Certified Application Security Engineer (CASE)

Course Outline

Module 01: Understanding Application Security, Threats, and Attacks

- What is a Secure Application
- Need for Application Security
- Most Common Application Level Attacks
  - SQL Injection Attacks
  - Cross-site Scripting (XSS) Attacks
  - Parameter Tampering
  - Directory Traversal
  - Cross-site Request Forgery (CSRF) Attack
  - Denial-of-Service (DoS) Attack
    - Denial-of-Service (DoS): Examples
  - Session Attacks
    - Cookie Poisoning Attacks
    - Session Fixation
- Why Applications become Vulnerable to Attacks
  - Common Reasons for Existence of Application Vulnerabilities
  - Common Flaws Existed due to Insecure Coding Techniques
  - Improper Input Validation
  - Insufficient Transport Layer Protection
  - Improper Error Handling
  - Insecure Cryptographic Storage
  - Broken Authentication and Session Management
  - Unvalidated Redirects and Forwards

- Insecure Direct Object References
- Failure to Restrict URL Access

- What Constitutes a Comprehensive Application Security?
  - Application Security Frame
  - 3W's in Application Security
- Insecure Application: A Software Development Problem
  - Solution: Integrating Security in Software Development Life Cycle (SDLC)
  - Functional vs Security Activities in SDLC
  - Advantages of Integrating Security in SDLC
- Software Security Standards, Models, and Frameworks
  - The Open Web Application Security Project (OWASP)
  - OWASP TOP 10 Attacks-2017
  - The Web Application Security Consortium (WASC)
  - WASC Threat Classification
  - Software Security Framework
    - Software Assurance Maturity Model (SAMM)
    - Building Security in Maturity Model (BSIMM)
  - BSIMM vs OpenSAMM

Module 02: Security Requirements Gathering

- Importance of Gathering Security Requirements
  - Security Requirements
  - Gathering Security Requirements
  - Why We Need Different Approach for Security Requirements Gathering
  - Key Benefits of Addressing Security at Requirement Phase
  - Stakeholders Involvement in Security Requirements Gathering
  - Characteristics of Good Security Requirement: SMART
  - Types of Security Requirements
    - Functional Security Requirements
    - Security Drivers
- Security Requirement Engineering (SRE)

- SRE Phases
  - Security Requirement Elicitation
  - Security Requirement Analysis
  - Security Requirement Specification
  - Security Requirement Management
- Common Mistakes Made in Each Phase of SRE
- Different Security Requirement Engineering Approaches/Model

▪ Abuse Case and Security Use Case Modeling

- Abuse Cases
- Threatens Relationship
- Abuse Case Modeling Steps
- Abuse Cases: Advantages and Disadvantages
- Abuse Case Template
- Security Use Cases
- Security Use Cases are Abuse Case Driven
- Modeling Steps for Security Use Cases
- Mitigates Relationship
- Abuse Case vs Security Use Case
- Security Use Case: Advantages and Disadvantages
- Security Use Case Template
- Security Use Case Guidelines
- Example 1: Use Case for Online Bidding System
- Example 1: Abuse Case for Online Bidding System
- Example 1: Security Use Case for Online Bidding System
- Example 2: Use Case for ATM System
- Example 2: Abuse Case for ATM System
- Example 2: Security Use Case for ATM System
- Example 3: Use Case for E-commerce System
- Example 3: Abuse Case for E-commerce System
- Example 3: Security Use Case for E-commerce System

- Effectiveness of Abuse and Security Case

- Abuser and Security Stories
  - Textual Description Template: Abuser Stories and Security Stories
  - Examples: Abuser Stories and Security Stories
  - Effectiveness of Abuser and Security Stories
  - Abuser Stories: Advantages and Disadvantages
- Security Quality Requirements Engineering (SQUARE)
  - SQUARE Effectiveness
  - SQUARE Process
  - SQUARE: Advantages and Disadvantages
- Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE)
  - OCTAVE Effectiveness
  - OCTAVE Steps
  - OCTAVE: Advantages and Disadvantages

Module 03: Secure Application Design and Architecture

- Relative Cost of Fixing Vulnerabilities at Different Phases of SDLC
- Secure Application Design and Architecture
- Goal of Secure Design Process
- Secure Design Actions
  - Security Requirement Specifications
  - Secure Design Principles
  - Threat Modeling
  - Secure Application Architecture
- Secure Design Principles
  - Define Secure Design principles
  - Secure Design Principles
    - Security through obscurity
    - Secure the Weakest Link
    - Use Least Privilege Principle
    - Secure by Default

        - Fail Securely
        - Apply Defense in Depth
        - Do Not Trust User Input
        - Reduce Attack Surface
        - Enable Auditing and Logging
        - Keep Security Simple
        - Separation of Duties
        - Fix Security Issues Correctly
        - Apply Security in Design Phase
        - Protect Sensitive Data
        - Exception Handling
        - Secure Memory Management
        - Protect Memory or Storage Secrets
        - Fundamentals of Control Granularity
        - Fault Tolerance
        - Fault Detection
        - Fault Removal
        - Fault Avoidance
        - Loose Coupling
        - High Cohesion
        - Change Management and Version Control
- Threat Modeling
    - Threat Modeling Phases
        - Attack Surface Evaluation
        - Threat Identification
        - Impact Analysis
        - Control Recommendations
    - Threat Modeling Process
        - Identify Security Objective
        - Application Overview
        - Decompose Application

        - Identify Threats
        - Identify Vulnerabilities
    - Identify Security Objective
        - How to Identify Security Objectives
    - Create an Application Overview
    - Draw the End-to-End Deployment Architecture
    - Identify Various User Roles
    - Identify Use Cases Scenarios
    - Identify Technologies
    - Identify Application Security Mechanisms
- Decompose Application
    - Prepare and Document Threat Model Information
        - Example: Threat Model Information
    - Identify the External Dependencies
        - External Dependencies: Example
    - Identify the Entry Points
        - Entry Points: Example
    - Identify the Assets
        - Assets: Example
    - Identify the Trust Levels
        - Trust Levels: Example
    - Define Trust Levels to Entry points
    - Define Trust Levels to Assets
    - Perform Application Modelling using Data Flow Diagrams (DFDs)
    - Determine the Threats: Identify the Goal of an Attacker and Create Threat Profile
        - Example: Attacker's Goal/Threat Profile and Vulnerabilities Associated
    - Determine the Threats: Create a Security Profile
    - Identify the Threats
        - The STRIDE Model
            - Example: Threat Categorized and Identified using STRIDE
    - Determine Countermeasures and Mitigation Security Controls

- Document the Threats
- Rating the Threats
  - Rating the Threats: DREAD Model

- Secure Application Architecture
  - Design Secure Application Architecture

Module 04: Secure Coding Practices for Input Validation

- Input Validation Pattern
- Validation and Security Issues
- Impact of Invalid Data Input
- Data Validation Techniques
- Input Validation using Frameworks and APIs
- Open Source Validation Framework for Java
- Servlet Filters
- Validation Filters for Servlet
- Data Validation using OWASP ESAPI
- Data Validation: Struts Framework
  - Struts Validator
  - Struts Validation and Security
  - Data Validation using Struts Validator
  - Avoid Duplication of Validation Forms
  - Secure and Insecure Struts Validation Code
  - Struts Validator Class
  - Secure and Insecure Code for Struts Validator Class
  - Enable the Struts Validator
  - Secure and Insecure Struts Validator Code
  - Struts 2 Framework Validator
  - Struts 2 Framework: Built-in Data Validators
  - Struts 2 Framework Annotation Based Validators
  - Struts 2 Custom Validation: Workflow Interceptor
  - Struts 2 Ajax Validation: jsonValidation Interceptor

- Data Validation: Spring Framework
  - Spring Validator
  - Data Validation: Spring MVC Framework
  - Implementing Validator
  - JSR 380 Bean Validator API
  - Configuring JSR 380
  - Custom Validator Implementation in Spring
  - Spring Validation and Security
- Input Validation Errors
  - Improper Sanitization of Untrusted Data
  - Improper Validation of Strings
  - Improper Logging of User Inputs
  - Improper Incorporation of Malicious Inputs into Format Strings
  - Inappropriate Use of Split Characters in Data Structures
  - Improper Validation of Non-Character Code Points
  - Improper Use of String Modification
  - Improper Comparison of Locale-dependent Data
  - Best Practices for Input Validation
- Common Secure Coding Practices
  - SQL Injection
  - Prepared Statement
  - Stored Procedures
    - Vulnerable and Secure Code for Stored Procedures
  - Stored Procedure for Securing Input Validation
  - Cross-site Scripting (XSS)
  - Whitelisting vs Blacklisting
    - Vulnerable and Secure Code for Blacklisting & Whitelisting
  - Regular Expressions
    - Vulnerable and Secure Code for Regular Expressions
  - Character Encoding

  - Vulnerable and Secure Code for Character Encoding
  - Checklist for Character Encoding
- Cross-site Scripting (XSS) Countermeasures
- HTML Encoding
  - Vulnerable and Secure Code for HTML Encoding
- HTML Encoding using ESAPI Encoder
- Cross-site Request Forgery (CSRF)
  - Cross-site Request Forgery (CSRF) Countermeasures
- Directory Traversal
  - Directory Traversal Countermeasures
- HTTP Response Splitting
  - HTTP Response Splitting Countermeasures
- Parameter Manipulation and Countermeasures
- Protecting Application from Log Injection Attack
- XML Injection
- Command Injection
- LDAP Injection
- XML External Entity Attack
- Unrestricted File Upload Attack
- Prevent Unrestricted File Upload: Validate File Extension
- Injection Attacks Countermeasures
- CAPTCHA
  - Sample Code for Creating CAPTCHA
  - Sample Code for CAPTCHA Verification
  - Sample Code for Displaying CAPTCHA
- Best Practices for Input Validation

Module 05: Secure Coding Practices for Authentication and Authorization

- Introduction to Authentication
  - Java Container Authentication
  - Authentication Mechanism Implementation

- Types of Authentication
  - Declarative vs Programmatic Authentication
  - Declarative Security Implementation
  - Programmatic Security Implementation
  - Java EE Authentication Implementation Example
  - Basic Authentication
  - How to Implement Basic Authentication?
  - Form-based Authentication
  - Form-based Authentication Implementation
  - Implementing Kerberos-Based Authentication
  - Secured Kerberos Implementation
  - Client Certificate Authentication
  - Certificate Generation with Keytool
  - Implementing Encryption and Certificates in Client Application
- Authentication Weaknesses and Prevention
  - Brute Force Attack
  - Web-based Enumeration Attack
  - Weak Password Attacks
- Introduction to Authorization
  - JEE Based Authorization
    - Declarative
    - Programmatic
- Access Control Model
  - Discretionary Access Control (DAC)
  - Mandatory Access Control (MAC)
  - Role-based Access Control (RBAC)
  - Servlet Container
  - Authorizing Users by Servlets
- EJB Authorization
  - EJB Authorization Controls

- Declarative Security with EJBs
- Programmatic Security with EJBs

- Java Authentication and Authorization (JAAS)
  - JAAS Features
  - JAAS Architecture
  - Pluggable Authentication Module (PAM) Framework
  - JAAS Classes
  - JAAS Subject and Principal
  - Authentication in JAAS
    - Authentication Steps in JAAS
  - Authorization in JAAS
    - Authorization Steps in JAAS
  - Subject Methods doAs() and doAsPrivileged()
  - Impersonation in JAAS
  - JAAS Permissions
  - LoginContext in JAAS
  - Creating LoginContext
  - LoginContext Instantiation
  - JAAS Configuration
  - Locating JAAS Configuration File
  - JAAS CallbackHandler and Callbacks
  - Login to Standalone Application
  - JAAS Client
  - LoginModule Implementation in JAAS
  - Methods Associated with LoginModule
  - LoginModule Example
  - Phases in Login Process
- Java EE Security
  - Java EE Application Architecture
  - Java EE Servers as Code Hosts

- Declaring Roles
- HTTP Authentication Schemes

- Authorization Common Mistakes and Countermeasures
  - Common Mistakes
- Authentication and Authorization in Spring Security Framework
  - Spring Security Framework
  - Spring Security Modules
  - Spring Authentication
  - Storing Username and Password
  - Securing Authentication Provider
  - Implementing HTTP Basic Authentication
  - Form-based Authentication
  - Implementing Digest Authentication
  - Security Expressions
  - URL-based Authorization
  - JSP Page Content Authorization
  - JSP Page Content Authorization with Domain Object's ACL
  - Method Authorization
  - Configuring Anonymous Login
  - Logout Feature Configuration
  - Remember-Me Authentication
  - Integrating Spring Security with JAAS
  - Spring JAAS Implementation
- Defensive Coding Practices against Broken Authentication and Authorization
  - Do Not Store Password in Java String Object
  - Avoid Cookie based Remember-Me Use Persistent Remember-Me
  - Implement Appropriate Session Timeout
  - Prevent Session Stealing by Securing SessionID Cookie
- Secure Development Checklists: Broken Authentication and Session Management

Module 06: Secure Coding Practices for Cryptography

- Java Cryptography
  - Need for Java Cryptography
  - Java Security with Cryptography
  - Java Cryptography Architecture (JCA)
  - Java Cryptography Extension (JCE)
- Encryption and Secret Keys
  - Attack Scenario: Inadequate/Weak Encryption
  - Encryption: Symmetric and Asymmetric Key
  - Encryption/Decryption Implementation Methods
  - SecretKeys and KeyGenerator
  - Implementation Methods of KeyGenerator Class
  - Creating SecretKeys with KeyGenerator Class
- Cipher Class
  - The Cipher Class
  - Implementation Methods of Cipher Class
  - Insecure Code for Cipher Class using DES Algorithm
  - Secure Code for Cipher Class using AES Algorithm
- Digital Signatures
  - Attack Scenario: Man-in-the-Middle Attack
  - Digital Signatures
  - The Signature Class
  - Implementation Methods of Signature Class
  - The SignedObjects
  - Implementing Methods of SignedObjects
  - The SealedObjects
  - Implementation Methods of SealedObject
  - Insecure and Secure Code for Signed/Sealed Objects
  - Java XML Digital Signature
- Secure Socket Layer (SSL)

- Java Secure Socket Extension (JSSE)
- SSL and Security: Example 1
- SSL and Security: Example 2
- JSSE and HTTPS
- Insecure HTTP Server Code
- Secure HTTP Server Code

- Key Management
  - Attack Scenario: Poor Key Management
  - Keys and Certificates
  - Key Management System
  - KeyStore
  - Implementation Method of KeyStore Class
  - KeyStore: Persistent Data Stores
  - Key Management Tool: KeyTool
- Digital Certificates
  - Certification Authorities
  - Signing Jars
  - Signing JAR Tool: Jarsigner
- Signed Code Sources
  - Insecure Code for Signed Code Sources
  - Secure Code for Signed Code Sources
- Hashing
  - Hashing Algorithms
  - Securing Hashed Password with Salt
  - Implementing Hashing with Salt in Spring Security
- Java Card Cryptography
- Spring Security: Crypto Module
  - Crypto Module
  - Spring Security Crypto Module
    - Key Generators

        - PasswordEncoder
    - Implementing BCryptPasswordEncoder()
    - Configuring BCryptPasswordEncoder() in Spring Security
    - JavaScript Object Signing and Encryption (JOSE)
    - Attacks against JWT, JWS and JWE
    - Implementing JWS using Jose4J
    - Implementing JWE using Jose4J
    - Implementing JWK using Jose4J
- Dos and Don'ts in Java Cryptography
    - Dos and Don'ts
        - Avoid using Insecure Cryptographic Algorithms
        - Avoid using Statistical PRNG, Inadequate Padding and Insufficient Key Size
        - Implement Strong Entropy
        - Implement Strong Algorithms
- Best Practices for Java Cryptography

Module 07: Secure Coding Practices for Session Management

- Session Management
- Session Tracking
    - Session Tracking Methods
        - HttpSession
        - Cookies
            - Setting a Limited Time Period for Session Expiration
            - Preventing Session Cookies from Client-Side Scripts Attacks
        - URL Rewriting
            - Example Code for URL Rewriting
        - Hidden Fields
        - Session Objects
- Session Management in Spring Security
    - Spring Session Management
    - Session Management using Spring Security

- Restricting Concurrent Sessions per User using Spring Security
- Controlling Session Timeout
- Prevent using URL Parameters for Session Tracking
- Prevent Session Fixation with Spring Security
- Use SSL for Secure Connection

- Session Vulnerabilities and their Mitigation Techniques
  - Session Vulnerabilities
  - Types of Session Hijacking Attacks
  - Countermeasures for Session Hijacking
  - Countermeasures for Session ID Protection
- Best Practices and Guidelines for Secured Sessions Management
  - Best Coding Practices for Session Management
- Checklist to Secure Credentials and Session IDs
- Guidelines for Secured Session Management

Module 08: Secure Coding Practices for Error Handling

- Introduction to Exceptions
  - Exception and Error Handling
    - Checked Exceptions
    - Unchecked Exceptions
  - Example of an Exception
  - Handling Exceptions in Java
  - Exception Classes Hierarchy
  - Exceptions and Threats
- Erroneous Exceptional Behaviors
  - Suppressing or Ignoring Checked Exceptions
  - Disclosing Sensitive Information
  - Logging Sensitive Data
  - Restoring Objects to Prior State, if a Method Fails
  - Avoid using Statements that Suppress Exceptions
  - Prevent Access to Untrusted Code that Terminates JVM

- Never Catch java.lang.NullPointerException
- Never Allow methods to Throw RuntimeException, Exception, or Throwable
- Never Throw Undeclared Checked Exceptions
- Never Let Checked Exceptions Escape from Finally Block

- Dos and Don'ts in Error Handling
  - Dos and Don'ts in Exception Handling
  - Avoid using Log Error and Throw exception at Same Time
- Spring MVC Error Handling
  - Handling Controller Exceptions with @ExceptionHandler Annotation
  - Handling Controller Exceptions with HandlerExceptionResolver
  - Spring MVC: Global Exception Handling
  - Global Exception Handling: HandlerExceptionResolver
  - Mapping Custom Exceptions to Statuscode with @ResponseStatus
  - Configure Custom Error Page in Spring MVC
- Exception Handling in Struts 2
  - Exception Handling: Struts 2
- Best Practices for Error Handling
  - Best Practices for Handling Exceptions in Java
- Introduction to Logging
  - Logging in Java
  - Example for Logging Exceptions
  - Logging Levels
- Logging using Log4j
  - Log4j and Java Logging API
  - Java Logging using Log4j
- Secure Coding in Logging
  - Vulnerabilities in Logging
  - Logging: Vulnerable Code and Secure Code
- Secured Practices in Logging

Module 09 Static and Dynamic Application Security Testing (SAST & DAST)

- Static Application Security Testing
  - Static Application Security Testing (SAST)
  - Objectives of SAST
  - Why SAST
  - Skills required for SAST
  - What to look for in SAST
  - Common Vulnerabilities Identified Through SAST
  - Types of SAST
    - Automated Source Code Analysis
    - Manual Source Code Review
  - Where does Secure Code Review Fit in SDLC?
  - SAST Steps
  - SAST Activities- flow Chart
  - Recommendation for Effective SAST
  - SAST Deliverable
  - Automated Source Code Analysis
    - Static Code Analysis using Checkmarx Static Code Analysis
    - Static Code Analysis using Visual Code Grepper (VCG)
    - Static Code Analysis using HP Fortify
    - Static Code Analysis using Rational AppScan Source Edition
  - Selecting Static Analysis Tool
  - Manual Secure Code Review
- Manual Secure Code Review for Most Common Vulnerabilities
  - Code Review for PCI DSS Compliance
  - Code Review for Blacklisting Validation Approach
  - Code Review for Client-side Validation Approach
  - Code Review for Non-parametrized SQL Query
  - Code Review for XSS Vulnerability
  - Code Review for Weak Password Authentication

- Code Review for Hard-coded Passwords
- Code Review for Empty Password in Connection String
- Code Review for Insecure Basic Authentication
- Code Review for Open Redirect
- Code Review for Insecure LDAP Authentication
- Code Review for Insecure Authorization Mechanism
- Code Review for Weak Password Length
- Code Review for use of Weak Hashing Algorithm
- Code Review for Use of Weak Random Number Generator
- Code Review for Use of Insecure PBE Work Factor
- Code Review for use of Weak Encryption Algorithm
- Code Review for Use of Insufficient Encryption Key Size
- Code Review for Unsafe Decoding
- Code Review for Use of SSL
- Code Review for OS Command Injection
- Code Review for LDAP Injection
- Code Review for XML Injection
- Code Review for Unsafe use of request parameter to execute SQL Query
- Code Review for Insecure File Upload
- Code Review for Directory Traversal
- Code Review for Sensitive Information Exposure
- Code Review for Sensitive Information Leakage
- Code Review for Generic Exception Throwing and Catching
- Code Review for Cookies Vulnerable to Client-side Scripts Attacks
- Code Review for Cookies Vulnerable to CSRF Attacks
- Code Review for Enabling Directory Listing

▪ Code Review: Check List Approach

- Sample Checklist
  - Input Validation
  - Authentication

        - Authorization
        - Session Management
        - Cryptography
        - Exception Handling
        - Logging
- SAST Finding
- SAST Report
    - SAST Reporting
- Dynamic Application Security Testing (DAST)
    - Types of DAST
        - Automated Application Vulnerability Scanning
        - Manual Application Penetration Testing
    - SAST Vs DAST
- Automated Application Vulnerability Scanning Tools
    - Web Application Security Scanners
        - WebInspect
        - IBM SecurityAppScan
    - Additional Web Application Vulnerability Scanners
- Proxy-based Security Testing Tools
    - Burp Suite
    - OWASP Zed Attack Proxy (ZAP)
    - Additional Proxy-based Security Testing Tools
- Choosing Between SAST and DAST

Module 10: Secure Deployment and Maintenance

- Secure Deployment
- Prior Deployment Activity
    - Check the Integrity of Application Package Before Deployment
    - Review the Deployment Guide Provided by the Software Vendor
- Deployment Activities: Ensuring Security at Various Levels
- Ensuring Security at Host Level

- Check and Configure the Security of Machine Hosting Web Server, Application Server, Database Server and Network Devices
- Physical Security
- Host Level Security

- Ensuring Security at Network Level
  - Network Level Security
    - Router
    - Firewall
    - Switch
- Ensuring Security at Application Level
  - Web Application Firewall (WAF)
    - Benefits of WAF
    - WAF Limitations
    - WAF Vendors
- Ensuring Security at Web Container Level (Tomcat)
  - Install and Configure Tomcat Securely
  - Remove Server Banner
  - Start Tomcat with Security Manager
  - Configure Default Servlet Not to Serve Index Pages
  - Replace Default Error Page
  - Replace Default server.xml
  - Protect Shutdown Port
  - Restrict Access to Tomcat Manager Applications
  - Protecting Resources with Realms
  - Store Passwords as Digest
  - Do Not Run Tomcat as Root
  - Configure Restricted Datasets
  - Session Handling using App Mode in Tomcat
  - Role Based Security
  - Securing Tomcat at Network level
  - Java Runtime Security Configurations

- Tomcat General Security Setting
- Verify Trace Element Setting in sever.xml
- Verify CustomError Settings in web.xml
- Verify maxPostSize Setting
- Tomcat Security Checklist
- Checklist for Security Configuration in server.xml File in Apache Tomcat
- Tomcat High Availability
- Best Practices for Securing Tomcat

▪ Ensuring Security in Oracle

- Oracle Database General Security Overview
- Methods of Authentication in Oracle
- Authentication by Oracle Database
- Oracle Security Features
- Default Database Installation and Configuration Security
- Managing User Accounts Securely for the Site
- Securing User Accounts
- Password Management
- Lock all Expired Accounts
- Assign Users to Password Profile
- Disable Remote Operating System Authentication
- Securing Data
- Restrict Access to Operating System Directories
- Securing Database Installation and Configuration
- Securing Network
- How to Configure Encryption on the Client and the Server
- Control Access Data
- Virtual Private Database
- Oracle Label Security
- Database Vault
    - Management and Reports

- Disabling the Recycle Bin
- Audit Vault
- Built-in Audit Tools
  - Standard Database Auditing
    - Standard Auditing Enable Network Auditing
  - Value Based Auditing
  - Fine Grained Auditing (FGA)
- Recommended Audit Settings

- Security Maintenance and Monitoring
  - Post Deployment Activities: Security Maintenance and Monitoring
  - Security Maintenance Activities at OS Level
  - Security Maintenance Activities at Web Container Level
  - Security Maintenance Activities at Application Level