EDUSUM

#1 Online Certification Guide

ISC2 CSSLP

ISC2 Secure Software Lifecycle Professional Certification Questions & Answers

Exam Summary – Syllabus –Questions

CSSLP

ISC2 Certified Secure Software Lifecycle Professional (CSSLP)

125 Questions Exam – 700/1000 Cut Score – Duration of 180 minutes

Table of Contents:

Know Your CSSLP Certification Well: 2

ISC2 CSSLP Secure Software Lifecycle Professional Certification Details: 2

CSSLP Syllabus: 3

Secure Software Concepts - 10% 3
Secure Software Requirements - 14% 4
Secure Software Architecture and Design - 14% 4
Secure Software Implementation - 14% 6
Secure Software Testing - 14% 7
Secure Software Lifecycle Management - 11% 9
Secure Software Deployment, Operations, Maintenance - 12% 10
Secure Software Supply Chain - 11% 12

ISC2 CSSLP Sample Questions: 13

Study Guide to Crack ISC2 Secure Software Lifecycle Professional CSSLP Exam: 15

Know Your CSSLP Certification Well:

The CSSLP is best suitable for candidates who want to gain knowledge in the ISC2 Cybersecurity. Before you start your CSSLP preparation you may struggle to get all the crucial Secure Software Lifecycle Professional materials like CSSLP syllabus, sample questions, study guide.

But don't worry the CSSLP PDF is here to help you prepare in a stress free manner. The PDF is a combination of all your queries like-

- What is in the CSSLP syllabus?
- How many questions are there in the CSSLP exam?
- Which Practice test would help me to pass the CSSLP exam at the first attempt?

Passing the CSSLP exam makes you ISC2 Certified Secure Software Lifecycle Professional (CSSLP). Having the Secure Software Lifecycle Professional certification opens multiple opportunities for you. You can grab a new job, get a higher salary or simply get recognition within your current organization.

ISC2 CSSLP Secure Software Lifecycle Professional Certification Details:

Exam Name	ISC2 Certified Secure Software Lifecycle Professional (CSSLP)
Exam Code	CSSLP
Exam Price	$599 (USD)
Duration	180 mins
Number of Questions	125
Passing Score	700/1000
Schedule Exam	**Pearson VUE**
Sample Questions	**ISC2 CSSLP Sample Questions**
Practice Exam	**ISC2 CSSLP Certification Practice Exam**

CSSLP Syllabus:

Topic	Details
Secure Software Concepts - 10%	
Core Concepts	- Confidentiality (e.g., covert, overt, encryption) - Integrity (e.g., hashing, digital signatures, code signing, reliability, modifications, authenticity) - Availability (e.g., redundancy, replication, clustering, scalability, resiliency) - Authentication (e.g., multifactor authentication (MFA), identity & access management (IAM), single sign-on (SSO), federated identity) - Authorization (e.g., access controls, permissions, entitlements) - Accountability (e.g., auditing, logging) - Nonrepudiation (e.g., digital signatures, block chain)
Security Design Principles	- Least privilege (e.g., access control, need-to-know, run-time privileges) - Separation of duties (e.g., multi-party control, secret sharing and split knowledge) - Defense in depth (e.g., layered controls, input validation, security zones) - Resiliency (e.g., fail safe, fail secure, no Single Point of Failure (SPOF)) - Economy of mechanism (e.g., Single Sign-On (SSO), password vaults, resource) - Complete mediation (e.g., cookie management, session management, caching of credentials) - Open design (e.g., Kerckhoffs's principle) - Least common mechanism (e.g., compartmentalization/isolation, white-listing) - Psychological acceptability (e.g., password complexity, screen layouts, Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA), biometrics) - Component reuse (e.g., common controls, libraries) - Diversity of defense (e.g., geographical diversity, technical diversity, distributed systems)

Topic	Details
Secure Software Requirements - 14%	
Define Software Security Requirements	- Functional (e.g., business requirements, use cases, stories) - Non-functional (e.g., operational, deployment, systemic qualities)
Identify and Analyze Compliance Requirements	
Identify and Analyze Data Classification Requirements	- Data ownership (e.g., data owner, data custodian) - Labeling (e.g., sensitivity, impact) - Types of data (e.g., structured, unstructured data) - Data life-cycle (e.g., generation, retention, disposal)
Identify and Analyze Privacy Requirements	- Data anonymization - User consent - Disposition (e.g., right to be forgotten) - Data retention - Cross borders (e.g., data residency, jurisdiction, multi-national data processing boundaries)
Develop Misuse and Abuse Cases	
Develop Security Requirement Traceability Matrix (STRM)	
Ensure Security Requirements Flow Down to Suppliers/Providers	
Secure Software Architecture and Design - 14%	
Perform Threat Modeling	- Understand common threats (e.g., Advance Persistent Threat (APT), insider threat, common malware, third-party/supplier) - Attack surface evaluation - Threat intelligence (e.g., Identify credible relevant threats)
Define the Security Architecture	- Security control identification and prioritization - Distributed computing (e.g., client server, peer-to-peer (P2P), message queuing) - Service-oriented architecture (SOA) (e.g., Enterprise Service Bus (ESB), web services) - Rich internet applications (e.g., client-side exploits or threats,

Topic	Details
	remote code execution, constant connectivity) - Pervasive/ubiquitous computing (e.g., Internet of Things (IoT), wireless, location-based, Radio-Frequency Identification (RFID), near field communication, sensor networks) - Embedded (e.g., secure update, Field-Programmable Gate Array (FPGA) security features, microcontroller security) - 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, embedded Hardware Security Modules (HSM)) - Cognitive computing (e.g., Machine Learning (ML), Artificial Intelligence (AI)) - Control systems (e.g., industrial, medical, facility-related, automotive)
Performing Secure Interface Design	- Security management interfaces, Out-of-Band (OOB) management, log interfaces - Upstream/downstream dependencies (e.g., key and data sharing between apps) - Protocol design choices (e.g., Application Programming Interface (APIs), weaknesses, state, models)
Performing Architectural Risk Assessment	
Model (Non-Functional) Security Properties and Constraints	
Model and Classify Data	
Evaluate and Select Reusable Secure Design	- Credential management (e.g., X.509 and Single Sign-On (SSO)) - Flow control (e.g., proxies, firewalls, protocols, queuing) - Data loss prevention (DLP) - Virtualization (e.g., software defined infrastructure, hypervisor, containers) - Trusted computing (e.g., Trusted Platform Module (TPM), Trusted Computing Base (TCB)) - Database security (e.g., encryption, triggers, views, privilege management) - Programming language environment (e.g., Common Language

Topic	Details
	Runtime (CLR), Java Virtual Machine (JVM)) - Operating System (OS) controls and services - Secure backup and restoration planning - Secure data retention, retrieval, and destruction
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 - 14%	
Adhere to Relevant Secure Coding Practices (e.g., standards, guidelines and regulations)	- Declarative versus imperative (programmatic) security - Concurrency (e.g., thread safety, database concurrency controls) - Output sanitization (e.g., encoding, obfuscation) - Error and exception handling - Input validation - Secure logging & auditing - Session management - Trusted/Untrusted Application Programming Interface (APIs), and libraries - Type safety - Resource management (e.g., compute, storage, network, memory management) - Secure configuration management (e.g., parameter, default options, credentials) - Tokenizing - Isolation (e.g., sandboxing, virtualization, containers, Separation Kernel Protection Profiles (SKPP)) - 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))

Topic	Details
	- Processor microarchitecture security extensions (e.g., Software Guard Extensions (SGX), Advanced Micro Devices (AMD) Secure Memory Encryption(SME)/Secure Encrypted Virtualization(SEV), ARM TrustZone)
Analyze Code for Security Risks	- Secure code reuse - Vulnerability databases/lists (e.g., Open Web Application Security Project (OWASP) Top 10, Common Weakness Enumeration (CWE)) - Static Application Security Testing (SAST) (e.g., automated code coverage, linting) - Dynamic Application Security Testing (DAST) - Manual code review (e.g., individual, peer) - Look for malicious code (e.g., backdoors, logic bombs, high entropy) - Interactive Application Security Testing (IAST)
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	- Systems-of-systems integration (e.g., trust contracts, security testing and analysis)
Apply Security During the Build Process	- Anti-tampering techniques (e.g., code signing, obfuscation) - Compiler switches - Address compiler warnings
Secure Software Testing - 14%	
Develop Security Test Cases	- Attack surface validation - Penetration tests - Fuzzing (e.g., generated, mutated)

Topic	Details
	- Scanning (e.g., vulnerability, content, privacy) - Simulation (e.g., simulating production environment and production data, synthetic workloads) - Failure (e.g., fault injection, stress testing, break testing) - Cryptographic validation (e.g., Pseudo-Random Number Generator (PRNG), entropy) - Regression tests - Integration tests - Continuous (e.g., synthetic transactions)
Develop Security Testing Strategy and Plan	- Functional security testing (e.g., logic) - Nonfunctional security testing (e.g., reliability, performance, scalability) - Testing techniques (e.g., white box and black box) - Environment (e.g., interoperability, test harness) - Standards (e.g., International Organization for Standardization (ISO), Open Source Security Testing Methodology Manual (OSSTMM), Software Engineering Institute (SEI)) - Crowd sourcing (e.g., bug bounty)
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	- Bug tracking (e.g., defects, errors and vulnerabilities) - Risk Scoring (e.g., Common Vulnerability Scoring System (CVSS))
Secure Test Data	- Generate test data (e.g., referential integrity, statistical quality, production representative)

Topic	Details
	- Reuse of production data (e.g., obfuscation, sanitization, anonymization, tokenization, data aggregation mitigation)
Perform Verification and Validation Testing	
Secure Software Lifecycle Management - 11%	
Secure Configuration and Version Control (e.g., hardware, software, documentation, interfaces, patching)	
Define Strategy and Roadmap	
Manage Security Within a Software Development Methodology	- Security in adaptive methodologies (e.g., Agile methodologies) - Security in predictive methodologies (e.g., Waterfall)
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	- End of life policies (e.g., credential removal, configuration removal, license cancellation, archiving) - Data disposition (e.g., retention, destruction, dependencies)
Report Security Status (e.g., reports, dashboards, feedback loops)	
Incorporate Integrated Risk Management (IRM)	- Regulations and compliance - Legal (e.g., intellectual property, breach notification) - Standards and guidelines (e.g., International Organization for

Topic	Details
	Standardization (ISO), Payment Card Industry (PCI), National Institute of Standards and Technology (NIST), OWASP, Software Assurance Forum for Excellence in Code (SAFECode), Software Assurance Maturity Model (SAMM), Building Security In Maturity Model (BSIMM)) - Risk management (e.g., mitigate, accept, transfer, avoid) - Terminology (e.g., threats, vulnerability, residual risk, controls, probability, impact) - Technical risk vs. business risk
Promote Security Culture in Software Development	- Security champions - Security education and guidance
Implement Continuous Improvement (e.g., retrospective, lessons learned)	
Secure Software Deployment, Operations, Maintenance - 12%	
Perform Operational Risk Analysis	- Deployment environment - Personnel training (e.g., administrators vs. users) - Safety criticality - System integration
Release Software Securely	- Secure Continuous Integration and Continuous Delivery (CI/CD) pipeline - Secure software tool chain - Build artifact verification (e.g., code signing, checksums, hashes)
Securely Store and Manage Security Data	- Credentials - Secrets - Keys/certificates - Configurations
Ensure Secure Installation	- Bootstrapping (e.g., key generation, access, management) - Least privilege - Environment hardening - Secure activation (e.g., credentials, white listing, device configuration, network configuration, licensing) - Security policy implementation - Secrets injection (e.g., certificate, Open Authorization (OAUTH) tokens, Secure Shell (SSH) keys)

Topic	Details
Perform Post-Deployment Security Testing	
Obtain Security Approval to Operate (e.g., risk acceptance, sign-off at appropriate level)	
Perform Information Security Continuous Monitoring (ISCM)	- Collect and analyze security observable data (e.g., logs, events, telemetry, and trace data) - Threat intel - Intrusion detection/response - Secure configuration - Regulation changes
Support Incident Response	- Root cause analysis - Incident triage - Forensics
Perform Patch Management (e.g. secure release, testing)	
Perform Vulnerability Management (e.g., scanning, tracking, triaging)	
Runtime Protection (e.g., Runtime Application Self-Protection (RASP), Web Application Firewall (WAF), Address Space Layout Randomization (ASLR))	
Support Continuity of Operations	- Backup, archiving, retention - Disaster recovery (DR) - Resiliency (e.g., operational redundancy, erasure code, survivability)
Integrate Service Level Objectives	

Topic	Details
(SLO) and Service Level Agreements (SLA) (e.g., maintenance, performance, availability, qualified personnel)	
Secure Software Supply Chain - 11%	
Implement Software Supply Chain Risk Management	- Identify - Assess - Respond - Monitor
Analyze Security of Third-Party Software	
Verify Pedigree and Provenance	- Secure transfer (e.g., interdiction mitigation) - System sharing/interconnections - Code repository security - Build environment security - Cryptographically-hashed, digitally-signed components - Right to audit
Ensure 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
Support contractual requirements (e.g., Intellectual Property (IP) ownership, code escrow, liability, warranty, End-User License Agreement (EULA), Service Level Agreements (SLA))	

ISC2 CSSLP Sample Questions:

Question: 1

Which of the following is measured in dollars?

a) Exposure factor
b) SLE
c) ARO
d) Impact factor

Answer: b

Question: 2

Which of the following is not a mitigation method for threats identified in threat modeling?

a) Redesign to eliminate vulnerability.
b) Apply a standard mitigation.
c) Change the security requirements to eliminate the threat.
d) Accept the vulnerability.

Answer: c

Question: 3

An operational measure of what constitutes the minimum level of quality with respect to security in code is a description of:

a) ISO 9216 process element
b) OSSTMM report
c) Bug bar
d) SDL process requirement

Answer: c

Question: 4

The operations and management processes are lumped together into sustainment because:

a) They are at the end of the lifecycle.
b) They are the major activities during the software use lifecycle period.
c) They are neither development nor acquisition.
d) They are strictly control processes for sustaining assurance.

Answer: b

Question: 5

The repository where the current baseline is preserved is called the:

a) Controlled repository
b) Dynamic repositary
c) Archive repository
d) Master repository

Answer: a

Question: 6

Verifying that code can perform in a particular manner under production conditions is a task managed by:

a) Static code analysis
b) Dynamic code analysis
c) Production testing
d) Code walkthroughs

Answer: b

Question: 7

Data classification is performed at which stage of the lifecycle model?

a) Data retention
b) Disposal
c) Generation
d) Data reduction

Answer: c

Question: 8

A common language to describe and exchange information about the causes of software vulnerabilities is:

a) CVS
b) CVE
c) CSSLP
d) CNSS

Answer: b

Question: 9

The ability of an application to restore itself to expected functionality after the security protection is breached or bypassed is called:

a) Resilience
b) Recoverability
c) Reliability
d) Restoration

Answer: b

Question: 10

An international standard for establishing quality in software products is:

a) ISO 9000
b) ISO 27001
c) ISO 21827
d) ISO 9216

Answer: d

Study Guide to Crack ISC2 Secure Software Lifecycle Professional CSSLP Exam:

- Getting details of the CSSLP syllabus, is the first step of a study plan. This pdf is going to be of ultimate help. Completion of the syllabus is must to pass the CSSLP exam.
- Making a schedule is vital. A structured method of preparation leads to success. A candidate must plan his schedule and follow it rigorously to attain success.
- Joining the ISC2 provided training for CSSLP exam could be of much help. If there is specific training for the exam, you can discover it from the link above.
- Read from the CSSLP sample questions to gain your idea about the actual exam questions. In this PDF useful sample questions are provided to make your exam preparation easy.
- Practicing on CSSLP practice tests is must. Continuous practice will make you an expert in all syllabus areas.

Reliable Online Practice Test for CSSLP Certification

Make EduSum.com your best friend during your ISC2 Secure Software Lifecycle Professional exam preparation. We provide authentic practice tests for the CSSLP exam. Experts design these online practice tests, so we can offer you an exclusive experience of taking the actual CSSLP exam. We guarantee you 100% success in your first exam attempt if you continue practicing regularly. Don't bother if you don't get 100% marks in initial practice exam attempts. Just utilize the result section to know your strengths and weaknesses and prepare according to that until you get 100% with our practice tests. Our evaluation makes you confident, and you can score high in the CSSLP exam.

Start Online practice of CSSLP Exam by visiting URL
https://www.edusum.com/isc2/csslp-isc2-secure-software-lifecycle-professional