

Course 910: Java Security for Web Applications

This intense hands-on workshop is essential for developers who need to produce secure Java and J2EE-based web applications. Throughout the course, students learn the best practices for designing, implementing, and deploying secure web applications using Java and J2EE. This course is short on theory and long on application.

What You'll Learn

Students who attend **Java Security for Web Applications** will leave the course armed with the skills required to recognize software vulnerabilities (actual and potential) and implement defenses for those vulnerabilities. This course quickly introduces developers to the most common security vulnerabilities faced by web applications today. Each vulnerability is examined from a Java perspective, with specific examples of both the problems and resolutions.

This course examines best practices for defensive coding, cryptography, XML/Web Services. Finally, a set of security patterns are examined pertaining to web applications and identity management within the context of web applications.

Skills and Experiences Gained

- Understand the concepts and terminology behind defensive, secure, coding.
- Understand and use Threat Risk Modeling as a tool in identifying software vulnerabilities based on realistic threats against meaningful assets.
- Understand potential sources for untrusted data.
- Understand the consequences for not properly handling untrusted data such as denial of service, cross-site scripting, and injections.
- Prevent and defend the many potential vulnerabilities associated with untrusted data.
- Perform both static code reviews and dynamic application testing to uncover vulnerabilities in Java-based web applications.
- Understand the vulnerabilities of associated with authentication and authorization.
- Understand and work with Java 2 platform security to gain an appreciation for what it protects and how
- Understand the role of Java Authentication and Authorization Service (JAAS) in J2EE applications.
- Design and develop strong, robust authentication and authorization implementations within the context of J2EE.
- Understand the basics of Java Cryptography (JCA) and Encryption (JCE) and where they fit in the overall security picture.



- Understand the fundamentals of XML Digital Signature and XML Encryption as well as how they are used within the web services arena.
- Learn how J2EE security is implemented as well as the limitations of that security
- Apply J2EE security to an existing web application.
- Understand techniques and measures that can be used to harden web and application servers as well as other components in your infrastructure.

Workshop Overview

During this three-day course, students will be led through a series of advanced topics, where most topics consist of lecture, group discussion, comprehensive hands-on lab exercises, and lab review.

The initial portion of the course provides comparative examples of vulnerable and well-defended code to illustrate in very real terms the right way to implement secure applications. The second portion of the course covers a variety of best practices, again supported with labs that illustrate these best practices in code. The last portion of the course examines several design patterns that can be used to facilitate better application architecture, design, implementation, and deployment.

This workshop is about 50% hands-on labs and 50% lecture. Many examples are threaded into the course, designed to reinforce fundamental skills and concepts learned in the lessons, all working in the Java environment. Because these lessons, labs and projects are presented in a building block fashion, students will gain a solid understanding of not only the core concepts, but also how all the pieces fit together in a complete application.

At the end of each lesson, developers will be tested with a set of review questions to ensure that he/she has fully understood that topic.

Who Should Attend

This is an **intermediate to advanced** level Java course, designed for developers who wish to get up and running on developing well defended web applications. Familiarity with Java and J2EE is required, and real world programming experience is highly recommended. This course may be customized to suit your team's unique objectives.

Pre-Requisite

Ideally students should have approximately 6 months to a year of Java and J2EE experience.

Why Choose This Course?

- Students will learn the importance of developing well-defended web applications.
- Each lesson has performance driven objectives that ensure students will learn technologies and skills core to fundamental defensive coding – nothing more, nothing less.
- We offer more than a “laundry list” approach to teaching. All lessons have clear objectives, are fundamental to learning core defensive programming practices, and are reinforced by hands-on code labs and solid practical examples.



- Progressive labs are designed in such a way that students get a firm grasp on fundamental skills while they work toward defending a complete Java application.
- All labs are take-home, and all solution code is presented in an easy to use self-study format for future use and review.

Course Details

Part 1: Top Ten Security Vulnerabilities

- **Foundation**
 - Terminology and Players
 - Assets, Threats, and Attacks
 - OWASP
 - Reality
 - Survey of recent, relevant incidents
 - Find the security defects lab
- **#1 Unvalidated Input**
 - Overview with examples
 - Cause
 - Effect
 - Broken secure programming tenets
 - Resolution with examples
 - Identifying trust boundaries
 - Qualifying untrusted data
 - Custom validation framework
 - Struts/JSF validation framework
- **#2 Broken Access Control**
 - Overview with examples
 - Cause
 - Effect
 - Broken secure programming tenets
 - Resolution with examples
 - J2EE authorization security overview
 - ServletFilter turning off cache
 - Defending special privileges such as administrative functions
 - Application authorization best practices
- **#3 Broken Authentication and Session Management**
 - Overview with examples
 - Cause
 - Effect
 - Broken secure programming tenets
 - Resolution with examples
 - Turning on SSL
 - Password management strategies
 - Password handling with encryption



- Mitigating password caching
 - Alternative authentication mechanisms
 - Best practices for session management
 - Defending session hijacking attacks
 - Best practices for Single Sign-On (SSO)
- **#4 Cross Site Scripting (XSS) Flaws**
 - Overview with examples
 - Cause
 - Effect
 - Broken secure programming tenets
 - Resolution with examples
 - Character encoding complications
 - Blacklisting
 - Whitelisting
 - HTML/XML entity encoding
- **#5 Buffer Overflows**
 - Overview with examples
 - Cause
 - Effect
 - Broken secure programming tenets
 - Resolution with examples
 - Java's strong typing
 - Java's memory model
- **#6 Injection Flaws**
 - Overview with examples
 - Cause
 - Effect
 - Broken secure programming tenets
 - Resolution with examples
 - Qualifying untrusted data
 - JDBC with PreparedStatements
 - Hibernate best practices
 - XML best practices
 - Third party API's
- **#7 Improper Error Handling, Auditing, and Logging**
 - Overview with examples
 - Cause
 - Effect
 - Broken secure programming tenets
 - Resolution with examples
 - J2EE web application exception handling framework
 - Error response best practices
 - Error, auditing, and logging content management
 - Error, auditing, and logging service management
- **#8 Insecure Storage**
 - Overview with examples
 - Cause
 - Effect



- Broken secure programming tenets
- Resolution with examples
 - Risk minimization
 - Cryptography Overview
 - JCA/JCE
 - Data encryption
 - Partial/Complete
 - Property/Deployment/Configuration files
- **#9 Denial of Service**
 - Overview with examples
 - Cause
 - Effect
 - Broken secure programming tenets
 - Resolution with examples
 - Best practices for resource management
 - Preventing premature resource allocation
 - Balancing defense and access
- **#10 Insecure Configuration Management**
 - Overview with examples
 - Cause
 - Effect
 - Broken secure programming tenets
 - Resolution with examples
 - System hardening
 - J2EE application server configuration “Gotchas!”
 - Hardening software installation
- **#11 Dynamic Loading**
 - Overview with examples
 - Cause
 - Effect
 - Broken secure programming tenets
 - Resolution with examples
 - Java Byte Code Verifier
 - Reference ahead to Java best practices
 - XML/DTD/Schema/XSLT best practices

Part 2: Best Practices and Design Patterns

- **Best Practices**

Each Best Practices is illustrated with a working code example

 - Defensive Coding Principles
 - Attack Surface Management
 - Application States
 - Defense in Depth
 - Not Trusting the Untrusted
 - No Security Through Obscurity
 - Security Defect Mitigation
 - Leverage Experience



- Java Best Practices
 - Code obfuscation
 - JAAS usage
 - Java 2 security and policy files
 - Signing JAR files
- Cryptography Best Practices
 - Using strong functions and algorithms
 - Key management and storage
 - Preventing reversible authentication tokens
 - Safe UUID management
 - SSL best practices
- XML/Web Services Best Practices
 - Operating in safe mode
 - Appropriate protocol layer for WS Security
 - Using standards-based security
 - XML-aware security infrastructure
 - WSDL protection
 - Message validation, compliance, and inspection
- **J2EE Web Application Security Design Patterns**
Each Design Pattern is illustrated with a working code example
 - Authentication Enforcer
 - Authorization Enforcer
 - Intercepting Validator
 - Secure Base Action
 - Secure Logger
 - Secure Pipe
 - Secure Service Proxy
 - Intercepting Web Agent
 - Assertion Builder for Identity
 - SSO Delegator
 - Credential Tokenizer

Please contact your ROI representative to discuss course tailoring!!!