Developing Secure Java Web Applications (TT8320-J)

Modality: Virtual Classroom

Duration: 4 Days

About this course:

The training course of Java / JEE security is a hands-on, lab-intensive course, important for experienced enterprise designers who need to maintain, engineer, and support secure JEE-based web applications. Furthermore, teaching basic skills of secure programming, this course dives deep into sound practices and processes that execute the whole lifecycle of software development.

The main component of our Best Training series of Defense IT Security, this workshop has many developer-oriented seminars and courses. Though this course edition is Java-specific and it may also be presented using .Net or other languages of programming.

Understudies completely examine best practices for protectively coding web applications in this course, including rich interfaces, XML processing, and both SOAP and RESTful based web services. Understudies will more than once assault and afterward shield different resources related to fully-functional web services and web applications. This approach drives home the mechanics of the way to protect the applications of .Net web in the most functional of terms.

The specialists of security admitted that the least successful way to deal with security is "penetrate and patch". It is progressively successful to "bake" security into an application all through its lifecycle. In the wake of spending noteworthy time trying to shield a worst planned (from a security point of view) web application, developers are prepared to figure out the way to construct secure web applications starting at task inception. The final segment of this course expands on the previously learned mechanics for building barriers by exploring the method to analyze and design can be utilized to make solid applications from the earliest starting point of the software lifecycle.

The normal pay of a Java Developer is $69,722 every year.

Course Objective:

- To test web applications with different strategies of assault to determine the presence of and adequacy of layered defenses.
- Comprehend the vulnerabilities related to authorization and authentication.
- To detect, attack, and implement defenses for both RESTful and SOAP-based web services and functionality
- Comprehend the essentials of XML Encryption and XML Digital Signature and also how they are utilized within the arena of web services.
- Comprehend the terminology and concepts behind secure, defensive, and coding
- Perform both dynamic application testing and static code reviews to uncover vulnerabilities in .Net-based web applications
- To detect, assault, and apply safeguards against Injection and XSS assaults.
- Develop and design solid, robust authorization and authentication executions within the

context of .Net.
- Function with a comprehensive plan of testing for recognized weaknesses and vulnerabilities.
- Acquire the tools, skills, and best practices for code and design reviews and also testing initiatives
- Understand the fundamentals of security planning and testing
- Comprehend and apply the measures and processes associated with the Secure Software Development (SSD)
- Design and develop strong, robust authentication and authorization implementations within the context of .Net
- Understand the fundamentals of XML Digital Signature and XML Encryption as well as how they are used within the web services arena
- Comprehend the consequences for not appropriately handling untrusted information, for example, the cross-site scripting, denial of service, and injections.

Audience:

This course is an intermediate - level JEE/web services programming course, intended for developers who like to find a good pace on developing much-protected software applications. To suit the unique objectives of your team, this course may be customized.

Prerequisite:

Awareness of Java and JEE is necessary and supportable programming experience is suggested. Ideally, understudies ought to have around a half year to a time of Java and JEE practical information.

Course Outline:

Module 1: Foundation

Lesson: Who is Safe?

- Assumptions We Make
- Security: The Complete Picture
- Anthem, Sony, Target, Heartland, and TJX Debriefs
- Verizon’s 2017 Data Breach Report
- Attack Patterns and Recommendations
- Tutorial: Working with Eclipse (JEE Version) and Tomcat
- Tutorial: Working with the HSQL Database
- Exercise: Case Study Setup and Review

Lesson: Security Concepts

- Motivations: Costs and Standards
- Open Web Application Security Project
- Web Application Security Consortium
- CERT Secure Coding Standards
- Microsoft SDL

- Assets and Trust Boundaries
- Threat Modeling
- Exercise: Case Study Asset Analysis

Lesson: Principles of Information Security

- Security Is a Lifecycle Issue
- Minimize Attack Surface Area
- Layers of Defense: Tenacious D
- Compartmentalize
- Consider All Application States
- Do NOT Trust the Untrusted

Module 2: Vulnerabilities (Part 1)

Lesson: Unvalidated Input

- Buffer Overflows
- Integer Arithmetic Vulnerabilities
- Unvalidated Input: From the Web
- Defending Trust Boundaries
- Whitelisting vs Blacklisting
- Exercise: Defending Trust Boundaries
- Exercise: Defending Trust Boundaries With Regular Expressions

Lesson: Broken Access Control

- Access Control Issues
- Excessive Privileges
- Insufficient Flow Control
- Unprotected URL/Resource Access
- Examples of Shabby Access Control
- Module s and Module Management

Lesson: Broken Authentication

- Broken Quality/DoS
- Authentication Data
- Username/Password Protection
- Exploits Magnify Importance
- Handling Passwords on Server Side
- Single Sign-on (SSO)
- Exercise: Defending Authentication

Lesson: Cross Site Scripting (XSS)

- XSS Patterns

- Persistent XSS
- Reflective XSS
- Best Practices for Untrusted Data
- Exercise: Defending Against XSS

Lesson: Injection

- Injection Flaws
- SQL Injection Attacks Evolve
- Drill Down on Stored Procedures
- Other Forms of Injection
- Minimizing Injection Flaws
- Exercise: Defending Against SQL Injection

Module 3: Vulnerabilities (Part 2)

Lesson: Error Handling and Information Leakage

- Fingerprinting a Web Site
- Error-Handling Issues
- Logging In Support of Forensics
- Solving DLP Challenges
- Exercise: Error Handling

Lesson: Insecure Data Handling

- Protecting Data Can Mitigate Impact
- In-Memory Data Handling
- Secure Pipes
- Failures in TLS/SSL Framework
- Exercise: Defending Sensitive Data

Lesson: Insecure Configuration Management

- System Hardening: IA Mitigation
- Application Whitelisting
- Least Privileges
- Anti-Exploitation
- Secure Baseline

Lesson: Direct Object Access

- Remote File Inclusion
- Redirects and Forwards
- Direct Object References
- Exercise: Unsafe Direct Object References

Lesson: Spoofing, CSRF, and Redirects

- Name Resolution Vulnerabilities
- Fake Certs and Mobile Apps
- Targeted Spoofing Attacks
- Cross Site Request Forgeries (CSRF)
- CSRF Defenses
- Exercise: Cross-Site Request Forgeries

Module 4: Best Practices

Lesson: Cryptography Overview

- Strong Encryption
- Message Digests
- Encryption/Decryption
- Keys and Key Management
- NIST Recommendations

Lesson: Understanding What's Important

- Common Vulnerabilities and Exposures
- OWASP 2017 Top Ten
- CWE/SANS Top 25 Most Dangerous SW Errors
- Monster Mitigations
- Strength Training: Project Teams/Developers
- Strength Training: IT Organizations
- Exercise: Recent Incidents

Module 5: Defending XML, Services, and Rich Interfaces

Lesson: Defending XML

- XML Signature
- XML Encryption
- XML Attacks: Structure
- XML Attacks: Injection
- Safe XML Processing
- Exercise: Safe XML Processing
- Exercise: Dynamic Loading Using XSLT

Lesson: Defending Web Services

- Web Service Security Exposures
- When Transport-Level Alone is NOT Enough
- Message-Level Security

- WS-Security Roadmap
- Java's XWSS API
- Web Service Attacks
- Web Service Appliance/Gateways
- Exercise: Web Service Attacks

Lesson: Defending Rich Interfaces and REST

- How Attackers See Rich Interfaces
- Attack Surface Changes When Moving to Rich Interfaces and REST
- Bridging and its Potential Problems
- Three Basic Tenets for Safe Rich Interfaces
- OWASP REST Security Recommendations
- OAuth 2.x and OpenID
- Exercise: Working with OAuth