

Course 914: Creating Secure Code (5 days)

Course Description...

According to research by the National Institute of Standards, 92% of all security vulnerabilities are now considered application vulnerabilities and not network vulnerabilities. This intense hands-on workshop is essential for developers who need to produce secure Java and J2EE applications, integrating security measures into the development process from requirements to deployment and maintenance.

Throughout the course, students learn the best practices for designing, implementing, and deploying secure programs in Java and J2EE. Students will take an application from requirements through to implementation, analyzing and testing for software vulnerabilities.

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. The final portion of this course 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.

Students who attend Java Secure Coding will leave the course armed with the required skills to recognize software vulnerabilities (actual and potential) and implement defenses for those vulnerabilities. This course quickly introduces developers to the various types of threats against their software.

The concept and process of Threat Risk Modeling is introduced as a key enabler for implementing effective and appropriate security for software and information assets. This course includes coverage of the many security-related technologies and APIs that exist in the Java and J2EE world.

Hands-On Workshops...

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

This workshop is about 40% hands-on lab and 60% lecture. Multiple complete "mini-projects" are laced throughout 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 understands that topic.



Learning Objectives...

- Understand the concepts and terminology behind defensive coding.
- Understand and use Threat Risk Modeling as a tool in identifying software vulnerabilities based on realistic threats against meaningful assets.
- Learn the entire spectrum of threats and attacks that take place against software applications in today's world.
- Use Threat Risk Modeling to identify potential vulnerabilities in a real life case study.
- Perform both static code reviews and dynamic application testing to uncover vulnerabilities in Java applications.
- Understand the vulnerabilities of the Java programming language and the JVM as well as how to harden both.
- Understand and work with Java 2 platform security to gain an appreciation for what is protected and how
- Understand the role that Java Authentication and Authorization Service (JAAS) has in both Java and J2EE applications.
- Use JAAS in conjunction with a Java application for both authentication and authorization.
- 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.
- Understand and implement the processes and measures associated with the security development lifecycle
- Understand the basics of security testing and planning
- Work through a comprehensive testing plan for recognized vulnerabilities and weaknesses

Who should attend...

This is an intermediate level Java course, designed for developers who wish to get up and running on developing well defended software applications. This course may be customized to suit your team's unique objectives.

Familiarity with Java and object-oriented technologies is required, and real world programming experience is highly recommended.

Prerequisites...

Students should have basic development skills and a working knowledge in the following topics, or attend these courses as a pre-requisite:

- Understanding Internet Architectures
- Essential Java Programming
- Building J2EE Web Applications

See next page for a detailed course outline...



Course Outline...

Defensive Coding Overview

- Security Concepts
- Principles of Defensive Coding
- Threat Risk Modeling
- Threat Risk Modeling of Case Study

Vulnerabilities

- Security Attacks
- Information Attacks
- System Attacks
- Data Attacks

Defensive Coding Applied to Java

- Basic Principles Revisited
- Defensive Java Coding

Java Tools for Cryptographic Key Management

- What is “strong” encryption?
- Symmetric ciphers
- Public-key cryptography
- Message digests
- Digital signatures
- Choosing the algorithm and key lengths
- Key exchange mechanisms
- Key management (KeyStore)

Java Cryptography APIs and Extensions

- The Java cryptography APIs
- Types of ciphers
- Encrypting data
- Decrypting data
- Exchanging keys
- Extracting keys from key stores
- Block ciphers
- Padding
- Salting
- Message Digests in Java
- Message Authentication Codes
- Cryptographic providers
- Installing and configuring a provider

Code location-based security in Java

- Language security
- Prohibited operations
- Why the compiler is not enough
- Byte code modification attacks
- Byte code verification
- Types of class loaders
- Class loaders as implicit namespaces
- CodeSource, SecureClassLoader and URLClassLoader
- Creating custom class loader
- findClass



- Invoking methods through reflection
- On using encrypted classes to prevent decompilation
- Code obfuscation
- Byte working toolkits
- Best practices when using class loaders

Trusted code

- Limitations of Java sandbox model
- Signing code
- Security policies
- Permission classes
- Defining a security policy
- Built-in permission classes
- Using CodeSource and Principal
- Bringing it together with ProtectionDomain
- Dynamic policies
- Policy enforcement
- Basic enforcement with SecurityManager
- Using AccessControlContext and DomainCombiner
- Customized enforcement with AccessController
- Comparing SecurityManager to AccessController
- Signing JAR files
- Secure JRE installation
- Preventing security properties file overrides
- Configuring application specific policies
- Deploying security provider code

User-based J2SE security

- Limitations of code location-based security
- Creating a custom permission class
- Permission “implies”
- General vs specific permissions
- Granting access to selected users
- Principals and Subjects
- Login contexts
- Required, sufficient, requisite and optional
- PrivilegedAction and PrivilegedExceptionAction
- Subject’s doAsPrivileged
- Creating a custom security manager
- Creating a custom login module
- 2-phase login protocols
- Custom callbacks
- Storing and clearing passwords

Java Network Security

- Need for secure communications
- Authenticating over untrusted networks
- Securing data transmitted over untrusted networks
- The SSL process
- SSL sockets
- Getting and processing SSL data
- HttpsURLConnection
- X509 trust manager
- Secure message exchanges



- Generic security service (GSS)
- Using GSS with JAAS login
- SASL for pluggable Internet authentication
- Choosing amongst SASL and JAAS/JGGS
- SASL client and server
- Negotiated security layers

Java code-level Security Gotchas and best practices

- Exception management
- Access control to Fields and methods
- Problems with package-level access
- Package sealing
- Inner classes and generics
- Problems with Java initialization
- Problems with code signing
- Cloneability, serialization and deserialization
- Mutable objects, arrays and collections
- Comparing classes
- Prefer private and final
- Static, non-final variables
- Defensive firewalls
- Input validation
- SQL injection attacks
- Tainted variables
- Unit testing with random scenarios
- Privileged code blocks
- When to add security checks
- JNI best practices

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

Security Development Lifecycle (SDL)

- SDL Process Overview
 - CLASP Defined
 - CLASP Applied
- Asset, Boundary, and Vulnerability Identification, Vulnerability Response, Design and Code Reviews
- Applying Security Policies and Practices
- Risk Analysis

Security Testing

- Testing as Lifecycle Process
- Testing Planning and Documentation
- Testing Tools
- Approaches for Testing:
 - Information Leakage
 - Business Logic



- Authentication
- Session Management
- Input Data Validation
- Denial of Service
- Web Services