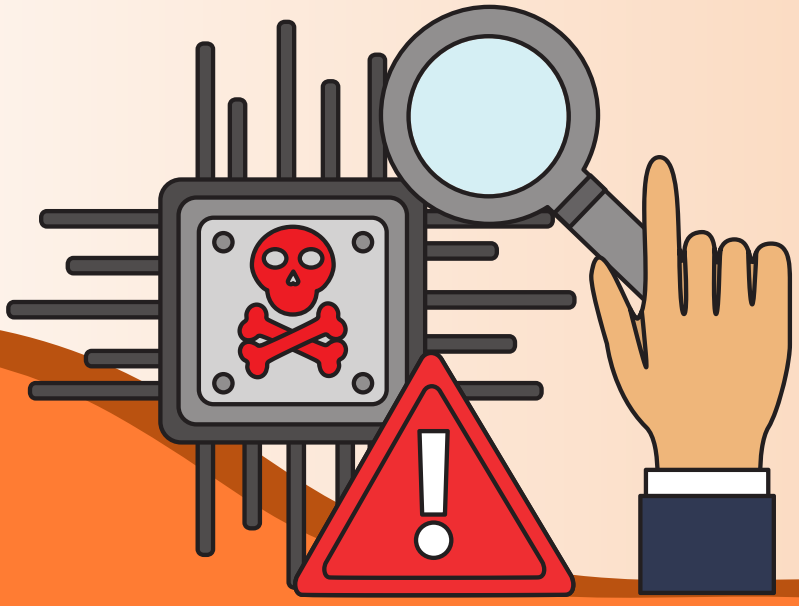
SPRINTZEAL
Empowering Professionals

“CERTIFIED SECURE SOFTWARE LIFECYCLE PROFESSIONAL (CSSLP)”

Course Agenda



Course Overview:

Program Overview

Obtaining the CSSLP secure software development certification is an excellent way to advance your career and increase your ability to incorporate security into all stages of the software development process. This certification demonstrates your expertise in application security. It informs employers and coworkers that you have extensive technical understanding in areas like authentication, authorization, and auditing. With this certification, you may demonstrate your skills, develop your career, and join a network of cybersecurity leaders who are eager to help you on your professional journey.

The syllabus you will be learning

The CSSLP Common Body of Knowledge (CBK®) covers a wide range of topics, ensuring its relevance across all disciplines of information security. Successful candidates demonstrate competence in the following eight domains:

Key Learning

- Confidentiality (e.g., encryption)
- Integrity (e.g., hashing, digital signatures, code signing, reliability, modifications, authenticity)
- Availability (e.g., redundancy, replication, clustering, scalability, resiliency)
- Authentication (e.g., multi-factor authentication (MFA), identity & access management (IAM), single sign-on (SSO), federated identity, biometrics)
- Authorization (e.g., access controls, permissions, entitlements)
- Accountability (e.g., auditing, logging)
- Nonrepudiation (e.g., digital signatures, block chain)
- Governance, risk and compliance (GRC) standards (e.g., regulatory authority, legal, industry)

Module 1.2 - Understand security design principles

- Least privilege (e.g., access control, need-to-know, run-time privileges, Zero Trust)
- Segregation of Duties (SoD) (e.g., multi-party control, secret sharing, split knowledge)
- Defense in depth (e.g., layered controls, geographical diversity, technical diversity, distributed systems)
- Resiliency (e.g., fail safe, fail secure, no single point of failure, failover)
- Economy of mechanism (e.g., single sign-on (SSO), password vaults, resource efficiency)
- Complete mediation (e.g., cookie management, session management, caching of credentials)
- Open design (e.g., Kerckhoffs's principle, peer review, open source, crowd source)
- Least common mechanism (e.g., compartmentalization/isolation, allow/accept list)
- Psychological acceptability (e.g., password complexity, passwordless authentication, screen layouts, Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA))
- Component reuse (e.g., common controls, libraries)

Module 2: Secure Software Lifecycle Management

Module 2.1 - Manage security within a software development methodology (e.g., Agile, waterfall)

Module 2.2 - Identify and adopt security standards (e.g., implementing security frameworks, promoting security awareness)

Module 2.3 - Outline strategy and roadmap
Security milestones and checkpoints (e.g., control gate, break/build criteria)

Module 2.4 - Define and develop security documentation
Module 2.5 - Define security metrics (e.g., criticality level, average remediation time, complexity, Key Performance Indicators (KPI), objectives and key results)

Module 2.6 - Decommission applications

- End of Life (EOL) policies (e.g., credential removal, configuration removal, license cancellation, archiving, service-level agreements (SLA))
- Data disposition (e.g., retention, destruction, dependencies)

Module 2.7 - Create security reporting mechanisms (e.g., reports, dashboards, feedback loops)

Module 2.8 - Incorporate integrated risk management methods

- Regulations, standards and guidelines (e.g., International Organization for Standardization (ISO), Payment Card Industry (PCI), National Institute of Standards and Technology (NIST), Open Web Application Security Project (OWASP), Software Assurance Forum for Excellence in Code (SAFECode), Software Assurance Maturity Model (SAMM), Building Security in Maturity Model (BSIMM))
- Legal (e.g., intellectual property, breach notification)
- Risk management (e.g., risk assessment, risk analysis)
- Technical risk vs. business risk

Module 2.9 - Implement secure operation practices

- Change management process
- Incident response plan
- Verification and validation
- Assessment and Authorization (A&A) process

Module 3: Secure Software Requirements

Module 3.1 - Define software security requirements

- Functional (e.g., business requirements, use cases, stories)
- Non-functional (e.g., security, operational, continuity, deployment)

Module 3.2 - Identify compliance requirements

- Regulatory authority
- Legal
- Industry-specific (e.g., defense, healthcare, commercial, financial, Payment Card Industry (PCI))
- Company-wide (e.g., development tools, standards, frameworks, protocols)

Module 3.3 - Identify data classification requirements

- Data ownership (e.g., data dictionary, data owner, data custodian)
- Data labeling (e.g., sensitivity, impact)
- Data types (e.g., structured, unstructured)
- Data lifecycle (e.g., generation, storage, retention, disposal)
- Data handling (e.g., personally identifiable information (PII), publicly available information)

Module 3.4 - Identify privacy requirements

- Data collection scope
- Data anonymization (e.g., pseudo-anonymous, fully anonymous)
- User rights (legal) and preferences (e.g., data disposal, right to be forgotten, marketing preferences, sharing and using third parties, terms of service)
- Data retention (e.g., how long, where, what)
- Cross-border requirements (e.g., data residency, jurisdiction, multi-national data processing boundaries)

Module 3.5 - Define data access provisioning

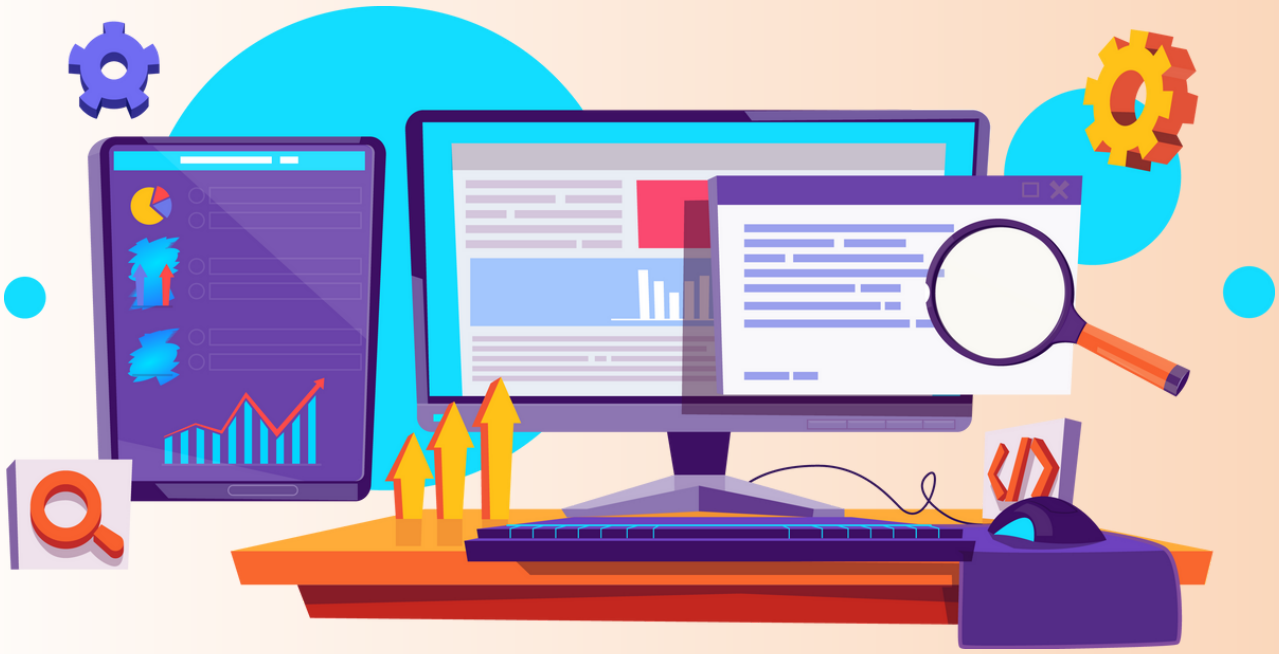
- User provisioning
- Service accounts
- Reapproval process

Module 3.6 - Develop misuse and abuse

Mitigating control identification

Module 3.7 - Develop security requirement traceability matrix

Module 3.8 - Define third-party vendor security requirements



Module 4: Secure Software Architecture and Design

Module 4.1 - Define the security architecture

- Secure architecture and design patterns (e.g., Sherwood Applied Business Security Architecture (SABSA), security chain of responsibility, federated identity)
- Security controls identification and prioritization
- Distributed computing (e.g., client server, peer-to-peer (P2P), message queuing, N-tier)
- Service-oriented architecture (SOA) (e.g., enterprise service bus, web services, microservices)
- Rich internet applications (e.g., client-side exploits or threats, remote code execution, constant connectivity)
- Pervasive/ubiquitous computing (e.g., Internet of Things (IoT), wireless, location-based, Radio-Frequency Identification (RFID), Near Field Communication (NFC), sensor networks, mesh)
- Embedded software (e.g., secure boot, secure memory, secure update)
- Cloud architectures (e.g., Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS))
- Mobile applications (e.g., implicit data collection privacy)
- Hardware platform concerns (e.g., side-channel mitigation, speculative execution mitigation, secure element, firmware, drivers)
- Cognitive computing (e.g., artificial intelligence (AI), virtual reality, augmented reality)
- Industrial Internet of Things (IoT) (e.g., facility-related, automotive, robotics, medical devices, software-defined production processes)

Module 4.2 - Perform secure interface design

- Security management interfaces, out-of-band management, log interfaces
- Upstream/downstream dependencies (e.g., key and data sharing between apps)
- Protocol design choices (e.g., application programming interfaces (API), weaknesses, state, models)

Module 4.3 - Evaluate and select reusable technologies

- Credential management (e.g., X.509, single sign-on (SSO))
- Flow control (e.g., proxies, firewalls, protocols, queuing)
- Data loss prevention (DLP)
- Virtualization (e.g., Infrastructure as code (IaC), hypervisor, containers)
- Trusted computing (e.g., Trusted Platform Module (TPM), Trusted Computing Base (TCB))
- Database security (e.g., encryption, triggers, views, privilege management, secure connections)
- Programming language environment (e.g., common language runtime, Java virtual machine (VM), Python, PowerShell)
- Operating system (OS) controls and services
- Secure backup and restoration planning
- Secure data retention, retrieval, and destruction

Module 4.4 - Perform threat modeling

- Threat modeling methodologies (e.g., Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege (STRIDE), Process for Attack Simulation and Threat Analysis (PASTA), Hybrid Threat Modeling Method, Common Vulnerability Scoring System (CVSS))
- Common threats (e.g., advanced persistent threat (APT), insider threat, common malware, third-party suppliers)
- Attack surface evaluation
- Threat analysis
- Threat intelligence (e.g., identify credible relevant threats, predict)

Module 4.5 - Perform architectural risk assessment and design reviews

Module 4.6 - Model (non-functional) security properties and constraints

Module 4.7 - Define secure operational architecture (e.g., deployment topology, operational interfaces, Continuous Integration and Continuous Delivery (CI/CD))

SPRINTZEAL
Empowering Professionals

Module 5: Secure Software Implementation

Module 5.1 - Adhere to relevant secure coding practices (e.g., standards, guidelines, regulations)

- Declarative versus imperative (programmatic) security
- Concurrency (e.g., thread safety, database concurrency controls)
- Input validation and sanitization
- Error and exception handling
- Output sanitization (e.g., encoding, obfuscation)
- Secure logging & auditing (e.g., confidentiality, privacy)
- Session management
- Trusted/untrusted application programming interfaces (API), and libraries
- Resource management (e.g., compute, storage, network, memory management)
- Secure configuration management (e.g., baseline security configuration, credentials management)
- Tokenization
- Isolation (e.g., sandboxing, virtualization, containerization, Separation Kernel Protection Profiles)
- Cryptography (e.g., payload, field level, transport, storage, agility, encryption, algorithm selection)
- Access control (e.g., trust zones, function permissions, role-based access control (RBAC), discretionary access control (DAC), mandatory access control (MAC))
- Processor microarchitecture security extensions

Module 5.2 - Analyze code for security risks

- Secure code reuse
- Vulnerability databases/lists (e.g., Open Web Application Security Project (OWASP) Top 10, Common Weakness Enumerations (CWE), SANS Top 25 Most Dangerous Software Errors)
- Static application security testing (SAST) (e.g., automated code coverage, linting)
- Manual code review (e.g., peer review)
- Inspect for malicious code (e.g., backdoors, logic bombs, high entropy)

Module 5.3 - Implement security controls (e.g., watchdogs, file integrity monitoring, anti-malware)

Module 5.4 - Address the identified security risks (e.g., risk strategy)

Module 5.5 - Evaluate and integrate components

- Systems-of-systems integration (e.g., trust contracts, security testing, analysis)
- Reusing third-party code or open-source libraries in a secure manner (e.g., software composition analysis)

Module 5.6 - Apply security during the build process

- Anti-tampering techniques (e.g., code signing, obfuscation)
- Compiler switches
- Address compiler warnings

Module 6: Secure Software Testing

Module 6.1 - Develop security testing strategy & plan

- Standards (e.g., International Organization for Standardization (ISO), Open Source Security Testing Methodology Manual, Software Engineering Institute)
- Functional security testing (e.g., logic)
- Nonfunctional security testing (e.g., reliability, performance, scalability)
- Testing techniques (e.g., known environment testing, unknown environment testing, functional testing, acceptance testing)
- Testing environment (e.g., interoperability, test harness)
- Security researcher outreach (e.g., bug bounties)

Module 6.2 - Develop security test cases

- Attack surface validation
- Automated vulnerability testing (e.g., dynamic application security testing (DAST), interactive application security testing (IAST))
- Penetration tests (e.g., security controls, known vulnerabilities, known malware)
- Fuzzing (e.g., generated, mutated)
- Simulation (e.g., simulating production environment and production data, synthetic transactions)
- Failure (e.g., fault injection, stress testing, break testing))
- Cryptographic validation (e.g., pseudorandom number generators, entropy)
- Unit testing and code coverage
- Regression tests
- Integration tests
- Continuous testing
- Misuse and abuse test cases

Module 6.3 - Verify and validate documentation (e.g., installation and setup instructions, error messages, user guides, release notes)

Module 6.4 - Identify undocumented functionality

Module 6.5 - Analyze security implications of test results (e.g., impact on product management, prioritization, break/build criteria)

Module 6.6 - Classify and track security errors

- Bug tracking (e.g., defects, errors and vulnerabilities)
- Risk scoring (e.g., Common Vulnerability Scoring System (CVSS))

Module 6.7 - Secure test data

- Generate test data (e.g., referential integrity, statistical quality, production representative)
- Reuse of production data (e.g., obfuscation, sanitization, anonymization, tokenization, data aggregation mitigation)

Module 6.8 - Perform verification and validation testing (e.g., independent/internal verification and validation, acceptance test

Module 7: Secure Software Deployment, Operations, Maintenance

Module 7.1 - Perform operational risk analysis

- Deployment environment (e.g., staging, production, quality assurance (QA))
- Personnel training (e.g., administrators vs. users)
- Legal compliance (e.g., adherence to guidelines, regulations, privacy laws, copyright, etc.)
- System integration

Module 7.2 - Secure configuration and version control

- Hardware
- Baseline configuration
- Version control/patching
- Documentation practices

Module 7.3 - Release software securely

- Secure Continuous Integration and Continuous Delivery (CI/CD) pipeline (e.g., DevSecOps)
- Application security toolchain
- Build artifact verification (e.g., code signing, hashes)

Module 7.4 - Store and manage security data

- Credentials
- Secrets
- Keys/certificates
- Configurations

Module 7.5 - Ensure secure installation

- Secure boot (e.g., key generation, access, management)
- Least privilege
- Environment hardening (e.g., configuration hardening, secure patch/updates, firewall)
- Secure provisioning (e.g., credentials, configuration, licensing, Infrastructure as code (IaC))
- Security policy implementation

Module 7.6 - Obtain security approval to operate (e.g., risk acceptance, sign-off at appropriate level)

Module 7.7 - Perform information security continuous monitoring

- Observable data (e.g., logs, events, telemetry, trace data, metrics)
- Threat intelligence
- Intrusion detection/response
- Regulation and privacy changes
- Integration analysis (e.g., security information and event management (SIEM))

Module 7.8 - Execute the incident response plan

- Incident triage
- Forensics
- Remediation
- Root cause analysis

Module 7.9 - Perform patch management (e.g. secure release, testing)

Module 7.10 - Perform vulnerability management (e.g., tracking, triaging, Common Vulnerabilities and Exposures (CVE))

Module 7.11 - Incorporate runtime protection (e.g., Runtime Application Self Protection (RASP), web application firewall

(WAF), Address Space Layout Randomization (ASLR), dynamic execution prevention)

Module 7.12 - Support continuity of operations Backup, archiving, retention

- Disaster recovery plan (DRP)
- Resiliency (e.g., operational redundancy, erasure code, survivability, denial-of-service (DoS))
- Business continuity plan (BCP)

Module 7.13 - Integrate service level objectives and service-level agreements (SLA) (e.g., maintenance, performance, availability, qualified personnel)

Module 8: Secure Software Supply Chain

Module 8.1 - Implement software supply chain risk management (e.g., International Organization for Standardization (ISO), National Institute of Standards and Technology (NIST))

- Identification and selection of the components
- Risk assessment of the components (e.g., mitigate, accept)
- Maintaining third-party components list (e.g., software bill of materials)
- Monitoring for changes and vulnerabilities

Module 8.2 - Analyze security of third-party software

- Certifications
- Assessment reports (e.g., cloud controls matrix)
- Origin and support

Module 8.3 - Verify pedigree and provenance

- Secure transfer (e.g., chain of custody, authenticity, integrity)
- System sharing/interconnections
- Code repository security
- Build environment security
- Cryptographically-hashed, digitally-signed components
- Right to audit

Module 8.4 - Ensure and verify supplier security requirements in the acquisition process

- Audit of security policy compliance (e.g., secure software development practices)
- Vulnerability/incident notification, response, coordination, and reporting
- Maintenance and support structure (e.g., community versus commercial, licensing)
- Security track record
- Scope of testing (e.g., shared responsibility model)
- Log integration into security information and event management (SIEM)

Module 8.5 - Support contractual requirements (e.g., intellectual property ownership, code escrow, liability, warranty, End-User License Agreement (EULA), service-level agreements (SLA))

About Sprintzeal's "Certified Secure Software Lifecycle Professional (CSSLP)" Training

Sprintzeal CSSLP secure software development certification enhances career advancement by integrating security into software development stages. It showcases expertise in application security, demonstrating understanding of authentication, authorization, and auditing. The comprehensive syllabus covers topics relevant to information security, equipping students with tools and techniques for success in the ever-evolving cybersecurity world.