Faysal Aziz

Understanding Secure Coding and Threat Models

Elevate Your Code Security with Industry-Proven Best Practices - 1 Day

FA-0450 · 1 Day

Software Development · Cybersecurity

Introduction

This one-day training is designed to equip developers, security professionals, and team leads with critical knowledge on how to write secure code by focusing on the OWASP Top 10 vulnerabilities, secure coding practices, and key threat models such as STRIDE and DREAD. Whether you're just getting started or looking to reinforce your secure coding techniques, this course will guide you through real-world examples, best practices, and the potential consequences of security oversights.

With over 30 years of industry experience, our instructor will teach you actionable, industry-demanded content—moving beyond just theoretical academic concepts. You'll learn how to fortify your code, prevent vulnerabilities, and stay compliant with modern security standards.

Learning outcomes

By the end of this training, participants will:

- Understand each item in the OWASP Top 10 2021 list and the associated attack vectors.
- Grasp the principles and importance of secure coding, along with its impact on cybersecurity.
- Learn how to implement secure coding practices across real-world applications.
- Recognize common pitfalls and mistakes that lead to insecure code.
- Use checklists and versioning control to ensure secure coding practices are consistently applied.
- Understand and apply STRIDE and DREAD threat models for effective risk management.

Prerequisites

- Basic knowledge of software development and programming concepts.
- Familiarity with web applications and general cybersecurity principles is helpful but not mandatory.

Training outline

1. Overview of OWASP Top 10 2021

Introduction to OWASP and its mission

1. Overview of OWASP's role in web security.
2. Key reasons to focus on the OWASP Top 10 list for secure coding.

OWASP Top 10 2021 Vulnerabilities and Associated Attacks

Broken Access Control

1. Types of attacks: Elevation of privilege, unauthorized access.

Cryptographic Failures

1. Attacks: Man-in-the-middle (MITM), weak encryption exploits.

Injection

1. Examples: SQL injection, command injection.

Insecure Design

1. Risks of insecure architectural patterns.

Security Misconfiguration

1. Real-world examples: Unpatched systems, exposed configurations.

Vulnerable and Outdated Components

1. Risks: Exploiting known vulnerabilities in outdated libraries.

Identification and Authentication Failures

1. Examples: Broken authentication, credential stuffing.

Software and Data Integrity Failures

1. Attacks: Exploiting software supply chain vulnerabilities.

Security Logging and Monitoring Failures

1. Example: Lack of detection of intrusions or attacks.

Server-Side Request Forgery (SSRF)

1. Impact: Unauthorized internal network access.

2. What is Secure Coding?

Definition of Secure Coding

1. Ensuring code is developed with security considerations from the start.

Importance of Secure Coding

1. Protects against breaches, maintains system integrity.
2. Consequences of neglecting secure coding: Financial loss, reputation damage, regulatory penalties.

3. Compliance for Secure Coding

Security Standards and Regulatory Compliance

1. GDPR, HIPAA, PCI-DSS: How secure coding fits into these frameworks.
2. Industry guidelines for maintaining secure code.

   Internal vs. External Compliance

   1. Ensuring alignment with both internal policies and industry regulations.

4. Practicing Secure Coding with OWASP Top 10

Good and Bad Examples for Each OWASP Top 10 Item

Broken Access Control:

1. Good: Role-based access control (RBAC) implemented.
2. Bad: Exposed sensitive APIs without proper authentication.

Cryptographic Failures:

1. Good: Use of strong, up-to-date encryption protocols (e.g., AES-256).
2. Bad: Use of deprecated algorithms like MD5 or SHA-1.

Injection:

1. Good: Parameterized queries and ORM usage.
2. Bad: Unsanitized user input leading to SQL injection.

4. (Continue similar structure for the remaining OWASP Top 10 vulnerabilities.)

5. Common Mistakes in Secure Coding

Misunderstanding of Input Validation

1. Failure to sanitize input leads to injection vulnerabilities.

Incorrect Use of Cryptography

1. Using weak or outdated cryptographic algorithms.

Ignoring Error Handling

1. Leaking sensitive information through unhandled exceptions.

Overly Permissive Security Configurations

1. Default passwords, excessive permissions.

6. Secure Coding Checklist

Comprehensive Checklist to Ensure Secure Coding Practices

1. Input validation, authentication mechanisms, proper error handling, encryption standards.
2. Versioning control and consistent code reviews.
3. Automated testing tools for vulnerability scans.

7. Versioning Control and Frequency of Secure Coding

How Often to Review and Update Code for Security

1. Regular reviews aligned with development sprints.
2. Best practices for version control (e.g., Git), documenting security changes.

Integrating Secure Coding into the Development Lifecycle

1. Shift-left approach: Addressing security early in the SDLC.
2. Continuous monitoring and updates to stay ahead of emerging threats.

8. Threat Modeling Techniques: STRIDE and DREAD

STRIDE Model

Explanation and Application

1. Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege.

Real-world Use Cases

1. Applying STRIDE to various software systems for threat identification.

DREAD Threat Risk Ranking Model

Overview of DREAD Components

1. Damage, Reproducibility, Exploitability, Affected Users, Discoverability.

Risk Assessment with DREAD

1. Quantifying and ranking potential security threats.

Practical Examples

1. Using DREAD to prioritize security patches and mitigate risks.

This comprehensive one-day training will provide you with actionable insights and practical techniques to integrate secure coding practices into your everyday workflow. You'll leave with the confidence to minimize vulnerabilities, comply with industry standards, and create robust, secure applications.

© 2026 Faysal Aziz · Kuala Lumpur