

**Security Testing Java Web Applications**

**Course Number:** SEC-138  
**Duration:** 3 days

**Overview**

This Security Testing Java Web Applications training teaches attendees how to pinpoint and fix web application security issues found in Java code. Students learn how to find potential problems during testing and how to implement security testing methodology, techniques, and tools into their Java programming. References to A1, A2, etc. in the outline reference specific vulnerabilities in the [OWASP Top 10](https://owasp.org/www-project-top-ten/).

**Note:** To ensure ample one-on-one engagement with the instructor, this class is capped at 12 people, overriding Accelebrate’s default cap of 15.

**Prerequisites**

All students must have general Java, web development, and testing experience.

**Materials**

All Java security training attendees receive comprehensive courseware.

**Software Needed on Each Student PC**

Attendees will not need to install any software on their computer for this class. The class will be conducted in a remote environment that Accelebrate will provide; students will only need a local computer with a web browser and a stable Internet connection. Any recent version of Microsoft Edge, Mozilla Firefox, or Google Chrome will be fine.

**Objectives**

* Be familiar with essential cyber security concepts
* Understand web application security issues
* Analyze the OWASP Top Ten elements
* Put Web application security in the context of Java
* Go beyond the low hanging fruits
* Understand security testing methodology and approaches
* Be familiar with common security testing techniques and tools
* Handle security challenges in your Java code
* Identify vulnerabilities and their consequences
* Learn the security best practices in Java

**Outline**

* Introduction
* Cyber Security Basics
  + What is security?
  + Threat and risk
  + Cyber security threat types
  + Consequences of insecure software
* OWASP Top 10 (Part 1)
  + A1: Injection
    - Injection principles
    - Injection attacks
    - SQL injection
    - SQL injection best practices
    - Code injection
    - Injection best practices
  + A2: Broken Authentication
    - Authentication basics
    - Authentication weaknesses
    - Spoofing on the Web
    - Testing for weak authentication
    - Case study – PayPal 2FA bypass
    - Password management
* Security Testing
  + Security testing vs functional testing
  + Manual and automated methods
  + Security testing methodology
    - Security testing – goals and methodologies
    - Overview of security testing processes
    - Identifying and rating assets
    - Threat modeling
    - Security testing approaches
* The OWASP Top 10 (Part 2)
  + A3: Sensitive Data Exposure
    - Information exposure
    - Exposure through extracted data and aggregation
    - Case study – Strava data exposure
  + A4: XML External Entities (XXE)
    - DTD and the entities
    - Entity expansion
    - External Entity Attack (XXE)
  + A5: Broken Access Control
    - Access control basics
    - Failure to restrict URL access
    - Testing for authorization issues
    - Confused deputy
    - File upload
  + A6: Security Misconfiguration
    - Configuration principles
    - Configuration management
    - Java related components – best practices
  + A7: Cross-site Scripting (XSS)
    - Cross-site scripting basics
    - Cross-site scripting types
    - XSS protection best practices
  + A8: Insecure Deserialization
    - Serialization and deserialization challenges
    - Deserializing untrusted streams
    - Deserialization best practices
    - Using ReadObject
    - Sealed objects
    - Look ahead deserialization
    - Testing for insecure deserialization
    - Property Oriented Programming (POP)
  + A9: Using Components with Known Vulnerabilities
    - Using vulnerable components
    - Untrusted functionality import
    - Importing JavaScript
    - Case study – The British Airways data breach
    - Vulnerability management
  + A10: Insufficient Logging & Monitoring
    - Logging and monitoring principles
    - Insufficient logging
    - Plaintext passwords at Facebook
    - Logging best practices
    - OWASP security logging library for Java
* Web Application Security Beyond the Top Ten
  + Client-side security
  + Tabnabbing
  + Reverse tabnabbing
  + Frame sandboxing
    - Cross-Frame Scripting (XFS) attack
    - Clickjacking
    - Clickjacking beyond hijacking a click
    - Clickjacking protection best practices
    - Using CSP to prevent clickjacking
* Security Testing Techniques and Tools
  + Code analysis
    - Security aspects of code review
    - Static Application Security Testing (SAST)
    - Using static analysis tools
  + Dynamic analysis
    - Security testing at runtime
    - Penetration testing
    - Stress testing
    - Dynamic analysis tools
    - Dynamic Application Security Testing (DAST)
    - Web vulnerability scanners
    - SQL injection tools
    - Proxy servers
    - Fuzzing
* Common software security weaknesses
  + Input validation
    - Blacklists and whitelists
    - Data validation techniques
    - What to validate – the attack surface
    - Where to validate – defense in depth
    - How to validate – validation vs transformations
    - Output sanitization
    - Encoding challenges
    - Validation with regex
  + Unsafe reflection
    - Reflection without validation
* Wrap Up
  + Secure coding principles
    - Principles of robust programming by Matt Bishop
    - Secure design principles of Saltzer and Schröder
  + And now what?
    - Software security sources and further reading
    - Java resources
    - Security testing resources