Cyber Resilience: Ist Ihre Software sicher genug?

Wie ein herkömmlicher Entwicklungsprozess zu einem Secure Software Development Lifecycle (SSDLC) wird

Software Development Life Cycle SDLC

CYBERSECURITY

Cyberattacke auf Läderach

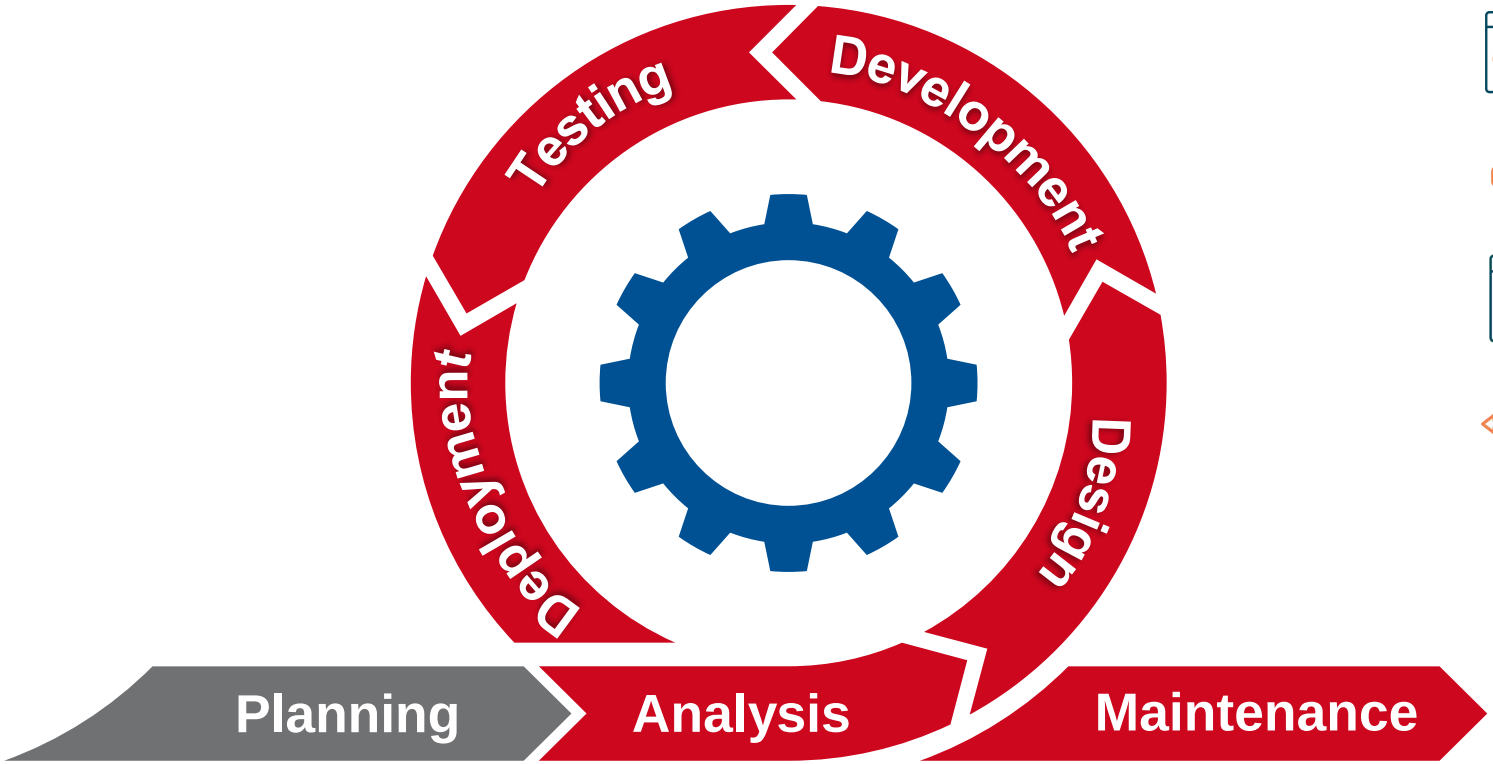
Cyberkriminelle attackieren Schweizer Schoggi

Do 08.09.2022 - 12:24 Uhr
von Coen Kaat und yzu

Cyberkriminelle haben Läderach attackiert. Der Verkauf ist nicht beeinträchtigt – im Gegensatz zu Produktion, Logistik und Administration des Chocolatiers.

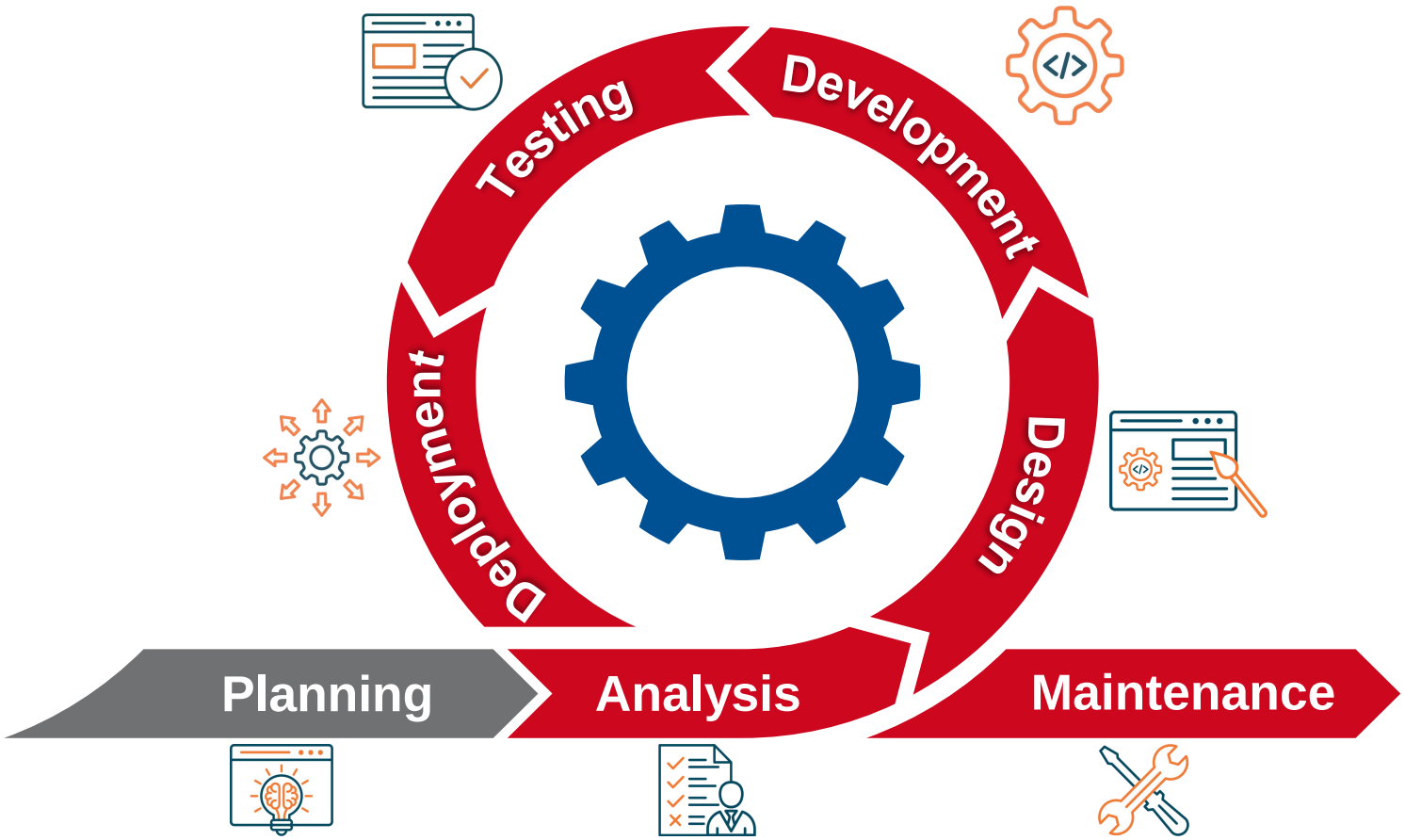
(Source: PublicDomainPictures / Pixabay)

Software Development Life Cycle SDLC



- Planning
- Analysis
- Design
- Development
- Testing
- Deployment
- Maintenance

Software Development Life Cycle SDLC



Wieso brauchen wir einen SDLC?

Um Risiken zu minimieren:

- Kosten
- Zeit
- Qualität & Erwartungen

SSDLC: Planning Phase

SDLC

- Umfang festlegen
- Projekt- und Zeitpläne
- Kostenschätzungen & Beschaffungsanforderung

SSDLC:

- Gedanken und Fragen bezüglich Security
- Erste grundlegende Security Anforderungen
- Security Awareness Training

Planning | Analysis | Design | Development | Testing | Deployment | Maintenance

SSDLC: Analysis Phase

SDLC

- Sammeln und bewerten von Requirements, Use-Cases
- Wandeln von Business Requirements in Software Features
- Requirements dokumentieren

SSDLC:

- Definieren von Misuse- und Abuse-Cases
- Sammeln und bewerten von Security Requirements
- Security Requirements dokumentieren
- Kritische Fragen zum Projekt stellen

SSDLC: Analysis Phase

Fragen zum Security Requirements:

- Wie wird das Benutzer-Management sicher gestellt
- Welche Schwachstellen haben ähnliche Projekte
- OWASP Define Security Requirements
- OWASP Application Security Verification Standard (ASVS)
- IEC 62443-4-2:2019

SSDLC: Analysis Phase

IEC 62443-3-3 4.0

7.7 CR 3.5 – Input validation

7.7.1 Requirement

Components shall validate the syntax, length and content of any input data that is used as an industrial process control input or input via external interfaces that directly impacts the action of the component.

SRs and REs	SL 1	SL 2	SL 3	SL 4
CR 3.5 – Input validation	✓	✓	✓	✓

Planning | Analysis | Design | Development | Testing | Deployment | Maintenance

SSDLC: Design Phase

SDLC:

- Architektur festlegen
- Design bestimmen
- Entscheidungen dokumentieren
- Erste Prototypen entstehen

SSDLC:

- Threat modelling & Risk assessment
- Secure Architectural and Design Patterns
- Architectural and Design Review

SSDLC: Design Phase

Ressourcen:

- Secure Design Patterns by SEI Carnegie Mellon
- Security Architecture by The OpenGroup
- Patternlandscape by Open Security Architecture

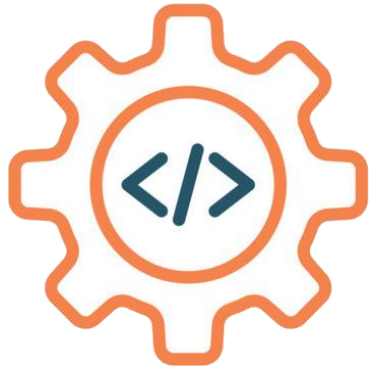
Software Engineering Institute
Carnegie Mellon

THE Open GROUP

OSA

Planning | Analysis | **Design** | Development | Testing | Deployment | Maintenance

SSDLC: Development Phase



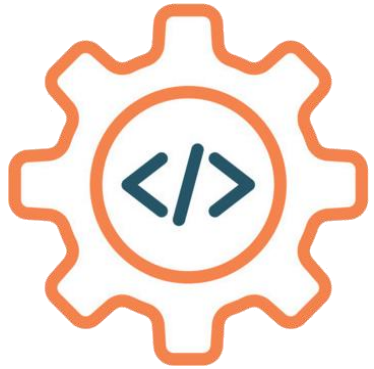
SDLC:

- Umsetzung gemäss Architektur und Design
- Code Reviews
- Komponenten- and Unit-Tests

SSDLC:

- Secure Coding Guidelines resp. Checkliste
- Developers: Bewusstsein für potentielle Security Risiken
- Static Application Security Testing (SAST)
- Software Composition Analysis (SCA)

SSDLC: Development Phase

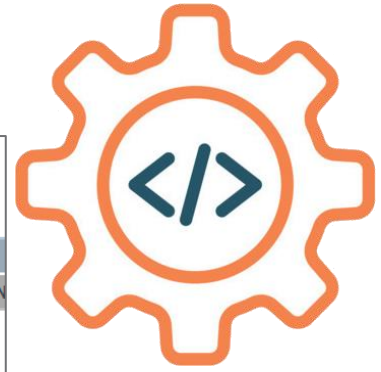


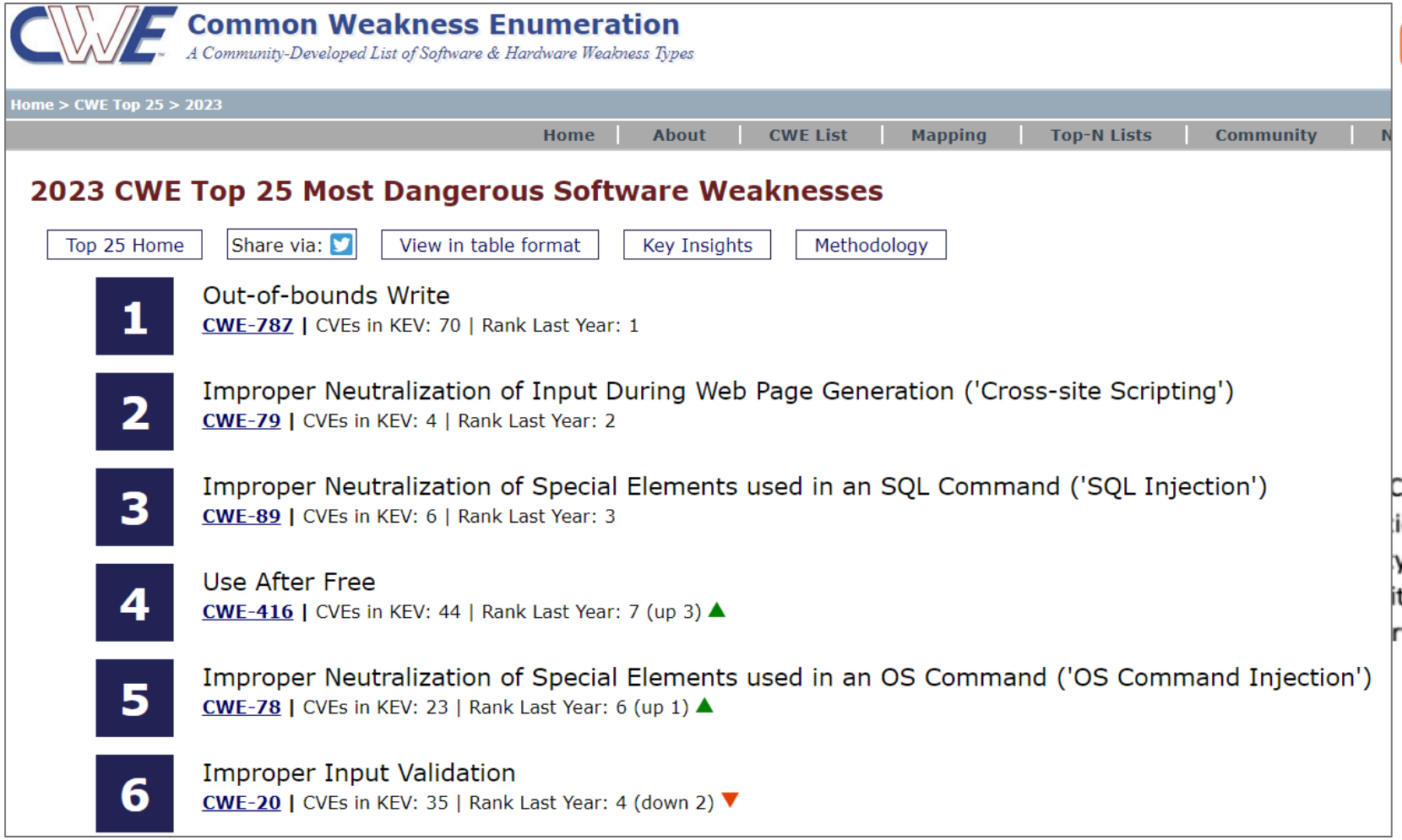
SAST:

- Code Quality Tools: clang-tidy, clang-sanitizer …
- Static Code Analyzer:
  - Free: CodeChecker
  - Commercial: SonarQube
- Secret detection/scanning tool
  - Free: detect-secrets, gitleaks
  - Commercial: Snyk, GitGuardian
- SCA -> SBOM
  - Free: cyclonDX & dependency-track, ScanCode.io
  - Commercial: Vigiles, Mend.io

Planning | Analysis | Design | **Development** | Testing | Deployment | Maintenance

SSDLC: Development Phase





Common Weakness Enumeration

A Community-Developed List of Software & Hardware Weakness Types

Home > CWE Top 25 > 2023

Home | About | CWE List | Mapping | Top-N Lists | Community | N

2023 CWE Top 25 Most Dangerous Software Weaknesses

Top 25 Home | Share via: | View in table format | Key Insights | Methodology

1. Out-of-bounds Write
   CWE-787 | CVEs in KEV: 70 | Rank Last Year: 1
2. Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
   CWE-79 | CVEs in KEV: 4 | Rank Last Year: 2
3. Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
   CWE-89 | CVEs in KEV: 6 | Rank Last Year: 3
4. Use After Free
   CWE-416 | CVEs in KEV: 44 | Rank Last Year: 7 (up 3) ▲
5. Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
   CWE-78 | CVEs in KEV: 23 | Rank Last Year: 6 (up 1) ▲
6. Improper Input Validation
   CWE-20 | CVEs in KEV: 35 | Rank Last Year: 4 (down 2) ▼

Components
ication Failures
y Failures
itoring Failures*
ry (SSRF)*

Planning | Analysis | Design | **Development** | Testing | Deployment | Maintenance

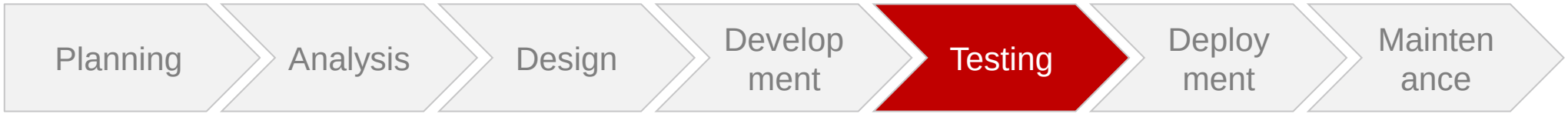
SSDLC: Testing Phase

SDLC:

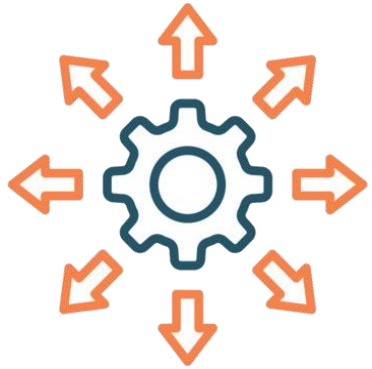
- QA-Team sucht nach Unzulässigkeiten
- Blackbox Testing
- Dokumentieren

SSDLC:

- Dynamic Application Security Testing (DAST)
- Interactive Application Security Testing (IAST)
- Penetration Testing



SSDLC: Deployment Phase



SDLC

- Deployment in das produktive Environment

SSDLC

- Konfiguriere dein System sicher -> CIS-Benchmarks
- Sichere Deployment Praktiken -> OpenStack
- Security Guidelines

CIS Center for Internet Security®
Creating Confidence in the Connected World.™

openstack®

Planning | Analysis | Design | Development | Testing | **Deployment** | Maintenance

SSDLC: Maintenance Phase

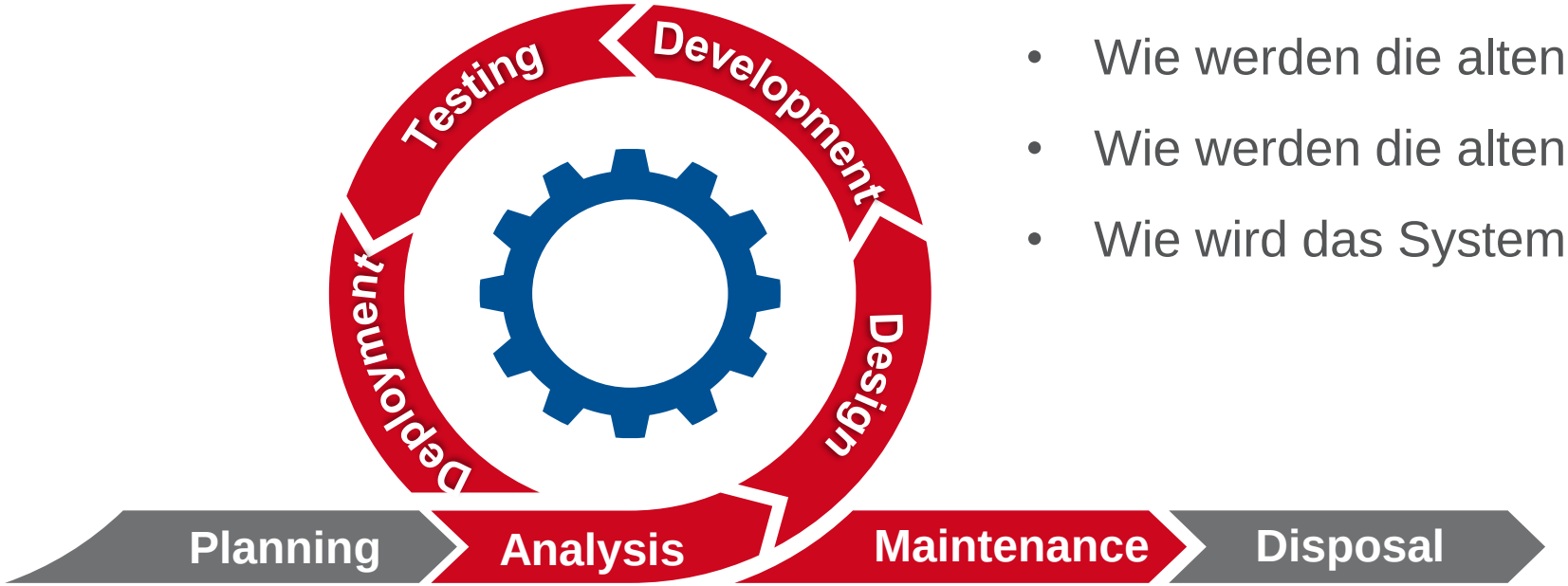
SDLC

- Support für zukünftige Updates und Bug Fixes

SSDLC

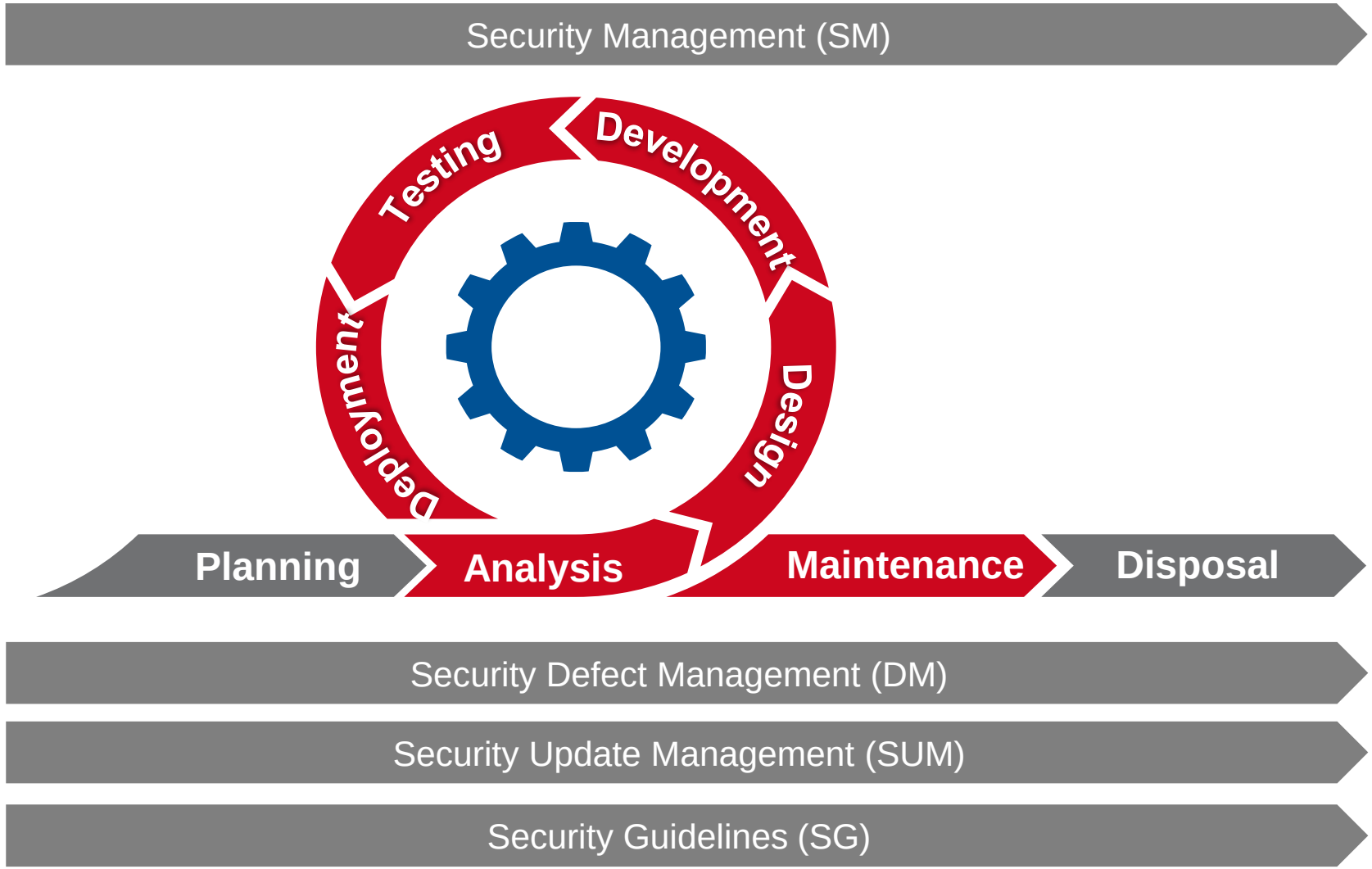
- Monitore das System (IPS, IDS)
- Vorfallreaktionsplan (Incident Response Plan)
- PSIRT Service Framework
- Für Benutzer: Ein einfaches rapportieren von Fehlern

SSDLC: Disposal Phase



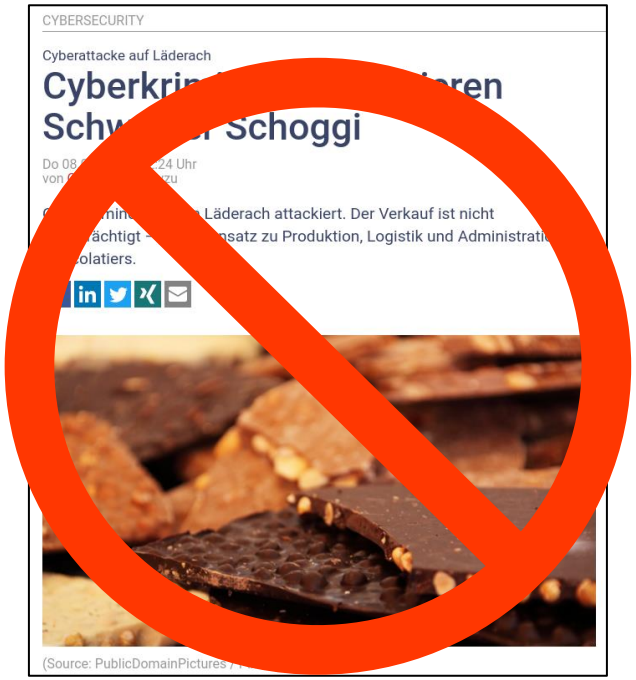
- Wie werden die alten Daten archiviert
- Wie werden die alten Daten migriert
- Wie wird das System sicher "entsorgt"

SSDLC nach IEC 62443-4-1



Vorteile eines Secure SDLC

- Security-Bewusstsein wird gefördert
- Hilft bei der Reduzierung von Risiken und Kosten
- Fehler und Schwachstellen werden frühzeitig erkannt
- Erhöht die Qualität wie Robustheit
- Stärkt das Vertrauen der Kunden



MAKING VISIONS WORK.

Jürgen Messerer

bbv Software Services AG
Heinrichstrasse 241
8005 Zürich
www.bbv.ch

juergen.messerer@bbv.ch
Telefon +41 41 429 01 11
www.bbv.ch