Code responsibly with generative AI in C# (desktop applications) (CRWGAIC)

ID CRWGAIC **Prix** CHF 2 250,– (Hors Taxe) **Durée** 3 jours

A qui s'adresse cette formation

C# developers using Copilot or other GenAI tools

Pré-requis

General C# development

Objectifs

- Understanding the essentials of responsible AI
- Getting familiar with essential cyber security concepts
- Input validation approaches and principles
- Identify vulnerabilities and their consequences
- Learn the security best practices in C#
- Correctly implementing various security features
- Managing vulnerabilities in third party components
- Understanding how cryptography supports security
- Learning how to use cryptographic APIs correctly in C#
- All this put into the context of GitHub Copilot

Contenu

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
- Input validation
  - Input validation principles
  - Denylists and allowlists
  - What to validate – the attack surface
  - Where to validate – defense in depth
  - When to validate – validation vs transformations
- Injection
  - Code injection
  - OS command injection
  - Lab – Command injection
  - OS command injection best practices
  - Avoiding command injection with the right APIs
  - Lab – Command injection best practices
  - Lab – Experimenting with command injection in Copilot
  - Case study – Command injection in Ruckus
- Integer handling problems
  - Representing signed numbers
  - Integer visualization
  - Integer overflow
  - Lab – Integer overflow
  - Signed / unsigned confusion
  - Case study – The Stockholm Stock Exchange
  - Lab – Signed / unsigned confusion
  - Lab – Experimenting with signed / unsigned confusion in Copilot
  - Integer truncation
  - Best practices
  - Upcasting
  - Precondition testing
  - Postcondition testing
  - Integer handling in C#
  - Lab – Checked arithmetics
  - Lab – Experimenting with integer overflow in Copilot
- Files and streams
  - Path traversal
  - Lab – Path traversal
  - Additional challenges in Windows
  - Case study – File spoofing in WinRAR
  - Path traversal best practices
  - Lab – Path canonicalization
  - Lab – Experimenting with path traversal in Copilot

Day 2

Input validation

- Unsafe reflection
  - Reflection without validation
  - Lab – Unsafe reflection
  - Lab – Experimenting with unsafe reflection in

Copilot
- Unsafe native code
  - Native code dependence
  - Lab – Unsafe native code
  - Best practices for dealing with native code
- Security features
  - Authentication
    - Authentication basics
    - Multi-factor authentication (MFA)
    - Case study – The InfinityGauntlet attack
    - Time-based One Time Passwords (TOTP)
  - Password management
    - Inbound password management
    - Storing account passwords
    - Password in transit
    - Lab – Is just hashing passwords enough?
    - Dictionary attacks and brute forcing
    - Salting
    - Adaptive hash functions for password storage
    - Lab – Using adaptive hash functions in C#
    - Lab – Using adaptive hash functions in Copilot
    - Case study – Veeam missing authentication and cleartext password storage
    - Password policy
    - NIST authenticator requirements for memorized secrets
    - Password database migration
    - Hard coded passwords
    - Best practices
    - Lab – Hardcoded password
  - Protecting sensitive information in memory
    - Challenges in protecting memory
    - Case study – Microsoft secret key theft via dump files
    - Storing sensitive data in memory
    - Case study – KeePass password leakage via strings
  - Information exposure
    - Exposure through extracted data and aggregation
    - Case study – Strava data exposure
  - Platform security
    - .NET platform security
    - Protecting .NET code and applications
    - Code signing
  - Denial of service
    - Flooding
    - Resource exhaustion
    - Algorithmic complexity issues
    - Regular expression denial of service (ReDoS)
    - Lab – ReDoS
    - Lab – Experimenting with ReDoS in Copilot
    - Dealing with ReDoS
  - Using vulnerable components
    - 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)

Day 3

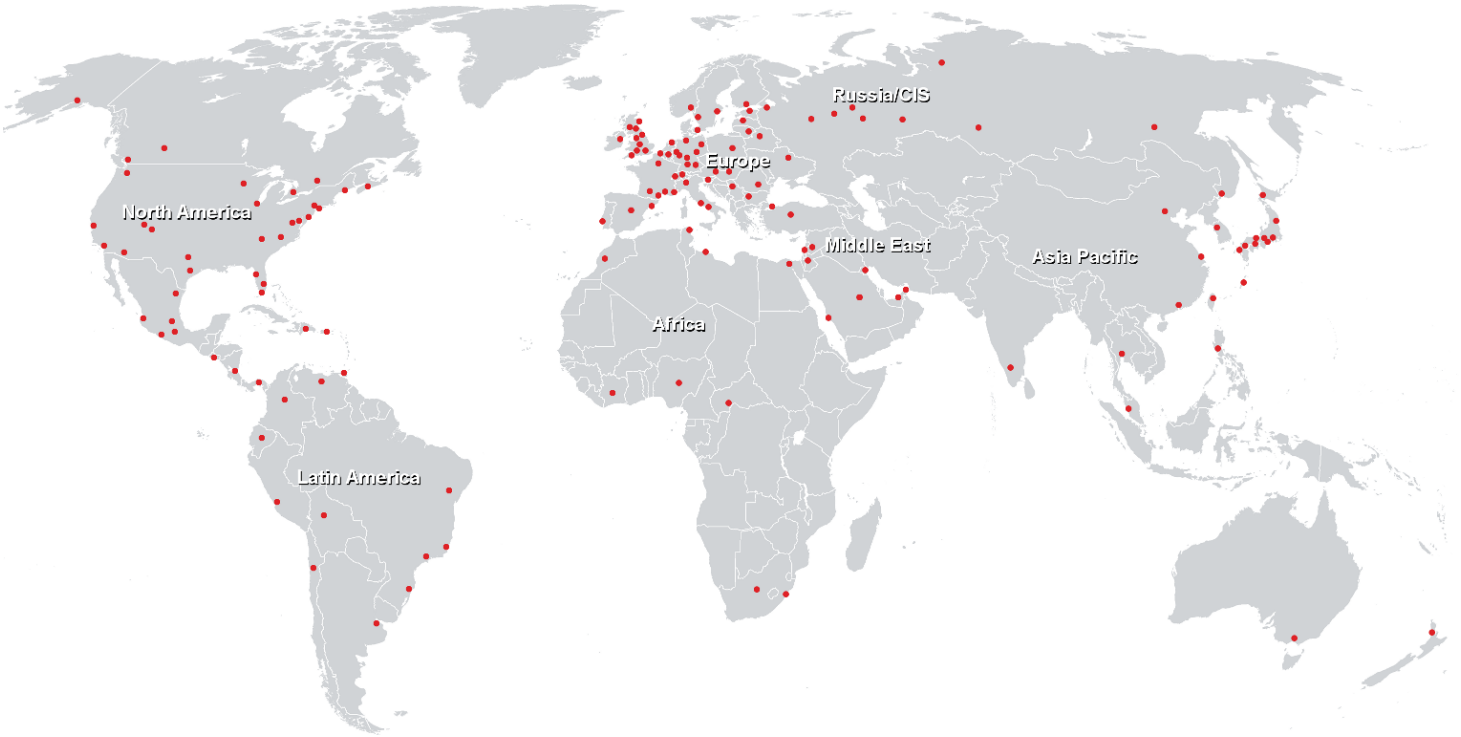
Cryptography for developers

- Cryptography basics
- Crypto APIs in C#
- Elementary algorithms
- Hashing
  - Hashing basics
  - Hashing in C#
  - Lab – Hashing in C# (exploring with Copilot)
- Random number generation
  - Pseudo random number generators (PRNGs)
  - Cryptographically secure PRNGs
  - Weak and strong PRNGs
  - Using random numbers in C#
  - Lab – Using random numbers in C# (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 C#
    - Symmetric encryption in C# with streams
    - Lab – Symmetric encryption in C# (exploring with Copilot)
    - Case study – Padding oracle used in RCE against Citrix ShareFile
  - Asymmetric encryption
    - The RSA algorithm
    - RSA in C#
    - Combining symmetric and asymmetric algorithms
  - Key exchange and agreement
    - Key exchange

        - Diffie-Hellman key agreement algorithm
        - Key exchange pitfalls and best practices
    - Integrity protection
        - Message Authentication Code (MAC)
        - Calculating HMAC in C#
        - Lab – Calculating MAC in C#
        - Digital signature
            - Digital signature with RSA
            - Elliptic Curve Cryptography
            - ECC basics
            - Digital signature with ECC
            - Digital signature in C#
            - Lab – Digital signature with ECDSA in C#
    - Common software security weaknesses
        - Code quality
            - Code quality and security
        - Data handling
            - Initialization and cleanup
            - Class initialization cycles
            - Lab – Initialization cycles (exploring with Copilot)
        - Object oriented programming pitfalls
            - Inheritance and overriding
            - Mutability
            - Lab – Mutable object (exploring with Copilot)
        - Serialization
            - 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 RCE in Veeam
    - Wrap up
        - Secure coding principles
        - Principles of robust programming by Matt Bishop
        - Secure design principles of Saltzer and Schroeder
        - And now what?
        - Software security sources and further reading
        - .NET and C# resources
        - Responsible AI principles in software development
        - Generative AI – Resources and additional guidance

Code responsibly with generative AI in C# (desktop applications) (CRWGAIC)

Centres de formation dans le monde entier





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