TEL: +971 4 246 2843
Office 3832, 38th Fl.
The One Tower
Sheikh Zayed Road
Barsha Heights
Dubai - UAE

Certified Secure Software Lifecycle Professional CSSLP

- **Course Overview:**

CSSLP certification recognizes leading application security skills. It shows employers and peers you have the advanced technical skills and knowledge necessary for authentication, authorization and auditing throughout the SDLC using best practices, policies and procedures established by the cybersecurity experts at (ISC)².

Duration: 5 Days

Exam: CSSLP

- **Target Audience:**

  - Software Architect
  - Software Engineer
  - Software Developer
  - Application Security Specialist
  - Software Program Manager
  - Quality Assurance Tester
  - Penetration Tester
  - Software Procurement Analyst
  - Project Manager
  - Security Manager
  - IT Director/Manager

- **Learning Objectives:**

Upon Completion of this Course, you will accomplish following:

- Creating an application security program in organization
- Dropping production costs, delivery delays and application vulnerabilities
- Increasing the integrity of an organization
- Reducing loss of income due to a breach resulting from insecure software

- **Course Content:**

Secure Software Concepts:

- Core Concepts
- Security Design Principles

Secure Software requirements:

- Define Software Security Requirements
- Identify and Analyze Compliance Requirements
- Identify and Analyze Data Classification Requirements
- Identify and Analyze Privacy Requirements
- Develop Misuse and Abuse Cases
- Develop Security Requirement Traceability Matrix (STRM)
- Ensure Security Requirements Flow Down to Suppliers/Providers

Secure Software Architecture Design:

- Perform Threat Modeling
- Define the Security Architecture
- Performing Secure Interface Design

- Performing Architectural Risk Assessment
- Model (Non-Functional) Security Properties and Constraints
- Model and Classify Data
- Evaluate and Select Reusable Secure Design
- Perform Security Architecture and Design Review
- Define Secure Operational Architecture (e.g., deployment topology, operational interfaces)
- Use Secure Architecture and Design Principles, Patterns, and Tools

Secure Software Implementation:

- Adhere to Relevant Secure Coding Practices (e.g., standards, guidelines and regulations)
- Analyze Code for Security Risks
- Implement Security Controls (e.g., watchdogs, File Integrity Monitoring (FIM), anti-malware)
- Address Security Risks (e.g. remediation, mitigation, transfer, accept)
- Securely Reuse Third-Party Code or Libraries (e.g., Software Composition Analysis (SCA))
- Securely Integrate Components
- Apply Security During the Build Process

Secure Software Testing:

- Develop Security Test Cases
- Develop Security Testing Strategy and Plan
- Verify and Validate Documentation (e.g., installation and setup instructions, error messages, user guides, release notes)
- Identify Undocumented Functionality
- Analyze Security Implications of Test Results (e.g., impact on product management, prioritization, break build criteria)
- Classify and Track Security Errors
- Secure Test Data

Secure Software Lifecycle Management:

- Secure Configuration and Version Control (e.g., hardware, software, documentation, interfaces, patching)
- Define Strategy and Roadmap
- Manage Security Within a Software Development Methodology
- Identify Security Standards and Frameworks
- Define and Develop Security Documentation
- Develop Security Metrics (e.g., defects per line of code, criticality level, average remediation time, complexity)
- Decommission Software
- Report Security Status (e.g., reports, dashboards, feedback loops)
- Incorporate Integrated Risk Management (IRM)
- Promote Security Culture in Software Development
- Implement Continuous Improvement (e.g., retrospective, lessons learned)