Code responsibly with generative AI in Java (CRWGAIJ)

ID CRWGAIJ **Price** CHF 2,250.—(excl. VAT) **Duration** 3 days

Who should attend

Java developers using Copilot or other GenAI tools

Prerequisites

OWASP, SEI CERT, CWE and Fortify Taxonomy

Course Objectives

- Understanding the essentials of responsible AI
- Getting familiar with essential cyber security concepts
- Understanding how cryptography supports security
- Learning how to use cryptographic APIs correctly in Java
- Understanding Web application security issues
- Detailed analysis of the OWASP Top Ten elements
- Putting Web application security in the context of Java
- Going beyond the low hanging fruits
- Managing vulnerabilities in third party components
- All this put into the context of GitHub Copilot

Course Content

Day 1

Coding responsibly with GenAI

- What is responsible AI?
- What is security?
- Threat and risk
- Cyber security threat types – the CIA triad
- Consequences of insecure software
- Security and responsible AI in software development
- GenAI tools in coding: Copilot, Codeium and others
- The OWASP Top Ten from Copilot's perspective
    - The OWASP Top Ten 2021
        - A01 – Broken Access Control
            - Access control basics
            - Case study – Broken authn/authz in Apache OFBiz
            - Confused deputy
            - Insecure direct object reference (IDOR)
            - Path traversal
            - Lab – Insecure Direct Object Reference
            - Path traversal best practices
            - Lab – Experimenting with path traversal in Copilot
            - Authorization bypass through user-controlled keys
            - Case study – Remote takeover of Nexx garage doors and alarms
            - Lab – Horizontal authorization (exploring with Copilot)
            - File upload
                - Unrestricted file upload
                - Good practices
                - Lab – Unrestricted file upload (exploring with Copilot)
                - Case study – File upload vulnerability in Netflix Genie
        - A02 – Cryptographic Failures
            - Cryptography for developers
            - Cryptography basics
            - Java Cryptographic Architecture (JCA) in brief
            - Elementary algorithms
            - Hashing
                - Hashing basics
                - Hashing in Java
                - Lab – Hashing in JCA (exploring with Copilot)
            - Random number generation
                - Pseudo random number generators (PRNGs)
                - Cryptographically secure PRNGs
                - Weak and strong PRNGs in Java
                - Lab – Using random numbers in Java (exploring with Copilot)
                - Case study – Equifax credit account freeze
            - Confidentiality protection
                - Symmetric encryption
                    - Block ciphers
                    - Modes of operation
                    - Modes of operation and IV – best

practices
        - Symmetric encryption in Java
        - Symmetric encryption in Java with streams
        - Lab – Symmetric encryption in JCA (exploring with Copilot)
    - Asymmetric encryption
    - Combining symmetric and asymmetric algorithms
    - Key exchange and agreement
        - Key exchange
        - Diffie-Hellman key agreement algorithm
        - Key exchange pitfalls and best practices

Day 2

The OWASP Top Ten from Copilot's perspective

- A03 – Injection
    - Injection principles
    - Injection attacks
        - SQL injection
            - SQL injection basics
            - Lab – SQL injection
            - Attack techniques
                - Content-based blind SQL injection
                - Time-based blind SQL injection
            - SQL injection best practices
            - Input validation
            - Parameterized queries
            - Lab – Using prepared statements
            - Lab – Experimenting with SQL injection in Copilot
            - Database defense in depth
            - Case study – SQL injection in Fortra FileCatalyst
        - Code injection
            - OS command injection
            - OS command injection best practices
            - Using Runtime.exec()
            - Case study – Shellshock
            - Lab – Shellshock
            - Case study – Command injection in VMware Aria
        - HTML injection – Cross-site scripting (XSS)
            - Cross-site scripting basics
            - Cross-site scripting types
                - Persistent cross-site scripting
                - Reflected cross-site scripting
                - Client-side (DOM-based) cross-site scripting
            - Lab – Stored XSS
            - Lab – Reflected XSS
            - XSS protection best practices
            - Protection principles – escaping
            - XSS protection APIs in Java
            - Lab – XSS fix / stored (exploring with Copilot)
            - Lab – XSS fix / reflected (exploring with Copilot)
            - Additional protection layers – defense in depth
            - Case study – XSS vulnerabilities in DrayTek Vigor routers
    - A04 – Insecure Design
        - The STRIDE model of threats
        - Secure design principles of Saltzer and Schroeder
            - Economy of mechanism
            - Fail-safe defaults
            - Complete mediation
            - Open design
            - Separation of privilege
            - Least privilege
            - Least common mechanism
            - Psychological acceptability
        - Client-side security
        - Frame sandboxing
        - Cross-Frame Scripting (XFS) attacks
        - Lab – Clickjacking
        - Clickjacking beyond hijacking a click
        - Clickjacking protection best practices
        - Lab – Using CSP to prevent clickjacking (exploring with Copilot)
    - A05 – Security Misconfiguration
        - Configuration principles
        - XML entities
            - DTD and the entities
            - Entity expansion
            - External Entity Attack (XXE)
            - File inclusion with external entities
            - Server-Side Request Forgery with external entities
            - Lab – External entity attack
            - Preventing XXE
            - Lab – Prohibiting DTD
            - Case study – XXE vulnerability in Ivanti products
            - Lab – Experimenting with XXE in

Copilot

Day 3

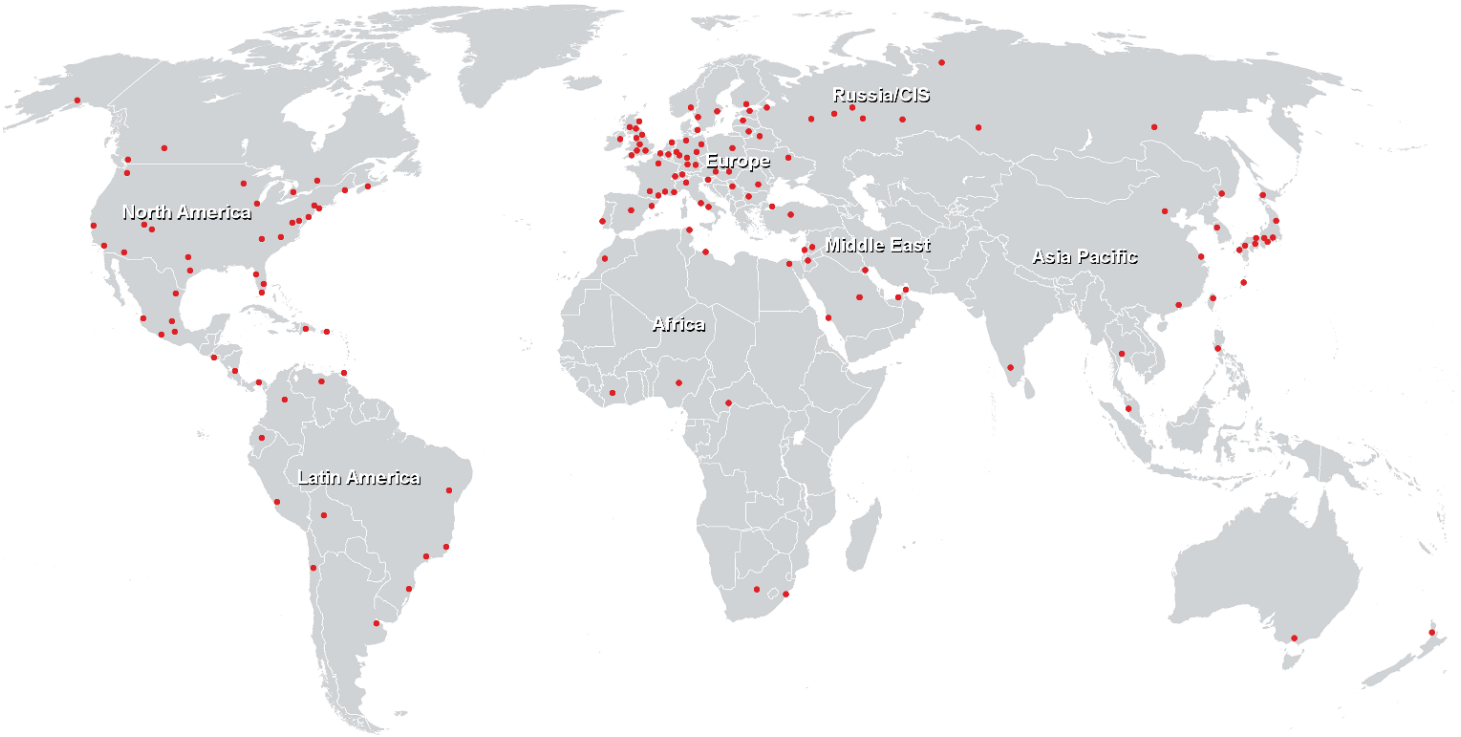
The OWASP Top Ten from Copilot's perspective

- A06 – Vulnerable and Outdated Components
  - Using vulnerable components
  - Untrusted functionality import
  - Case study – The Polyfill.io supply chain attack
  - Vulnerability management
  - Lab – Finding vulnerabilities in third-party components
  - Security of AI generated code
  - Practical attacks against code generation tools
  - Dependency hallucination via generative AI
  - Case study – A history of GitHub Copilot weaknesses (up to mid 2024)
- A07 – Identification and Authentication Failures
  - Authentication
    - Authentication basics
    - Multi-factor authentication (MFA)
    - Case study – The InfinityGauntlet attack
  - Password management
    - Inbound password management
    - Storing account passwords
    - Lab – Is just hashing passwords enough?
    - Dictionary attacks and brute forcing
    - Salting
    - Adaptive hash functions for password storage
    - Lab – Using adaptive hash functions in JCA
    - Lab – Using adaptive hash functions in Copilot
    - Password policy
    - NIST authenticator requirements for memorized secrets
- A08 – Software and Data Integrity Failures
  - Integrity protection
    - Message Authentication Code (MAC)
    - Calculating MAC in Java
    - Lab – Calculating MAC in JCA
  - Digital signature
    - Elliptic Curve Cryptography
    - ECC basics
    - Digital signature with ECC
    - Digital signature in Java
    - Lab – Digital signature with ECDSA in JCA
  - Subresource integrity
    - Importing JavaScript
    - Lab – Importing JavaScript (exploring with Copilot)
    - Case study – The British Airways data breach
  - Insecure deserialization
    - Serialization and deserialization challenges
    - Integrity – deserializing untrusted streams
    - Integrity – deserialization best practices
    - Look ahead deserialization
    - Property Oriented Programming (POP)
    - Creating a POP payload
    - Lab – Creating a POP payload
    - Lab – Using the POP payload
    - Case study – Deserialization RCEs in NextGen Mirth Connect
- A09 – Security Logging and Monitoring Failures
  - Logging and monitoring principles
  - Log forging
  - Log forging – best practices
  - Case study – Log interpolation in log4j
  - Case study – The Log4Shell vulnerability (CVE-2021-44228)
  - Case study – Log4Shell follow-ups (CVE-2021-45046, CVE-2021-45105)
  - Lab – Log4Shell
- A10 – Server-side Request Forgery (SSRF)
  - Server-side Request Forgery (SSRF)
  - Case study – SSRF in Ivanti Connect Secure
- Wrap up
  - Secure coding principles

- Principles of robust programming by Matt Bishop
- And now what?
- Software security sources and further reading
- Java resources
- Responsible AI principles in software development
- Generative AI – Resources and additional guidance

Code responsibly with generative AI in Java (CRWGAIJ)

Training Centres worldwide



Fast Lane Institute for Knowledge Transfer (Switzerland) AG

Husacherstrasse 3
CH-8304 Wallisellen
Tel. +41 44 832 50 80

info@flane.ch, https://www.flane.ch