

- Still looking for a job?
- Malaysian Diploma / Degree holder?
- Interested in a career in ICT industry?

**FREE
TRAINING!**

JOIN MSC MALAYSIA JOB CAMP

Course Name: SECURING J2EE WEB APPLICATION

Duration: 15 days

Training Location: Nota Asia (M) Sdn Bhd, Subang
Jaya / KL Plaza

SECURING J2EE WEB APPLICATION

Modules Details / Curriculum

- Foundation
 - Top Ten Security Vulnerabilities
 - Best Practices and Design Patterns
- (Course outline as enclosed)*

Target Audience

This is an intermediate to advanced level J2EE course, designed for developers who wish to get up and running on developing well defended web applications. Prerequisites: 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. Ideally students should have approximately 6 months to a year of Java and J2EE experience.



MSC Malaysia via its K-Workers Development Initiatives (KDI) is driving the provision of last mile skills training to potential knowledge workers (k-workers) for the ICT industry. Trainings are currently done through partnership with training providers appointed by Multimedia Development Corporation (MDeC).

MSC Malaysia Job Camp, a KDI programme, provides fresh graduates and available k-workers the necessary training to fill immediate vacancies in MSC, Malaysia Status companies.

For further inquiries, please contact:

Nota Asia (M) Sdn Bhd

Sharifah Zawanah

sufyana@notaasia.com

03.5636.2080

CT-08-04, Level 8, Subang Square Corporate Tower,
Jalan SS15/4G, 47500 Subang Jaya, Selangor.

In Collaboration



MSC MALAYSIA - Giving You the Edge Through ICT

Securing J2EE Web Applications

Duration: 15 days

Introduction

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.

Students who attend Security for J2EE Web Applications will leave the course armed with the skills required to recognize actual and potential software vulnerabilities, implement defenses for those vulnerabilities, and test those defenses for sufficiency.

This course quickly introduces developers to the most common security vulnerabilities faced by web applications today. Each vulnerability is examined from a Java/J2EE perspective through a process of describing the threat and attack mechanisms, recognizing associated vulnerabilities, and, finally, designing, implementing, and testing effective defenses. In many cases, there are labs that reinforce these concepts with real vulnerabilities and attacks. Students are then challenged to design and implement the layered defenses they will need in defending their own applications.

This course examines best practices for defensively coding J2EE web applications including XML and Web Services. Finally, a set of J2EE security patterns are examined with a lab that applies a security pattern in defending against an actual complex web attack.

COURSE OBJECTIVES:

Upon successful completion of this course, the student will be able to:

- Understand the concepts and terminology behind defensive, secure, coding.
- Understand the use of 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.
- Understand techniques and measures that can be used to harden web and application servers as well as other components in your infrastructure.

PROGRAM OVERVIEW

During this 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 lays down the foundation in basic terminology and concepts that is built upon in subsequent lessons. The second portion of the course steps through a series of vulnerabilities illustrating in very real terms the right way to implement secure web applications. 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 a code course, rather than theory and concepts, with 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 J2EE 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, ***trainees will be tested with a set of review questions*** to ensure that he/she has fully understands that topic.

This course is approximately 50% hands-on. There are many hands-on mini-projects interspersed throughout this course, presented in a building block fashion.

AUDIENCE:

This is an intermediate to advanced level J2EE course, designed for developers who wish to get up and running on developing well defended web applications. Prerequisites: 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. Ideally students should have approximately 6 months to a year of Java and J2EE experience.

TOPICS OUTLINE:

Part 1: Foundation

I. Foundation

- Terminology and Players
 - Assets, Threats, and Attacks
 - OWASP
 - Basic Principles
- Reality
 - Survey of recent, relevant incidents
 - Lab to find the security defects in an existing web application

Part 2: Top Ten Security Vulnerabilities

I. Unvalidated Input

- Overview with examples
 - Cause
 - Effect
- Broken secure programming tenets
- Resolution with examples
 - Identifying trust boundaries
 - Qualifying untrusted data
 - Implementing a layered defense that effectively protects quality of service as well as data integrity
 - Designing an appropriate response to a recognized attack
 - Testing defenses and responses for weaknesses

II. 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

III. Broken Authentication and Session Management

- Overview with examples
 - Cause
 - Effect
- Broken secure programming tenets
- Resolution with examples
 - Multi-layered defenses of authentication services
 - Password management strategies
 - Password handling with hashing
 - Mitigating password caching
 - Testing defenses and responses for weaknesses
 - Alternative authentication mechanisms

- Best practices for session management
- Defending session hijacking attacks
- Best practices for Single Sign-On (SSO)

IV. 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
 - Understanding the implications of trust boundary definition
 - Implementing a layered defense that effectively protects quality of service as well as XSS vulnerabilities
 - Designing an appropriate response to a recognized attack

V. Buffer Overflows

- Overview with examples
 - Cause
 - Effect
- Broken secure programming tenets
- Resolution with examples
 - Java's strong typing
 - Java's memory model

VI. Injection Flaws

- Overview with examples
 - Cause
 - Effect
- Broken secure programming tenets
- Resolution with examples
 - Qualifying untrusted data
 - JDBC with PreparedStatement
 - Hibernate best practices
 - XML best practices
 - Third party API's
 - Implementing a layered defense that effectively protects quality of service as well as injection vulnerabilities
 - Designing an appropriate response to a recognized attack

VII. 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
 - Best practices for supporting web attack forensics

VIII. Insecure Storage

- Overview with examples
 - Cause
 - Effect
- Broken secure programming tenets
 - Data leakage
- Resolution with examples
 - Risk minimization
 - Cryptography Overview
- JCA/JCE
- Data encryption
- Partial/Complete
- Property/Deployment/Configuration files

IX. Insecure Management of Configuration

- Overview with examples
 - Cause
 - Effect
- Broken secure programming tenets
- Resolution with examples
 - System hardening
 - J2EE application server configuration “Gotchas!”
 - Hardening software installation

X. 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

XI. Spoofing

- Overview with examples
 - Cause
 - Effect
- Broken secure programming tenets
- Resolution with examples
 - Protecting your clients
 - Defending against Cross Site Request Forgeries
 - Phishing Defenses

Part 3: Best Practices and Design Patterns

I. 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

II. Defending XML Processing and Web Services

- Understanding common attacks and how to defend
- 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

III. 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

Security experts agree that the least effective approach to security is “penetrate and patch”. It is far more effective to “bake” security into an application throughout its lifecycle. An optional fourth day builds on the previously learned mechanics for building defenses by exploring how design and analysis can be used to build stronger applications from the beginning of the software lifecycle.

IV. Secure Design and Analysis Design and Analysis Processes

- Motivation
- Security Development Lifecycle (SDL)
- CLASP applied

V. Application of Design and Analysis Processes

- Threat Risk Modeling
 - Lab applying threat risk modeling
- Testing and Review Best Practices
 - Lab applying review processes

Course Structure

Week	Day	Module
1	1	Pre-Test, Part 1, 2 (I, II, III) Discussions, Group activities
	2	Part 2 (IV, V, VI) Discussions, Group activities
	3	Part 2 (VII, VIII) Discussions, Group activities
	4	Part 2 (IX, X) Discussions, Group activities
	5	Part 2 (XI) Discussions, Group activities
2	1	Part 3 (I, II) Discussions, Group activities
	2	Part 3 (III) Discussions, Group activities
	3	Part 3 (IV) Discussions, Group activities
	4	Part 3 (V) Discussions, Group activities
	5	Post-Test, Discussions, Group activities
3	1	Project Day 1
	2	Project Day 2
	3	Project Day 3
	4	Project Day 4
	5	Project Presentation