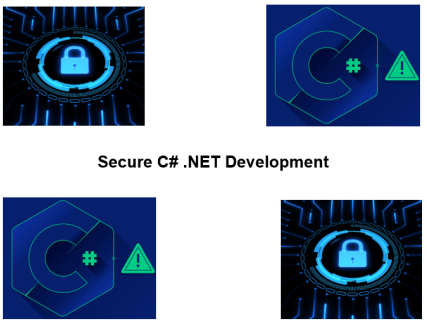
Security in C# .NET Development

Duration: 3 days **Price: € 2250**

Open Schedule



Audience Course Security in C# .NET Development
The course Security in C# .NET Development is intended for C# Developers who want to learn how to protect C# applications against the many security risks.

Prerequisites Course Security in C# .NET Development
To participate in this course, knowledge of and experience with C# and the .NET Platform is required.

Realization Training Security in C# .NET Development
The course Security in C# .NET Development is a hands-on course. Theory explanation based on demos and presentations is interchanged with practice based on exercises.

Certificate course Security in C# .NET Development
After successfully completing the training, attendants will receive a certificate of participation in the course Security in C# .NET Development.

Content Course Security in C# .NET Development

The course Security in C# .NET Development provides C# developers with the essential knowledge and practical skills to effectively tackle security problems with web applications. Common security issues, as described in the OWASP Top Ten, are covered as well as best practices for dealing with security challenges in .NET C# code.

Intro Secure Coding
The course Security in C# .NET Development starts with an overview of the application security landscape, including common attack vectors and potential risks when developing C# code.

Broken Access Control
The course proceed with a discussion of how to prevent vulnerabilities as a result of broken access control. Attention is paid to Role Based Access Control (RABC), the correct implementation of session management and Access Control Lists.

Cryptographic Failures
Cryptographic weaknesses such as flawed encryption algorithms and incorrect use of cryptographic functions are also treated.

Injection Flaws
Then the dangers of injection are covered, such as SQL injection and cross-site scripting (XSS) and cross-site request forgery (CSRF). Secure coding practices to prevent injection are also explained, such as input validation, output encoding and parameterized queries.

Insecure Design
Insecure design is also on the program of the course Security in C# .NET Development with inadequate input validation, incorrect error handling and insecure authentication.

Misconfiguration Failures
Then it is highlighted how configuration errors can lead to security risks, such as the use of default settings and insufficient protection of sensitive data.

NuGet Packages
And attention is paid to the risks of external NuGet packages, how to assess NuGet packages and best practices for safely integrating NuGet packages.

Logging and Monitoring
Finally it is discussed how logging and monitoring can improve the security of C# applications. Attention is paid to the importance of logging and monitoring for detecting and responding to security incidents.

Modules Course Security in C# .NET Development

Module 1 : Intro Secure Coding	Module 2 : Broken Access Control	Module 3 : Cryptographic Failures
Secure Coding practices Never trusting Input SQL injection and NoSQL injection OS command injection Session Fixation Cross Site Scripting and CSRF Sensitive Data Exposure Insecure Deserialization Security Misconfiguration Using Unsafe Components	Implement Proper Authentication Broken Authentication Role Based Access Control (RBAC) Implement Use Session Management Session Timeout Access Control Lists (ACLs) Principle of Least Privilege (PoLP) URL and API Authorization Error Handling Regular Security Testing	Sensitive Data Exposure Weak Key Generation Insecure Storage of Keys Using Outdated Algorithms Hardcoding Secrets Insufficient Key Management Avoid Homegrown Cryptography Verify Signatures Side-Channel Attacks Lack of Forward Secrecy
Module 4 : Injection Flaws	**Module 5 : Insecure Design**	**Module 6 : Misconfiguration Failures**
SQL Injection (SQLi) Cross-Site Scripting (XSS) Command Injection XML Injection LDAP Injection XPath Injection SSI Injection Object Injection Template Injection CRLF Injection	Inadequate Authentication Inadequate Authorization Lack of Input Validation Excessive Data Exposure Insecure Session Management Hardcoding Secrets Insufficient Logging and Monitoring Insecure Data Storage Cross-Site Request Forgery Improper Error Handling	Improper Access Control Unsecured APIs Open Database Ports Default Credentials Unused or Unnecessary Features Weak Password Policies Missing Security Updates Improper File Permissions Insecure Session Management Excessive Error Detail
Module 7 : NuGet Packages	**Module 8 : Authentication Mistakes**	**Module 9 : Logging and Monitoring**
Known Vulnerabilities Malicious Packages License Compliance Misconfigured Packages Dependency Chains Cryptographic Weaknesses Data Privacy and Compliance Resource Exhaustion Insecure Configuration Defaults	Weak Password Policies No Account Lockout Mechanism Inadequate Password Storage Hardcoding Credentials Lack of Multi-Factor Authentication (MFA) Insufficient Session Management Missing CAPTCHA or Rate Limiting Overly Permissive Access Controls Improper Handling Forgotten Passwords	Insufficient Logging Lack of Centralized Logging Logging Sensitive Information Inadequate Log Retention Unencrypted Logging Insufficient Access Controls Failure to Monitor Logs in Real-Time No Alerts or Notifications Ignoring Anomalous Activity