ISC2 CSSLP

ISC2 Secure Software Lifecycle Professional Certification Questions & Answers

Get Instant Access to Vital Exam Acing Materials | Study Guide | Sample Questions | Practice Test

CSSLP

ISC2 Certified Secure Software Lifecycle Professional

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

Table of Contents:

Discover More about the CSSLP Certification.....2

CSSLP ISC2 Secure Software Lifecycle Professional Certification Details: 2

CSSLP Syllabus: 2

Secure Software Concepts - 10% 2
Secure Software Requirements - 14% 3
Secure Software Architecture and Design - 14% 4
Secure Software Implementation - 14% 6
Secure Software Testing - 14% 8
Secure Software Lifecycle Management - 11% 9
Secure Software Deployment, Operations, Maintenance - 12% 11
Secure Software Supply Chain - 11% 13

Broaden Your Knowledge with ISC2 CSSLP Sample Questions: 14

Avail the Study Guide to Pass CSSLP ISC2 Secure Software Lifecycle Professional Exam: 17

Career Benefits: 17

Discover More about the CSSLP Certification

Are you interested in passing the ISC2 CSSLP exam? First discover, who benefits from the CSSLP certification. The CSSLP is suitable for a candidate if he wants to learn about Secure Software Development. Passing the CSSLP exam earns you the ISC2 Certified Secure Software Lifecycle Professional title.

While preparing for the CSSLP exam, many candidates struggle to get the necessary materials. But do not worry; your struggling days are over. The CSSLP PDF contains some of the most valuable preparation tips and the details and instant access to useful **CSSLP study materials just at one click.**

CSSLP ISC2 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)

Topic	Details
	- 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)
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	

Topic	Details
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

Topic	Details
	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, 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	- Credential management (e.g., X.509 and Single Sign-On (SSO))

Topic	Details
Design	- 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 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

Topic	Details
	- 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)) - 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	

Topic	Details
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) - 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

Topic	Details
	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) - 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,	

Topic	Details
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 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))

Topic	Details
	- 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)
Perform Post-	

Topic	Details
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)

Topic	Details
	- Resiliency (e.g., operational redundancy, erasure code, survivability)
Integrate Service Level Objectives (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	

Topic	Details
(IP) ownership, code escrow, liability, warranty, End-User License Agreement (EULA), Service Level Agreements (SLA))	

Broaden Your Knowledge with ISC2 CSSLP Sample Questions:

Question: 1

Using the principle of keeping things simple is related to what?

a) Layered security
b) Simple Security Rule
c) Economy of mechanism
d) Implementing least privilege for access control

Answer: c

Question: 2

Elements of defensive coding include all of the following except what?

a) Custom cryptographic functions to avoid algorithm disclosure
b) Exception handling to avoid program termination
c) Interface coding efforts to avoid API-facing attacks
d) Cryptographic agility to make cryptographic functions stronger

Answer: a

Question: 3

Complete mediation is an approach to security that includes what?

a) Protecting systems and networks by using defense in depth
b) A security design that cannot be bypassed or circumvented
c) Using interlocking rings of trust to ensure protection to data elements
d) Using access control lists to enforce security rules

Answer: b

Question: 4

What is the most important source of error information to employ when checking code?

a) Previous errors in the code base(s)
b) SANS Top 25 list of programming errors
c) OWASP Top 10 list of application errors
d) MITRE CWE database

Answer: a

Question: 5

What is the fundamental approach to security in which an object has only the necessary rights and privileges to perform its task with no additional permissions?

a) Layered security
b) Least privilege
c) Role-based security
d) Clark-Wilson model

Answer: b

Question: 6

What is the essential element for scoring the severity of bugs/vulnerabilities?

a) Use cases
b) Difficulty to fix
c) Cost to remediate
d) Impact

Answer: d

Question: 7

What describes the ability of a subject to interact with an object?

a) Authentication
b) Confidentiality
c) Mutual authentication
d) Access

Answer: d

Question: 8

Which testing methodology can improve maintainability of the code base?

a) Code walk-throughs
b) Static application security testing (SAST)
c) Dynamic application security testing (DAST)
d) Runtime application self-protection (RASP)

Answer: a

Question: 9

Qualification testing is always guided by what?

a) Prior results
b) The customer
c) A plan
d) A beta test

Answer: c

Question: 10

Which of the following is the best method of finding race conditions?

a) Code walk-throughs
b) SAST
c) IAST
d) DAST

Answer: d

Avail the Study Guide to Pass CSSLP ISC2 Secure Software Lifecycle Professional Exam:

- Find out about the CSSLP syllabus topics. Visiting the official site offers an idea about the exam structure and other important study resources. Going through the syllabus topics help to plan the exam in an organized manner.
- Once you are done exploring the **CSSLP syllabus**, it is time to plan for studying and covering the syllabus topics from the core. Chalk out the best plan for yourself to cover each part of the syllabus in a hassle-free manner.
- A study schedule helps you to stay calm throughout your exam preparation. It should contain your materials and thoughts like study hours, number of topics for daily studying mentioned on it. The best bet to clear the exam is to follow your schedule rigorously.
- The candidate should not miss out on the scope to learn from the CSSLP training. Joining the ISC2 provided training for CSSLP exam helps a candidate to strengthen his practical knowledge base from the certification.
- Learning about the probable questions and gaining knowledge regarding the exam structure helps a lot. Go through the **CSSLP sample questions** and boost your knowledge
- Make yourself a pro through online practicing the syllabus topics. CSSLP practice tests would guide you on your strengths and weaknesses regarding the syllabus topics. Through rigorous practicing, you can improve the weaker sections too. Learn well about time management during exam and become confident gradually with practice tests.

Career Benefits:

- Passing the CSSLP exam, helps a candidate to prosper highly in his career. Having the certification on the resume adds to the candidate's benefit and helps to get the best opportunities.

Here Is the Trusted Practice Test for the CSSLP Certification

EduSum.Com is here with all the necessary details regarding the CSSLP exam. We provide authentic practice tests for the CSSLP exam. What do you gain from these practice tests? You get to experience the real exam-like questions made by industry experts and get a scope to improve your performance in the actual exam. Rely on EduSum.Com for rigorous, unlimited two-month attempts on the **CSSLP practice tests**, and gradually build your confidence. Rigorous practice made many aspirants successful and made their journey easy towards grabbing the ISC2 Certified Secure Software Lifecycle Professional.

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