Cyber Secure Coder (Exam CSC-210)

Course Length:

3 days

Overview:

The stakes for software security are very high, and yet many development teams deal with software security only after the code has been developed and the software is being prepared for delivery. As with any aspect of software quality, to ensure successful implementation, security and privacy issues should be managed throughout the entire software development lifecycle.

This course presents an approach for dealing with security and privacy throughout the entire software development lifecycle. You will learn about vulnerabilities that undermine security, and how to identify and remediate them in your own projects. You will learn general strategies for dealing with security defects and misconfiguration, how to design software to deal with the human element in security, and how to incorporate security into all phases of development.

Course Objectives:

In this course, you will employ best practices in software development to develop secure software.

You will:

- Identify the need for security in your software projects.
- Eliminate vulnerabilities within software.
- Use a Security by Design approach to design a secure architecture for your software.
- Implement common protections to protect users and data.
- Apply various testing methods to find and correct security defects in your software.
- Maintain deployed software to ensure ongoing security.

This course includes hands on activities for each topic area. The goal of these activities is to demonstrate concepts utilizing two universal languages Python and Java Script. Developers who use alternate languages will be able apply the principles from the activities to any coding languages.

Hands on exercises are designed to keep the typing of code to a bare minimum. CertNexus provides students with all of the code they need to complete activities. The activities do not require a "deep dive" into code to understand the principles being covered.

Target Student:

This course is designed for software developers, testers, and architects who design and develop software in various programming languages and platforms, including desktop, web, cloud, and mobile, and who want to improve their ability to deliver software that is of high quality, particularly regarding security and privacy.

This course is also designed for students who are seeking the CertNexus Cyber Secure Coder (CSC) Exam CSC-210 certification.

Prerequisites:

This course presents secure programming concepts that apply to many different types of software development projects. Although this course uses Python®, HTML, and JavaScript® to demonstrate various programming concepts, you do not need to have experience in these languages to benefit from this course. However, you should have some programming experience, whether it be developing desktop, mobile, web, or cloud applications. Logical Operations provides a variety of courses covering software development that you might use to prepare for this course, such as:

- Python® Programming: Introduction
- Python® Programming: Advanced
- HTML5: Content Authoring with New and Advanced Features
- SQL Querying: Fundamentals (Second Edition)

Course Content

*Orange text indicates activity

Lesson 1: Identifying the Need for Security in Your Software Projects

Topic A: Identify Security Requirements and Expectations

- Security Throughout the Development Process
- Business Requirements
- Standards and Compliance Requirements
- User Impact
- User Expectations
- Platform Requirements
- Consequences of Not Meeting Security Requirements
- Guidelines for Identifying Security Requirements and Expectations
- Identifying Security Requirements and Expectations

Topic B: Identify Factors That Undermine Software Security

- Three Ps of Software Security
- Software Security Terminology
- Identifying Factors That Undermine Security

Topic C: Find Vulnerabilities in Your Software

- Builders and Breakers
- Hacking
- Phases of an Attack
- Common Attack Patterns
- Case Study: Protecting Against a Password Attack
- Guidelines for Identifying Software Security Vulnerabilities
- Identifying Vulnerabilities in an Application
- Cracking a Password Hash
- Fixing a Password Hash Vulnerability

Topic D: Gather Intelligence on Vulnerabilities and Exploits

- Vulnerability Intelligence
- Exploits
- Guidelines for Researching Vulnerabilities and Exploits
- Identifying Sources for Vulnerability Intelligence

Lesson 2: Handling Vulnerabilities

Topic A: Handle Vulnerabilities Due to Software Defects and Misconfiguration

- Software Defects
- Causes of Software Defects
- Guidelines for Preventing Security Defects
- Preventing Security Defects
- Problems in Third-Party Code
- Problems in Standard Libraries
- Dependencies
- Encryption Validation
- Security of Host Systems and Service Providers
- Guidelines for Using Third-Party Code and Services
- Host Platform Configuration
- Hypervisor Vulnerabilities
- Guidelines for Managing Vulnerabilities in External Hosts and Services
- Identifying Vulnerabilities in a Software Project
- Examining the Project Files
- Error Messaging
- Error Handling
- Fail-Safe
- Failure Recovery
- Guidelines for Secure Error Handling
- Identifying Software Defects and Misconfiguration

Topic B: Handle Vulnerabilities Due to Human Factors

- The Human Element in Software Security
- Vulnerabilities Attributed to the Human Element
- Social Engineering Attacks
- User Input
- Input Validation
- Security Policy Enforcement
- Guidelines for Managing People Risks
- Managing People Risks

Topic C: Handle Vulnerabilities Due to Process Shortcomings

- Development Process Approaches
- Building Security In
- The CIA Triad
- Requirements Phase
- Design Phase
- Development Phase
- Testing Phase
- Security Testing Tools
- Deployment Phase
- Maintenance Phase
- Development Process Security
- Guidelines for Software Development Processes
- Managing Software Development Process Risks

Lesson 3: Designing for Security

Topic A: Apply General Principles for Secure Design

- Security in the Design Phase
- Security by Obscurity vs. Security by Design
- OWASP Security Design Principles
- Minimize Attack Surface Area
- Establish Secure Defaults
- Least Privilege
- Least Common Mechanism
- Defense in Depth
- Fail Securely
- Don't Trust Services
- Separation of Duties
- Security by Obscurity
- Keep Security Simple
- Fix Security Issues Correctly
- Software Design Patterns
- Security Patterns
- Modular Design
- Benefits of Modular Design
- The Balance Between Defense in Depth and Simplicity
- Guidelines for Avoiding Common Design Mistakes
- Avoiding Common Security Design Flaws

Topic B: Design Software to Counter Specific Threats

- The Risk Equation
- Threat Modeling
- Benefits of Threat Modeling
- Step 1: Define General Security Objectives and Scope
- Tooling and Documentation
- Assets
- Step 2: Decompose the Software
- Trust Levels
- Entry and Exit Points
- External Dependencies
- Data Flow Diagrams
- Diagramming Symbols
- Diagramming the Catalog Application
- Step 3: Identify and Rank Threats
- STRIDE
- PASTA
- Misuse Cases
- Security Zones
- Strategies for Ranking Threats
- DREAD
- Risk Response Strategies
- Severity
- Risks Outside Your Control

- Guidelines for Identifying and Ranking Threats
- Step 4: Counter Each Threat
- Countermeasures
- Identifying Threats and Countermeasures

Lesson 4: Developing Secure Code

Topic A: Follow Best Practices for Secure Coding

- Development Documentation and Deliverables
- Application and Data Integrity
- Common General Programming Errors
- Insecure Deserialization
- Guidelines for Secure Coding
- Researching Your Secure Coding Checklist
- Buffer Overrun Defects
- Buffer Overflows
- Guidelines to Prevent Buffer Overflow Defects
- Buffer Overreads
- Guidelines to Prevent Buffer Overread Defects
- Integer Overflows
- Guidelines to Prevent Integer Overflow Defects
- Uncontrolled Format Strings
- Insecure Output Encoding
- XXE Attacks
- Guidelines to Prevent Uncontrolled Format String Defects
- Race Condition
- Impact of Race Conditions on Threading/Multiprocessing
- Guidelines to Prevent Race Condition Defects

- Performing a Memory-Based Attack

Topic B: Prevent Platform Vulnerabilities

- OWASP Top Ten Platform Vulnerabilities
- Authentication
- Authorization
- Broken Authentication
- Guidelines to Prevent Web Vulnerability Defects
- Guidelines to Prevent Mobile App Vulnerability Defects
- Guidelines to Prevent Internet of Things Vulnerability Defects
- Desktop Application Vulnerabilities
- DLL Injection
- Shellcode Injection
- Debugger Security
- Differences Among Desktop Platforms
- Managed vs. Unmanaged
- Desktop Application Attack Vectors
- Development Tool and Project Configuration
- Guidelines to Prevent Desktop Application Vulnerabilities
- Finding Common Web Vulnerabilities

Topic C: Prevent Privacy Vulnerabilities
- Privacy Vulnerability Defects
- Privacy by Design
- Data Anonymization
- Guidelines to Prevent Privacy Vulnerability Defects
- Handling Privacy Defects

Lesson 5: Implementing Common Protections
Topic A: Limit Access Using Login and User Roles
- Web Sessions
- Secure Session Management
- Methods for Passing Session IDs
- Access Control
- Guidelines for Secure Session Management
- User Provisioning
- Password Recovery
- Account Lockouts
- Guidelines for Secure Password Management
- Handling Authentication and Authorization Defects

Topic B: Protect Data in Transit and At Rest
- Encryption
- Uses for Encryption
- Cryptographic Lifecycle
- Symmetric Encryption
- Asymmetric Encryption
- Hashing
- Digital Signatures
- Digital Signature Non-repudiation
- Digital Certificates
- PKI
- PKI Components
- The PKI Process
- Key Management
- Key Management Factors
- Certificate Revocation
- Guidelines for Protecting Data in Transit and at Rest
- Protecting Data in Transit and at Rest

Topic C: Implement Error Handling and Logging
- Error Handling
- Uses for Error Handling
- Error Messaging
- Logging
- Guidelines for Implementing Error Handling and Logging
- Reviewing Error Handling
- Improving Error Handling

Topic D: Protect Sensitive Data and Functions
- Sensitive Data
- Output Restrictions
- Function Level Access Control
- Case Study: Cross-Site Scripting Defect
- Guidelines for Protecting Sensitive Data and Functions
- Protecting Sensitive Data and Functions
- Staging a Persisted XSS Attack on an Administrator Function

Topic E: Protect Database Access
- Case Study: SQL Injection Defect
- Query Parameterization
- Database Connection Credential Protection
- Guidelines for Protecting Database Access
- Protecting Database Access

Lesson 6: Testing Software Security

Topic A: Perform Security Testing
- The Role of Testing
- Phases of Software Testing
- Development Testing
- Unit Testing
- Integration Testing
- Documentation and Deliverables for Testing
- Manual Inspection and Code Review
- Code Review Strategies
- Guidelines for Security Testing
- Performing Manual Inspection and Review

Topic B: Analyze Code to find Security Problems
- Static Code Analysis
- Strategies for Using Static Analysis
- Dynamic Code Analysis
- Guidelines for Code Analysis
- Performing Code Analysis

Topic C: Use Automated Testing Tools to Find Security Problems
- Automated Testing
- Unit Testing
- Guidelines for Using Automated Testing Tools
- Using a Test Suite to Automate Unit Testing

Lesson 7: Maintaining Security in Deployed Software

Topic A: Monitor and Log Applications to Support Security
- Emerging Security Problems
- Situational Awareness
- Security Monitoring
- Intrusion Detection and Prevention
- Monitor Placement
- Logging
- Guidelines for Monitoring and Logging a Deployed Application
- Monitoring and Logging a Deployed Application

Course-specific Technical Requirements

Hardware

For this course, you will need one computer for each student and one for the instructor. Each computer will need the following minimum hardware configurations:

- 1 GHz or faster 32-bit (x86) or 64-bit (x64) processor
- 2 gigabytes (GB) RAM (32-bit or 64-bit)
- 20 GB available hard disk space (32-bit or 64-bit)
- Keyboard and mouse (or other pointing device)
- 1,024 x 768 resolution monitor recommended
- Projection system to display the instructor's computer screen
- Local area network and Internet connection

Software

- Windows® 10/8.1/8/7/Vista (64-bit). This course was successfully keyed on Windows 10. Some activity steps may not key exactly as written if students key on a different version of Windows.
- Python version 3.8.1 (python-3.8.1.amd64.msi, provided with the course data files).
- PyCharm Community Edition version 2019.3.3 (pycharm-community-2019.3.3.exe, provided with the course data files). Python is distributed under the Python Software Foundation License (PSFL). PyCharm Community Edition is distributed under the Apache® License 2.0.
- If necessary, software for viewing the course slides. (Instructor machine only.)