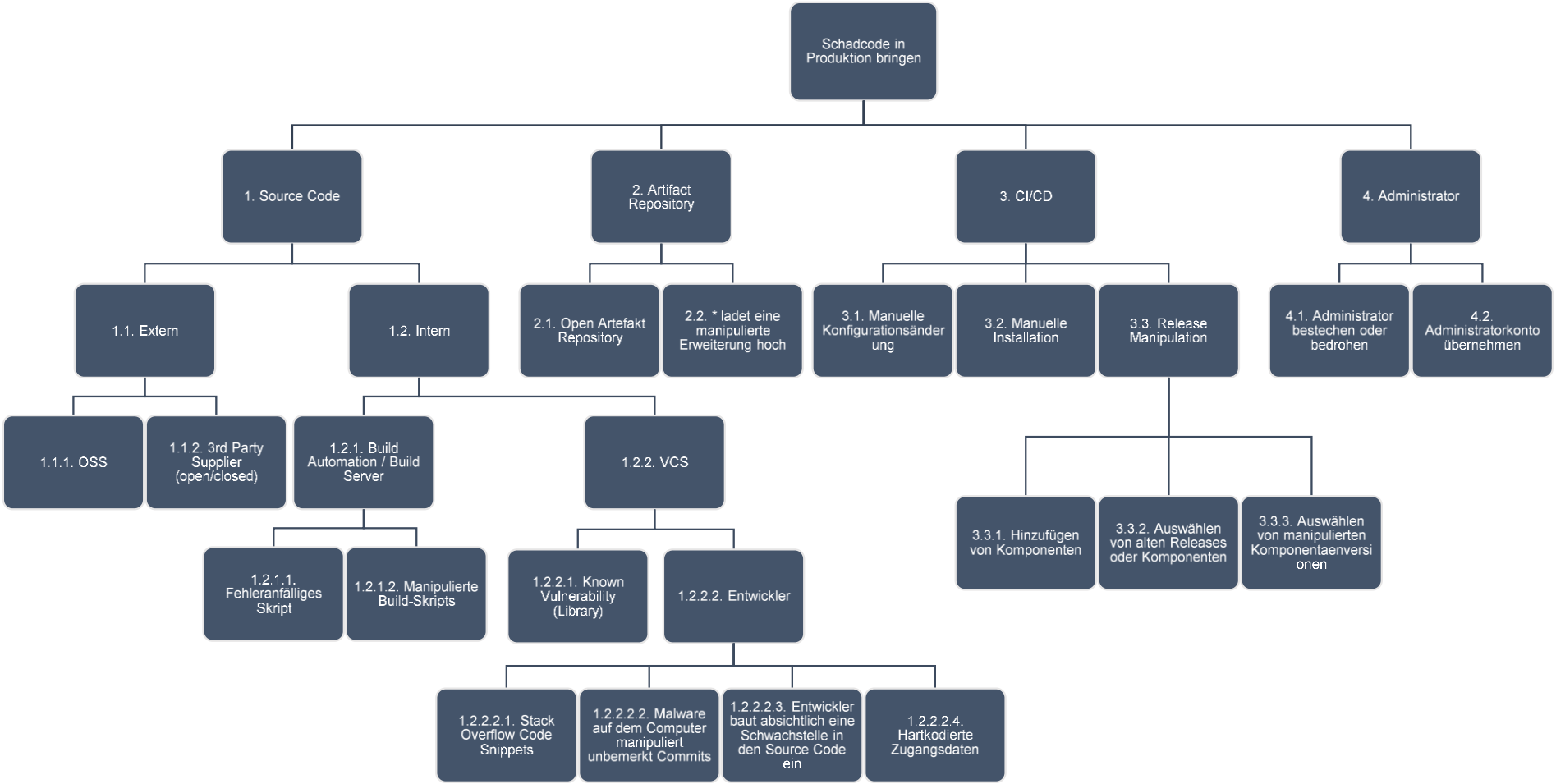
- In den folgenden Attack Trees wird folgendes nicht berücksichtigt:
  - Configuration Management & Control
  - Continuous Governance
  - Event Management & Incident Response

Attack Tree Ziele

- Definition
  - bieten einen formalen und strukturierten Weg um die Sicherheit eines Systems zu beschreiben
    - Ziel des Angriffs wird als Wurzel gekennzeichnet
    - Angriffe gegen das System, die zum Ziel führen sollen, werden als Blätter abgebildet

- Ziele:
  - Schadcode in Produktion bringen
  - Source Code und Konfigurationen mithilfe der Deployment Pipeline exfiltrieren
  - CI/CD Server übernehmen

Attack Tree – Ziel 1



Attack Tree – Ziel 1 (Auflistung)

- Source Code
  - Extern
    - OSS
    - 3rd Party Supplier (open/closed)
  - Intern
    - Build Automation / Build Server
      - Fehleranfälliges Skript
      - Manipulierte Build-Skripts
    - VCS
      - Known Vulnerability (Library)
      - Entwickler
        - Stack Overflow Code Snippets
        - Malware auf dem Computer manipuliert unbemerkt Commits
        - Entwickler baut absichtlich eine Schwachstelle in den Source Code ein
        - Hartkodierte Zugangsdaten
- Artifact Repository
  - Open Artefakt Repository
  - * ladet eine manipulierte Erweiterung hoch
- CI/CD
  - Manuelle Konfigurationsänderung
  - Manuelle Installation
  - Release Manipulation
    - Hinzufügen von Komponenten
    - Auswählen von alten Releases oder Komponenten
    - Auswählen von manipulierten Komponentenversionen
- Administrator
  - Administrator bestechen oder bedrohen
  - Administratorkonto übernehmen

Attack Tree – 1.1.1 OSS

- Open-Source Software birgt wie jede Software die Gefahr von Schwachstellen
- Beispiel einer OSS Schwachstelle wäre Heartbleed
- Mögliche Gegenmaßnahmen sind:
  - Continuous Monitoring
  - Dependency Check
  - SCA / SAST
  - Security Scanning
  - Threat Intelligence

https://upload.wikimedia.org/wikipedia/commons/thumb/d/dc/Heartbleed.svg/1200px-Heartbleed.svg.png

Attack Tree – 1.2.2.2.4 Hartkodierte Zugangsdaten

- Weiterhin allgegenwärtig
- Entdeckung durch z.B. Brute-Force Angriffe oder Reverse Engineering
- Mögliche Gegenmaßnahmen sind:
  - Pre-Commit Security Hooks
  - Manuelle- und Peer Reviews
  - SCA / SAST
  - Security Scanning
    - Funktionieren Testing Zugangsdaten in Produktion?
  - Security Training
  - Secure Coding Standards
  - Secrets Management

c't deckt auf: Notruf 112 mit fatalem Datenleck

16.03.2018 07:00 Uhr – Ronald Eikenberg

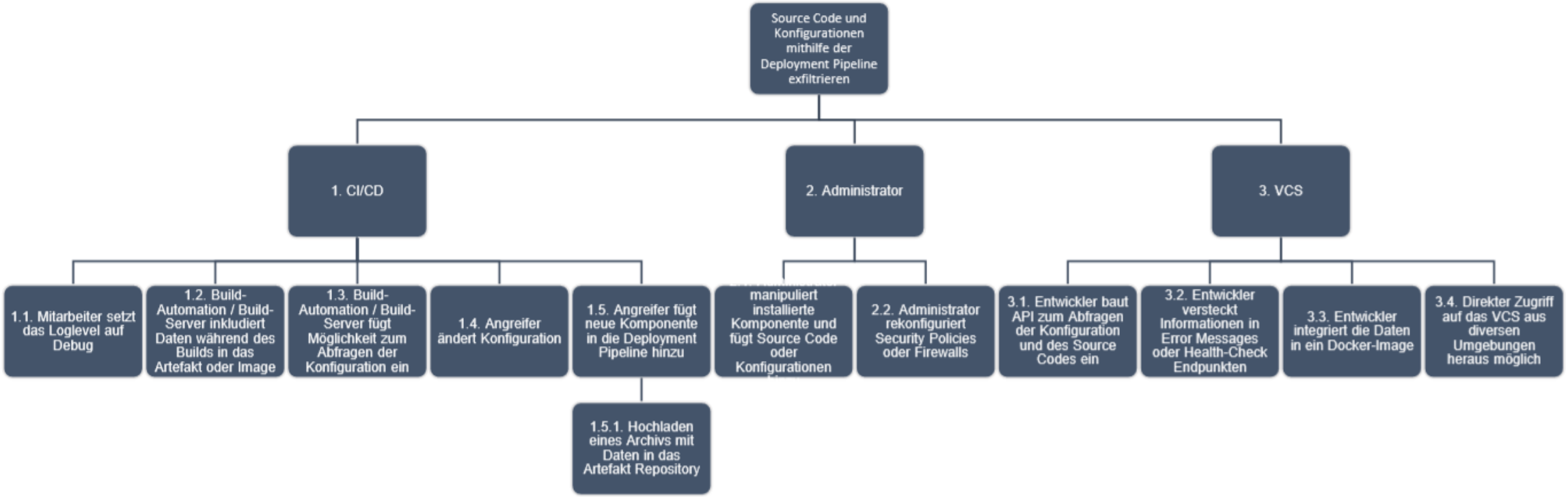
(Bild: Thomas Warnack)

c't hat in einer App für Rettungskräfte ein Datenleck entdeckt, das bis vor kurzem Einsatzdaten inklusive detaillierten Patienteninfos offenlegte.

Attack Tree – 2.1 Open Artefakt Repository

- Artefakte, die von einer Anwendung benötigt und dynamisch beim Build-Prozess nachgeladen werden
- Fehlende Einschränkungen ermöglichen das dynamische Nachladen jeglicher Abhängigkeiten aus dem Internet
- Mögliche Gegenmaßnahmen (außer der Sperrung des Nachladens aus dem Internet)
  - Dependency Check
  - SCA / SAST
  - Security Scanning
  - Secure Coding Standards

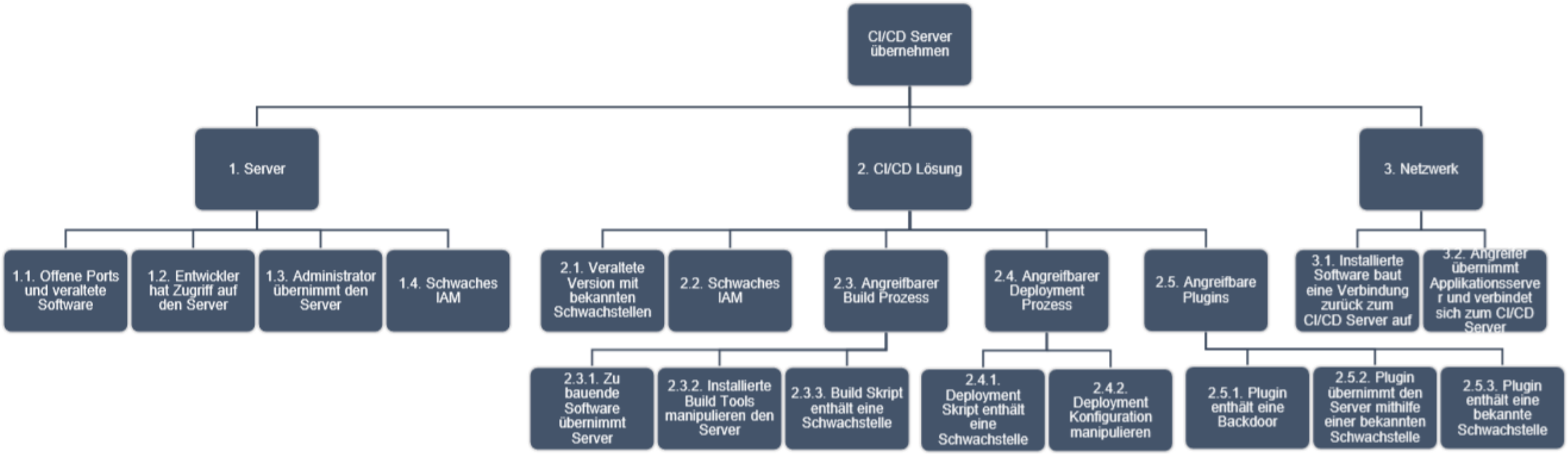
Attack Tree – Ziel 2



Attack Tree – Ziel 2 (Auflistung)

- CI/CD
  - Mitarbeiter setzt das Loglevel auf Debug
  - Build-Automation / Build-Server inkludiert Daten während des Builds in das Artefakt oder Image
  - Build-Automation / Build-Server fügt Möglichkeit zum Abfragen der Konfiguration ein
  - Angreifer ändert Konfiguration
  - Angreifer fügt neue Komponente in die Deployment Pipeline hinzu
    - Hochladen eines Archivs mit Daten in das Artefakt Repository
- Administrator
  - Administrator manipuliert installierte Komponente und fügt Source Code oder Konfigurationen hinzu
  - Administrator rekonfiguriert Security Policies oder Firewalls
- VCS
  - Entwickler baut API zum Abfragen der Konfiguration und des Source Codes ein
  - Entwickler versteckt Informationen in Error Messages oder Health-Check Endpunkten
  - Entwickler integriert die Daten in ein Docker-Image
  - Direkter Zugriff auf das VCS aus diversen Umgebungen heraus möglich

Attack Tree – Ziel 3



Attack Tree – Ziel 3 (Auflistung)

- Server
  - Offene Ports und veraltete Software
  - Entwickler hat Zugriff auf den Server
  - Administrator übernimmt den Server
  - Schwaches IAM
- CI/CD Lösung
  - Veraltete Version mit bekannten Schwachstellen
  - Schwaches IAM
  - Angreifbarer Build Prozess
    - Zu bauende Software übernimmt Server
    - Installierte Build Tools manipulieren den Server
    - Build Skript enthält eine Schwachstelle
  - Angreifbarer Deployment Prozess
    - Deployment Skript enthält eine Schwachstelle
    - Deployment Konfiguration manipulieren
  - Angreifbare Plugins
    - Plugin enthält eine Backdoor
    - Plugin übernimmt den Server mithilfe einer bekannten Schwachstelle
    - Plugin enthält eine bekannte Schwachstelle
- Netzwerk
  - Installierte Software baut eine Verbindung zurück zum CI/CD Server auf
  - Angreifer übernimmt Applikationsserver und verbindet sich zum CI/CD Server

Top 10 Auflistung der häufigsten Gegenmaßnahmen

Gegenmaßnahme	Häufigkeit
Continuous Monitoring	34
Security Scanning	18
Host Intrusion Detection System (HIDS)	17
SCA / SAST	15
Manuelle- und Peer Reviews	15
Server Hardening	14
Security Smoke Tests	8
Continuous Scanning	8
Immutable Infrastructure	7
Infrastructure Compliance Checks	7

Quellen (u.a.)

- SANS DevOps Toolchain - https://www.sans.org/security-resources/posters/secure-devops-toolchain-swat-checklist/60/download
- G. Kim, J. Humble, P. Debois und J. Willis, The DevOps Handbook
- J. Bird, B. Filkins, E. Johnson und F. Kim, 2017 State of Application Security: Balancing Speed and Risk
- SAFECode, Tactical Threat Modeling
- J. Bird, DevOpsSec: Delivering Secure Software Through Continuous Delivery
- N. MacDonald und I. Head, 10 Things to Get Right for Successful DevSecOps