



# Secure Coding and Vulnerability Remediation Training Course

15 – 19 March 2027  
Athens

## Secure Coding and Vulnerability Remediation Training Course

**Ref:** 763\_158082 **Date:** 15 – 19 March 2027 **Location:** Athens **Fees:** 6700 Euro

### Overview

Secure Coding and Vulnerability Remediation Training Course is a five-day intermediate course for developers, application engineers, technical leads, quality engineers, and security practitioners who leave with a Secure Coding Remediation Pack. Participants apply threat-aware coding decisions to input handling, authentication, sessions, access, secrets, errors, dependencies, review, and testing. The course uses OWASP checklists, NIST SP 800-218 SSDF context, code-review prompts, and repair exercises. Agile Leaders Training Center develops secure coding and vulnerability remediation practice.

### Who Should Attend

- Software development functions responsible for implementing application features
- Application engineering functions responsible for code quality and integration
- Technical leadership functions responsible for review standards and remediation priorities
- Quality engineering functions responsible for security-focused testing
- Security functions responsible for developer guidance and vulnerability validation

The course assumes participants can read and change application code, and leaves out penetration testing operations, infrastructure hardening, enterprise lifecycle governance, and certification preparation.

### Departments and Industries

The course supports departments and industries that build or maintain software applications.

- Software engineering, application development, and quality assurance
- Cybersecurity, product security, and technology risk
- Financial services and digital commerce
- Healthcare and education technology
- Telecommunications, logistics, and digital platforms

### Learning Objectives

By the end of this course, participants will be able to:

- Apply threat-aware secure coding decisions
- Use input validation and output encoding controls
- Analyze authentication, session, and access-control weaknesses
- Evaluate secrets, errors, files, and dependency risks
- Use code review and security testing checklists



- Build a Secure Coding Remediation Pack

## **Course Agenda**

### **Day 1: Frame Secure Coding Decisions**

- NIST SP 800-218 SSDF Practice Context Map
- OWASP Secure Coding Control Checklist
- Application Trust Boundary and Data Flow Sketch
- Threat-to-Code Decision Matrix
- Secure Coding Requirement Record

### **Day 2: Control Inputs and Outputs**

- Server-Side Input Validation Method
- Allowlist and Canonicalization Decision Guide
- Context-Specific Output Encoding Matrix
- Parameterized Query and Injection Prevention Pattern
- File Upload Validation and Storage Checklist

### **Day 3: Protect Identity and Access**

- Authentication Flow Review Worksheet
- Session Token Lifecycle Checklist
- Authorization and Object Access Decision Table
- Credential and Secret Handling Register
- Cryptographic Use and Key Responsibility Map

### **Day 4: Review and Remediate Code**

- Error Handling and Information Exposure Checklist
- Security Configuration Review Card
- Dependency and Component Risk Register
- Secure Code Review Comment Template
- Remediation Priority and Verification Matrix

### **Day 5: Practice Secure Code Repair**

- Exercise: Diagnose Input and Output Vulnerabilities
- Exercise: Repair Authentication and Session Weaknesses
- Exercise: Correct Access, Secret, and Error Controls
- Exercise: Verify Dependency and Code Review Findings
- Capstone: Secure Coding Remediation Pack

## Practical Exercises

The course uses suggested activities based on financial services, digital commerce, healthcare, and platform applications.

- Suggested activity: trace untrusted data through a sample flow and select validation and encoding controls.
- Suggested activity: review authentication, session, authorization, secret, and error-handling code fragments.
- Suggested activity: classify dependency and code-review findings by exploitability, impact, and repair effort.
- Suggested activity: assemble corrected patterns, review comments, tests, and evidence into a remediation pack.

## FAQs

### Who suits secure coding training, and what does it assume?

Developers, application engineers, technical leads, quality engineers, and security practitioners suit secure coding training; it assumes the ability to read and change application code.

### How does secure coding training differ from application security governance training?

Secure coding training concentrates on code-level decisions, review, testing, and repair, while application security governance training focuses on policies, lifecycle ownership, assurance processes, and portfolio oversight.

### How should secure coding prevent injection vulnerabilities?

Secure coding should combine server-side validation, canonicalization, allowlists, parameterized interfaces, context-specific encoding, constrained privileges, safe error handling, and tests that verify rejected and accepted inputs.

### What should a secure code review record contain?

A secure code review record should contain the affected component, data flow, weakness, evidence, risk reasoning, repair recommendation, owner, test case, verification result, limitations, and follow-up decision.

### How should developers prioritize vulnerability remediation?

Developers should prioritize remediation using exploitability, exposure, business impact, affected data, privilege, reachability, dependency context, available compensating controls, repair complexity, and verification needs.



## Conclusion

Participants take back a Secure Coding Remediation Pack connecting coding decisions, review findings, repair patterns, tests, and verification evidence. It changes isolated vulnerability fixes into a repeatable developer workflow. The pack supports clearer review comments, risk-based priorities, consistent repairs, peer verification, and lessons that reduce recurrence.