fortiss

fortiss

Landesforschungsinstitut des Freistaats Bayern für softwareintensive Systeme



Mit KI zu mehr Sicherheit bei der Softwareentwicklung

KI für den Mittelstand

München, 21. 10. 2026



Florian Angermeier
Wissenschaftler Requirements Engineering
+49 89 3603522 279
angermeier@fortiss.org

dina

EDIH European Digital Innovation Hubs Network

Co-funded by the European Union

Über mich

Sicherheit vom Betrieb zur Entwicklung

System Administrator

- Betrieb und Absicherung diverser IT-Infrastruktur

DevSecOps Engineer

- Integration und Automatisierung von Security Aktivitäten in DevOps & Agile

Wissenschaftler

- Automatisierung von Security Compliance
  - Proaktiv (z.B. Empfehlen relevanter Sicherheitsanforderungen)
  - Reaktiv (z.B. Automatisierte Compliance Analyse)



Workshop-Ziel & Agenda

Mit KI zu mehr Sicherheit bei der Softwareentwicklung

Ziel

Realistische Einschätzung, wie KI-Sicherheitsaktivitäten in der Softwareentwicklung **sinnvoll** automatisieren kann

Agenda

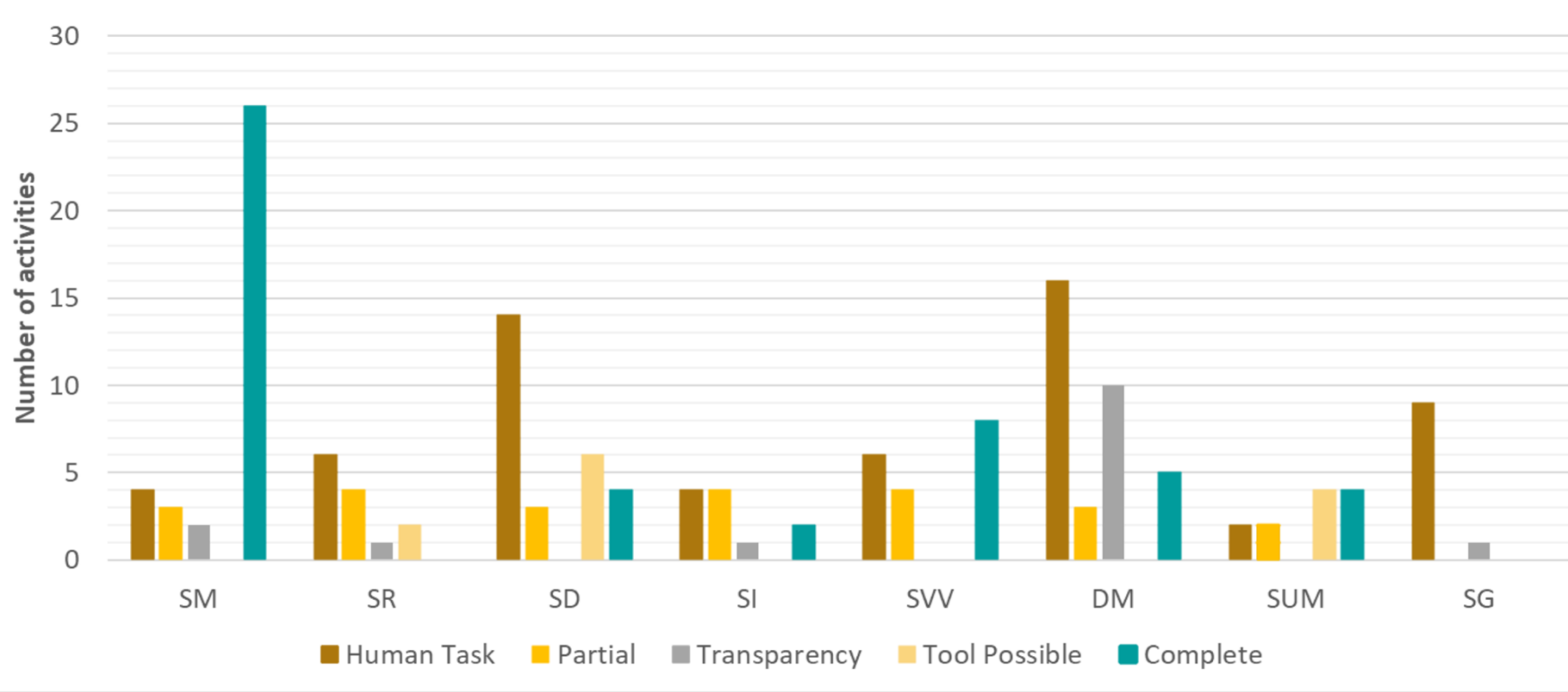
- Motivation
- Voraussetzungen für Cybersicherheitsautomatisierung
- 2-3 Anwendungsfälle von KI in der Cybersicherheit
    - Voraussetzungen
    - Grenzen und Risiken
    - Beispiele

Nicht im Scope dieses Workshops

- Performanz und Vergleiche bestimmter LLMs
- Rechtliche Aspekte (z.B. Datenschutz)

Motivation

Automatisierungspotenzial



- Manuell: 38%
- Teils: 31%
- Voll: 31%

Moyón et al., Integration of Security Standards in DevOps Pipelines: An Industry Case Study, 2020, https://doi.org/10.1007/978-3-030-64148-1_27

Faktoren erfolgreicher Integration

Beispiele

Prozess vor Technologie

Erst den Security-Prozess klären, dann automatisieren. KI beschleunigt nur, was schon funktioniert.

Kompetenz vor Technologie

Das Team muss KI-Ergebnisse fachlich prüfen können. Ohne Fachwissen bleibt die menschliche Prüfung wirkungslos.

Existenz der relevanten Daten

Architektur, Requirements und Findings müssen vorliegen. Ohne den nötigen Kontext, rät das Modell.

Integration in Toolchains

KI wirkt dort, wo gearbeitet wird: IDE, CI/CD, Ticketsystem. Kein zusätzliches Insel-Tool.

Prozess vor Technologie

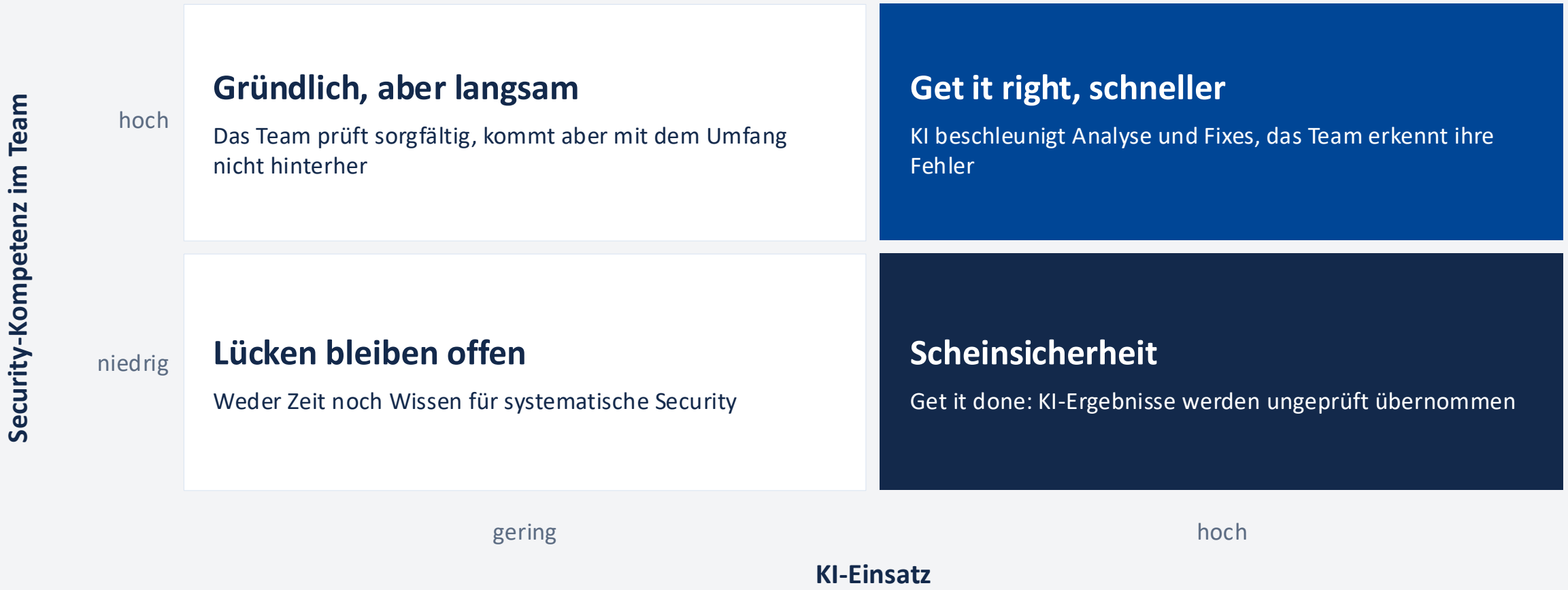
Prozesse bieten Struktur für effektive Security

	Ohne Prozess	Mit Prozess
Wo setzt KI an?	Wo gerade ein Tool verfügbar ist	An einer definierten Aktivität einer Phase
Wer prüft das Ergebnis?	Zuständigkeiten unklar	Die verantwortliche Rolle, z.B. Reviewer
Wo wird es dokumentiert?	Im Chatverlauf, wenn überhaupt	Als Nachweis, z.B. für die EU CRA
Woran misst man Erfolg?	Am Gefühl, schneller zu sein	An den Zielen der Aktivität

Strukturgeber: z.B. IEC 62443-4-1, NIST SDDF, BSI CON.8/TR-03185, EU CRA/TR-03183

Kompetenz vor Technologie

KI verstärkt die Kompetenz, die im Team vorhanden ist



Interaktiv

Auf welche Entwicklungsphasen sollen wir uns fokussieren?

Anforderungen

Schutzbedarfsanalyse

Erhebung Compliance Anforderungen

Ableitung konkreter Security Anforderungen

Priorisierung von Security Anforderungen

Architektur

Threat Modeling

Security by Design

Third-Party Komponenten

Dokumentation

Implementierung

Secure Coding Standards

Security Code Reviews

Prüfung sicherheitskritischen Codes

Automatisierte Security Fixes

Testing

Dynamische Anwendungssicherheitstests

Schwachstellen-Scans

Penetrationstests

Validierung gefundener Schwachstellen

Anforderungen

Anforderungen

Überblick ausgewählter Aktivitäten

- **Schutzbedarfsanalyse** von Daten, Funktionen und Assets
- Erhebung regulatorischer und vertraglicher **Compliance Anforderungen**
- Ableitung konkreter **Security Anforderungen**
- **Priorisierung** von Security Anforderungen

Schutzbedarfsanalyse

Anforderungen

- **Ziel:** Schutzbedarf von Daten, Funktionen und Assets nach Vertraulichkeit, Integrität, Verfügbarkeit bewerten
- **Input:** Asset- und Datenflussübersicht, Prozesse, Schadensszenarien, Compliance-Verzeichnis
- **Output:** Schutzbedarfsklassifizierung je Asset mit Begründung und Schadenskategorie
- **Nötige Kompetenzen:** Domänen- und Geschäftswissen, Methodik, Security-Grundlagen
- **Risiken:** Unvollständiges Asset-Inventar, Fehleinschätzung des Schadens, KI-Bewertungen ohne Kontext

Beispiel

- **CISO Assistant (Open Source):** Business Impact Analyse je Asset. Lokale KI-Engine und MCP-Server

Erhebung Compliance Anforderungen

Anforderungen

- **Ziel:** Alle für das Produkt geltenden regulatorischen, normativen und vertraglichen Sicherheitspflichten identifizieren
- **Input:** Produktbeschreibung, Zielmärkte, Kundenverträge, Regularien und Normen (z.B. EU CRA, NIS2, IEC 62443)
- **Output:** Verzeichnis anwendbarer Compliance-Anforderungen mit Quelle, Geltungsbereich und Verantwortlichen
- **Nötige Kompetenzen:** Regulatorisches und juristisches Fachwissen, Produktwissen, Security-Grundlagen
- **Risiken:** Übersehene oder falsch interpretierte Pflichten, veraltete Normstände, von der KI erfundene Normverweise

Beispiele

- **OSCAL Compass Agentic Authoring (CNCF):** KI-Skills wandeln Regelwerk-PDFs in Kontrollkataloge um
- **compliance-trestle-mcp (CNCF), mcp-server-for-oscal (AWS Labs):** OSCAL-Kataloge per MCP pflegen
- **CISO Assistant (Open Source):** 220+ Frameworks inkl. CRA, NIS2. KI-Chat und LLM-gestütztes Mapping

Ableitung konkreter Security Anforderungen

Anforderungen

- **Ziel:** Aus Schutzbedarf, Compliance-Pflichten und Bedrohungen prüfbare Sicherheitsanforderungen ableiten
- **Input:** Schutzbedarfsanalyse, Compliance Anforderungen, User Stories, Kataloge (z.B. OWASP ASVS)
- **Output:** Konkrete, testbare Security Requirements und Akzeptanzkriterien im Backlog
- **Nötige Kompetenzen:** Requirements Engineering, Security-Fachwissen, Kenntnis von Anforderungskatalogen
- **Risiken:** Generische, nicht testbare Anforderungen, Überspezifikation, KI übernimmt Lücken der Eingaben

Beispiele

- **OWASP OpenCRE:** verknüpft Anforderungen aus ASVS, ISO, NIST. OpenCRE Chat beantwortet Fragen per LLM

Priorisierung von Security Anforderungen

Anforderungen

- **Ziel:** Security-Anforderungen nach Risiko, Aufwand und Pflichtcharakter in eine Umsetzungsreihenfolge bringen
- **Input:** Security Requirements, Risikobewertung, Compliance-Fristen, Aufwandsschätzungen, Release-Plan
- **Output:** Priorisiertes Security-Backlog mit nachvollziehbarer Begründung je Anforderung
- **Nötige Kompetenzen:** Risikobewertung, Produktmanagement, Einschätzung von Entwicklungsaufwänden
- **Risiken:** Pflichtanforderungen werden zurückgestellt, KI-Scores ohne Geschäftskontext, Scheingenauigkeit

Beispiele

- **CISO Assistant (Open Source):** priorisiert Maßnahmen nach Risiko. KI-Engine und MCP-Server.

Architektur

Architektur

Überblick ausgewählter Aktivitäten

- **Threat Modeling** und Attack-Surface Analysis
- **Security by Design** und Auswahl geeigneter Security Controls
- Bewertung von **Third-Party Komponenten** und externen Services
- **Dokumentation** von Architektur-Risiken

Threat Modeling

Architektur

- **Ziel:** Bedrohungen und Angriffsflächen der Architektur systematisch identifizieren und bewerten
- **Input:** Architekturdiagramme, Datenflüsse, Vertrauensgrenzen, Schutzbedarf, Security Requirements
- **Output:** Bedrohungsmodell mit Bedrohungen (z.B. STRIDE), Angriffspfaden und Gegenmaßnahmen
- **Nötige Kompetenzen:** Architekturverständnis, Angreiferperspektive, Methodik (STRIDE, Attack Trees)
- **Risiken:** Generische Bedrohungslisten, übersehene systemspezifische Angriffe, veraltete Modelle

Beispiele

- **OWASP Threat Dragon:** Datenflussdiagramme mit STRIDE. Offizielles KI Tool erzeugt Bedrohungen per LLM
- **AWS Threat Designer (awslabs):** STRIDE-Modell aus Architekturdiagrammen. CLI lokal nutzbar

Security by Design

Architektur

- **Ziel:** Sicherheitsprinzipien in Architekturentscheidungen verankern und passende Security Controls wählen
- **Input:** Bedrohungsmodell, Security Requirements, Architekturentwurf, Control-Kataloge (z.B. NIST 800-53)
- **Output:** Architekturentscheidungen (ADRs) mit zugeordneten Security Controls
- **Nötige Kompetenzen:** Sicherheitsarchitektur, Design-Patterns, Kenntnis von Control-Katalogen
- **Risiken:** Unpassende oder überdimensionierte Controls, Herstellerbias der KI, übersehene Privacy-Aspekte

Beispiele

- **Project CodeGuard (CoSAI/OASIS):** Regeln leiten KI-Agenten schon beim Design an

Third-Party Komponenten

Architektur

- **Ziel:** Risiken von Bibliotheken, Open-Source-Komponenten und externen Services vor dem Einsatz bewerten
- **Input:** Komponentenliste bzw. SBOM, Lizenzen, Schwachstellendatenbanken, Lieferanteninformationen
- **Output:** Freigabeentscheidung je Komponente mit Risikobewertung und Auflagen
- **Nötige Kompetenzen:** Supply-Chain-Security, Lizenzverständnis, Bewertung der Projektgesundheit
- **Risiken:** Von der KI erfundene Paketnamen (Slopsquatting), unvollständige SBOM, veraltete Schwachstellendaten

Beispiele

- **OSV-Scanner:** prüft Abhängigkeiten gegen OSV.dev. Integrierter MCP-Server für KI-Agenten
- **Trivy (Open Source):** erzeugt SBOMs und prüft Lizenzen. Offizieller MCP-Server

Dokumentation

Architektur

- **Ziel:** Architekturrisiken nachvollziehbar dokumentieren, bewerten und aktuell halten
- **Input:** Bedrohungsmodell, Architekturentscheidungen, Code und IaC, Ergebnisse von Reviews
- **Output:** Aktuelle Sicherheitsarchitektur-Dokumentation und Risikoregister mit Maßnahmen
- **Nötige Kompetenzen:** Technische Dokumentation, Risikomanagement, Architekturverständnis
- **Risiken:** Dokumentation weicht vom Ist-Stand ab, KI beschönigt Risiken oder erfindet Details

Beispiele

- **OWASP Threat Dragon:** Bedrohungsmodelle versioniert im Repo. KI Tool ergänzt Bedrohungen per LLM
- **CISO Assistant (Open Source):** Risikoregister mit Maßnahmen. KI-Engine und MCP-Server

Implementierung

Implementierung

Überblick ausgewählter Aktivitäten

- Anwendung von **Secure Coding Standards**
- **Security Code Reviews** und SAST
- **Prüfung sicherheitskritischer** APIs und Funktionen
- Automatisierte **Security Fixes** bzw. Fix-Vorschläge

Anwendung von Secure Coding Standards

Implementierung

- **Ziel:** Sicherheitsregeln beim Schreiben von Code konsistent anwenden, auch durch KI-Coding-Assistenten
- **Input:** Coding-Standards (z.B. OWASP, SEI CERT, MISRA), Quellcode, Projektkontext
- **Output:** Standardkonformer Code, Hinweise direkt in IDE und Pull Requests
- **Nötige Kompetenzen:** Programmiersprache, typische Schwachstellen (CWE), Kenntnis der Standards
- **Risiken:** KI-Assistenten erzeugen selbst unsicheren Code, Regeln fehlen im Kontext des Assistenten

Beispiele

- **Project CodeGuard (CoSAI/OASIS):** Security-Regeln und Skills für KI-Coding-Assistenten
- **OpenSSF Guide for AI Code Assistant Instructions:** Vorlagen für sichere Assistenten-Prompts

Security Code Reviews

Implementierung

- **Ziel:** Schwachstellen im Quellcode vor dem Merge finden und bewerten
- **Input:** Pull Requests, Quellcode, Architektur- und Bedrohungskontext, SAST-Regeln
- **Output:** Priorisierte Findings mit Begründung und Fix-Vorschlag direkt im Pull Request
- **Nötige Kompetenzen:** Secure Coding, Bewertung von Findings, Verständnis der Anwendungslogik
- **Risiken:** False Positives, übersehene Logikfehler, blindes Vertrauen in KI-Reviews

Beispiele

- **CoSAI Security Code Review Agent:** KI-Agent prüft Code anhand der Project-CodeGuard-Regeln

Prüfung sicherheitskritischen Codes

Implementierung

- **Ziel:** Sicherheitskritische APIs und Funktionen (z.B. Authentifizierung, Krypto) vertieft prüfen
- **Input:** Quellcode, API-Spezifikationen (z.B. OpenAPI), Datenflüsse, Bedrohungsmodell
- **Output:** Inventar kritischer Stellen mit Prüfergebnis und Maßnahmen
- **Nötige Kompetenzen:** Tiefes Security-Know-how (Krypto, AuthN/AuthZ), Code-Analyse, Fuzzing
- **Risiken:** Kritische Stellen nicht erkannt, Fehlbewertung komplexer Logik wie Autorisierung

Beispiele

- **OSS-Fuzz-Gen (Google):** LLM erzeugt Fuzzing-Harnesses für kritische Funktionen
- **OSS-CRS (OpenSSF):** führt KI-Systeme aus der DARPA AIxCC lokal zur Fehlersuche per Fuzzing aus

Automatisierte Security Fixes

Implementierung

- **Ziel:** Fix-Vorschläge für gefundene Schwachstellen automatisch erzeugen und die Behebung beschleunigen
- **Input:** Findings aus SAST und SCA, betroffener Code, Tests, Abhängigkeitsinformationen
- **Output:** Pull Requests mit Fix-Vorschlag, Begründung und Tests
- **Nötige Kompetenzen:** Code-Verständnis zur Bewertung der Fixes, Testautomatisierung
- **Risiken:** Fixes brechen Funktionalität, verdecken die Ursache oder führen neue Schwachstellen ein

Beispiele

- **Strix (Open Source):** KI-Agenten bestätigen Schwachstellen und erzeugen Patches
- **OSS-CRS (OpenSSF):** KI-Patch-Agenten aus der DARPA AIxCC erzeugen und prüfen Fixes

Tests

Tests

Überblick ausgewählter Aktivitäten

- Durchführung von dynamischen Anwendungssicherheitstests (DAST)
- Schwachstellen-Scans
- Penetrationstests
- Validierung gefundener Schwachstellen

Dynamische Anwendungssicherheitstests

Tests

- **Ziel:** Laufende Anwendung von außen auf ausnutzbare Schwachstellen testen
- **Input:** Testumgebung, URLs und API-Spezifikationen, Test-Accounts, Authentifizierungsabläufe
- **Output:** Findings mit Request/Response-Nachweis und Schweregrad
- **Nötige Kompetenzen:** Web- und API-Security, Konfiguration von Scannern, Bewertung von Findings
- **Risiken:** Lückenhafte Abdeckung (Login, Single-Page-Apps), beschädigte Testdaten, False Positives

Beispiele

- **ZAP (Open Source):** DAST-Scanner. MCP-Server und LLM-Support-Add-on steuern Scans per KI
- **Strix (Open Source):** KI-Agenten testen die laufende Anwendung dynamisch

Schwachstellen-Scans

Tests

- **Ziel:** Bekannte Schwachstellen in Abhängigkeiten, Containern und Infrastruktur kontinuierlich erkennen
- **Input:** SBOM, Container-Images, Abhängigkeiten, Infrastruktur, Schwachstellendatenbanken (CVE, OSV)
- **Output:** Schwachstellenliste mit Schweregrad, Erreichbarkeit und verfügbarer Fix-Version
- **Nötige Kompetenzen:** Schwachstellenmanagement, Einordnung von CVSS und EPSS, Infrastrukturwissen
- **Risiken:** Alarmflut, nicht erreichbare Schwachstellen binden Kapazität, Lücken im Scan-Umfang

Beispiele

- **Trivy (Open Source):** scannt Container, Repos und IaC. Offizieller MCP-Server
- **OSV-Scanner:** prüft Abhängigkeiten gegen OSV.dev. Integrierter MCP-Server

Penetrationstests

Tests

- **Ziel:** Angriffe realistisch simulieren, um ausnutzbare Schwachstellen und Angriffsketten aufzudecken
- **Input:** Scope und Freigabe, Architektur- und Zugangsinformationen, Testumgebung, Bedrohungsmodell
- **Output:** Pentest-Bericht mit Angriffsketten, Nachweisen und priorisierten Empfehlungen
- **Nötige Kompetenzen:** Offensive Security, Kreativität, Kenntnis der rechtlichen Rahmenbedingungen
- **Risiken:** Schäden durch autonome Agenten, Verletzung des Scopes, trügerischer Eindruck von Vollständigkeit

Beispiele

- **Strix (Open Source):** autonome KI-Agenten führen Pentests durch und belegen Funde mit PoCs
- **PentestGPT (Open Source, MIT):** LLM-Assistent, der durch die Schritte eines Pentests führt

Validierung von Schwachstellen

Tests

- **Ziel:** Findings auf Ausnutzbarkeit prüfen, False Positives aussortieren und das Risiko bestätigen
- **Input:** Findings aus SAST, DAST, Scans und Pentests, Quellcode, Laufzeit- und Deploymentkontext
- **Output:** Bestätigte, bewertete Schwachstellen mit Nachweis, Priorität und VEX-Status
- **Nötige Kompetenzen:** Exploit-Verständnis, Kontextwissen zur Anwendung, Risikobewertung
- **Risiken:** Echte Schwachstellen als False Positive verworfen, KI-Begründungen nicht nachprüfbar

Beispiele

- **Strix (Open Source):** bestätigt Findings durch funktionierende Proof-of-Concept-Exploits
- **govulncheck (Go-Projekt):** prüft Erreichbarkeit. Per gopls-MCP-Server für KI-Agenten nutzbar

Zusammenfassung

1. **Erst Prozess und Kompetenz, dann die KI**
   KI setzt an definierten Aktivitäten an. Die Qualität wird manuell überprüft.

2. **Werkzeuge liefern die Evidenz, die KI liefert den Kontext**
   KI erklärt, priorisiert und berät basierend auf existierenden Tools.

3. **Der Mensch entscheidet**
   Freigabe, Risikoakzeptanz und Verantwortung bleiben beim Team.

4. **Klein und offen anfangen**
   Viele Werkzeuge sind Open Source. Ein LLM darüber ist ein günstiger Einstieg.

fortiss Potenzialanalyse

Wo lohnt sich KI in Ihrer Software-Entwicklung?

1. **Kick-off & Erstaufnahme**
2. **Vertiefung & Bewertung**
3. **Bericht & Abschlussworkshop**

Ergebnis: priorisierte Maßnahmen

fortiss.org/angebote/potenzialanalyse

Vielen Dank

für Ihre Aufmerksamkeit

Wir freuen uns auf den gemeinsamen Austausch mit Ihnen.

Kontaktieren Sie uns!
info@fortiss.org
www.fortiss.org

Folgen Sie uns!

fortiss © 2026



Florian Angermeir
Wissenschaftler Requirements Engineering
+49 89 3603522 279
angermeir@fortiss.org



fortiss Newsletter:
Bleiben Sie auf dem Laufenden zu aktuellen Forschungs- und Innovationsthemen, Infos zu Events und Angeboten.
Jetzt anmelden!