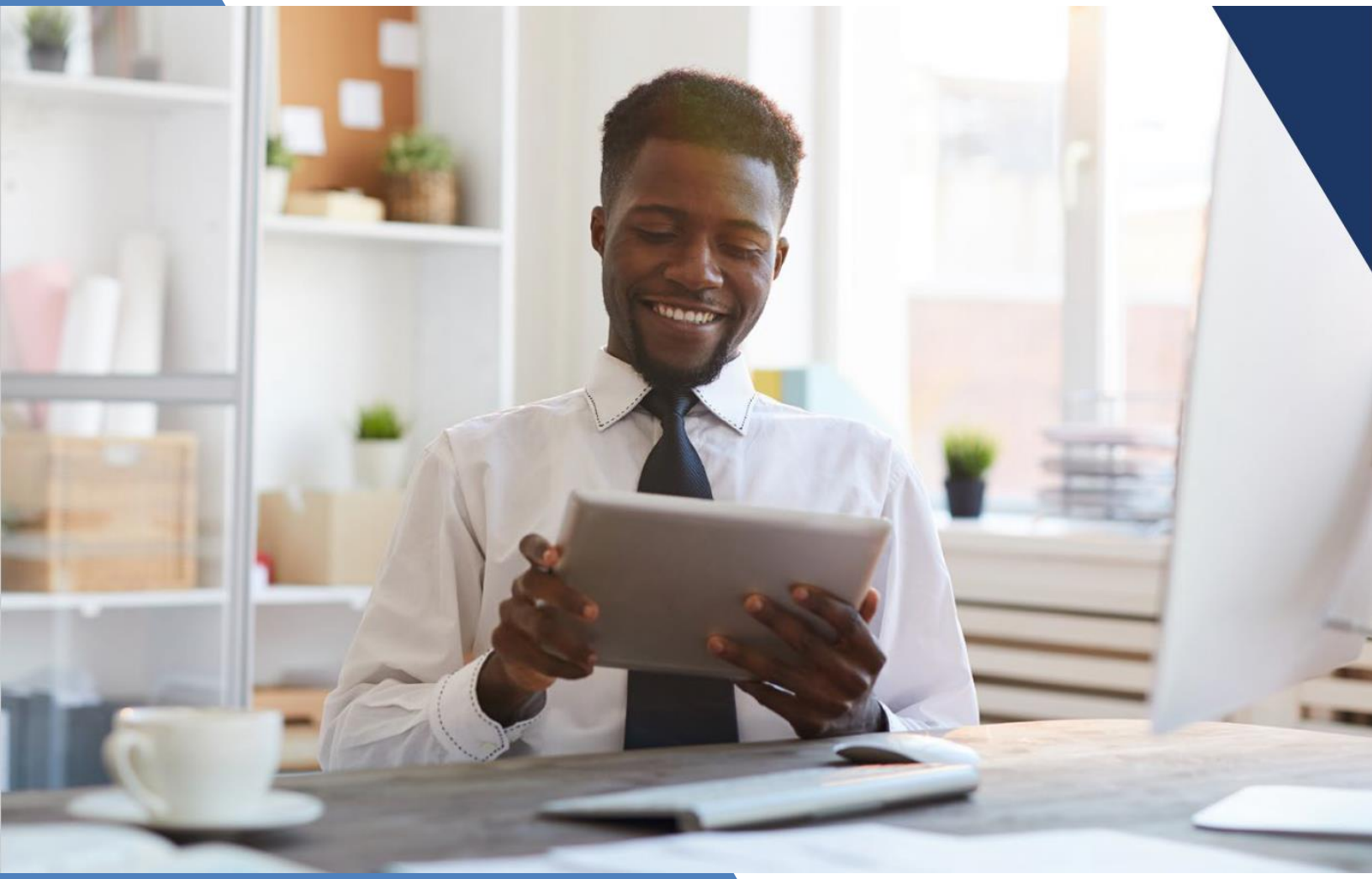
CertNexus®

Cyber Secure Software Developer™ (CSSD) Exam CSD-110

Date Issued: 12/15/2024
Date Modified: 06/01/2026
Version: 1.12



{CSSD}™

Introduction to CertNexus

CertNexus is a vendor-neutral certification body, providing emerging technology certifications and micro-credentials for business, data, developer, IT, and security professionals. CertNexus' mission is to assist closing the emerging tech global skills gap while providing individuals with a path towards rewarding careers in Cybersecurity, Data Science, Data Ethics, Internet of Things, and Artificial Intelligence (AI)/ Machine Learning (ML).

We rely on our Subject Matter Experts (SMEs) to provide their industry expertise and help us develop these credentials by participating in a Job Task Analysis, Exam Item Development, and determining the Cut Score. We also depend upon practitioners in the field to participate in a survey of the Job Task Analysis and beta testing to ensure that our certifications validate knowledge and skills relevant to the industry.

Acknowledgements

CertNexus was honored to have the following Subject Matter Experts contribute to the development of this exam blueprint.

Predrag Tasevski	Mondu	
Jennifer Fischer	Freelancer	
Moaz Mansour	Deloitte	
Ben Ottoman	Avaus / Cyber Security Finland	
Indrani Gorti	Affinity.co	
Jeny Teheran	Fermilab	
Sathish Kumar	Google	
Fadi Sodah	Government	
Mark Szewczul	Toyota North America/AllThingsQuantum	
Rasheen Whidbee	Hackensack Meridian Health	
Andrea Cogliati	LighTopTech Corp.	

Copyright ©2026 by CertNexus Inc. (a division of Logical Operations).
All rights reserved.
Exam objectives are subject to change without notice.

CertNexus® Cyber Secure Software Developer™ (CSSD) Exam CSD-110

Exam Information

The *CertNexus Cyber Secure Software Developer™ (CSSD)* is an industry-validated certification that helps professionals differentiate themselves from other job candidates by demonstrating their ability to build security into software products. Developers are tasked with ensuring that the software they create is protected against a variety of threats—not just at one point in time for one part of the software, but continuously, and for all aspects. Professionals holding the *CertNexus Cyber Secure Software Developer™ (CSSD)* certification have proven that they can follow secure software practices throughout a comprehensive process. This certification validates the skills to define, design, implement, test, deploy, and monitor software securely, so that it brings value to the business while also minimizing the impact of cyber attacks.

Candidate Eligibility

The *CertNexus Cyber Secure Software Developer™ (CSSD)* exam requires no application fee, supporting documentation, or other eligibility verification measures for you to be eligible to take the exam. Simply purchase an access key for the *CertNexus Cyber Secure Software Developer™ (CSSD) Exam CSD-110* course from the CertNexus Store here. This course includes access to the credential process directly through the CHOICE platform.

Exam Prerequisites

While there are no formal prerequisites to register for and schedule an exam, we strongly recommend that you first possess the knowledge, skills, and abilities to do the following:

- Identify the fundamental principles of security, as well as common vulnerabilities, exploits, and attacks that threaten security.
- Write and analyze use cases, technical requirements, specifications, and other software documentation.
- Design software to align with stakeholder requirements for security.
- Develop software using multiple programming languages and coding environments while following generally accepted coding best practices.
- Develop software for a variety of platforms: web, cloud, mobile, desktop, etc.
- Test the security of software using various testing methods: unit testing, regression testing, integration testing, etc.
- Deploy software into secure environments and maintain the security of that software after deployment.
- Work with common software and security tools, such as analysis, debugging, encryption, and penetration-testing tools.
- Identify common defense techniques against cyber attacks.

You can obtain this level of skill and knowledge by taking the following courseware, which is available through training providers located around the world, or by attending an equivalent third-party training program:

- *CertNexus Cyber Secure Software Developer™ (CSSD) Exam CSD-110*

Copyright ©2026 by CertNexus Inc. (a division of Logical Operations).
All rights reserved.
Exam objectives are subject to change without notice.

Exam Specifications

Number of Items: 25

Passing Score: 80% or 20/25 items

Duration: Estimated 30-60 minutes; candidates receive 1 complimentary retake.

Exam Delivery: Online through the CHOICE platform.

Item Formats: Multiple Choice/Multiple Response

Exam Description

Target Candidate:

The *CertNexus Cyber Secure Software Developer™ (CSSD)* exam is designed for any business professional involved in the software development process who desires or is required to develop highly secure software for organizational use. Candidates may be programmers, testers, architects, or any other personnel whose daily job duties involve directly building software; or, they may be project leaders, software engineers, and DevOps personnel who perform additional technical and non-technical duties as part of the overall software development lifecycle. Candidates may have experience in any programming language and any platform.

Exam Objective Statement:

The exam will certify that the successful candidate has the knowledge, skills, and abilities to effectively follow the secure software development lifecycle (SSDLC) by incorporating security into the requirements-definition phase; designing software with security in mind; writing secure code and implementing protection techniques; testing software for security issues and remediating them; deploying software in secure environments; and maintaining the security of software over time. The exam will also certify that the successful candidate can follow the SSDLC in a variety of software scenarios, regardless of language or platform.

Copyright ©2026 by CertNexus Inc. (a division of Logical Operations).
All rights reserved.
Exam objectives are subject to change without notice.

To ensure that candidates possess the aforementioned knowledge, skills, and abilities, the *CertNexus Cyber Secure Software Developer™ (CSSD)* exam will test them on the following domains with the following weightings:

Domain	% of Examination
1.0 Understand the Fundamentals of Secure Software Development	15%
2.0 Explain the Secure Software Development Lifecycle	22%
3.0 Develop Secure Code	33%
4.0 Defending Against Cyberattacks	15%
5.0 Engage in Governance, Risk Management, and Compliance	15%
Total	**100%**

The information that follows is meant to help you prepare for your certification exam. This information does not represent an exhaustive list of all the concepts and skills that you may be tested on during your exam. The exam domains, identified previously and included in the objectives listing, represent the large content areas covered in the exam. The objectives within those domains represent the specific tasks associated with the job role(s) being tested. The information beyond the domains and objectives is meant to provide examples of the types of concepts, tools, skills, and abilities that relate to the corresponding domains and objectives. All of this information represents the industry-expert analysis of the job role(s) related to the certification and does not necessarily correlate one-to-one with the content covered in your training program or on your exam. We strongly recommend that you independently study to familiarize yourself with any concept identified here that was not explicitly covered in your training program or products.

Domains and Objectives

Domain 1.0 Understand the Fundamentals of Secure Software Development [15%]

Objective 1.1 Understand the roles of participants in software development.

1.1.1 Role of project managers in secure development

- Overseeing Project timelines, budgets, and deliverables
- Ensuring security considerations are integrated into project planning
- Managing risk and compliance throughout the development lifecycle (high overview)

1.1.2 Role of software developers in secure development

- Importance of writing secure code and following best practices
- Importance of code review and implementing security fixes (high overview)
- Collaborating with security teams to mitigate vulnerabilities

1.1.3 Role of QA and testing teams in secure development

- Performing security testing (e.g., penetration testing, vulnerability scans, fuzz testing, EUX)
- Identifying and reporting security flaws and software releases
- Ensuring compliance and security standards before deployment

1.1.4 Role of marketing and product management in secure development

- Balancing security requirements with user experience and business goal
- Ensuring security is a competitive advantage in product positioning
- Understanding industry regulations and trends
- Communicating security features effectively to customers and clients

Copyright ©2026 by CertNexus Inc. (a division of Logical Operations).
All rights reserved.
Exam objectives are subject to change without notice.

1.1.5 Role of IT and DevOps in secure development

- Implementing secure CI/CD pipelines
- Managing infrastructure security, including cloud and on-premise environments
- Monitoring security incidents and responding to threats and vulnerabilities

Objective 1.2 Understand interdependence across core concepts.

1.2.1 The CIA Triad (Confidentiality, Integrity, and Availability)

- Confidentiality: Preventing unauthorized access to sensitive data
- Integrity: Ensuring data consistency and preventing unauthorized modifications
- Availability: Guaranteeing uptime, redundancy, and resilience against attacks
- Real-world applications, how CIA principles apply to authentication, encryption, and cloud security

1.2.2 The AAA model (Authentication, Authorization, and Accounting)

- Authentication: Verifying user identities using multi-factor authentication (MFA), biometrics, and federated identity management (FIM)
- Authorization: Implementing role-based access control (RBAC) and attribute-based access control (ABAC)
- Accounting: Monitoring user activities through logging, auditing, and anomaly detection
- How AAA aligns with regulatory requirements (e.g., SOC 2, HIPAA, and GDPR)

1.2.3 Protecting data-at-rest, in-use, and in-motion

- Data-at-rest security: Implementing full-disk encryption, database encryption, and access controls
- Data-in-use security: Protecting active data through memory encryption, tokenization, and secure computing environments
- Data-in-motion security: Ensuring secure transmission of data via TLS, VPNs, and secure API communication
- Challenges and emerging threats, including quantum computing risk and AI

1.2.4 The Principle of Least Privilege (PoLP) and Secure Access Management

- Understanding PoLP: Restricting user permissions to only what is necessary
- Implementing Zero Trust Architecture (ZTA) to enforce continuous verification
- Mitigating insider threats through privileged access management (PAM)
- Balancing security and usability to avoid operational bottlenecks

Objective 1.3 Explain the need for secure software development.

1.3.1 Basic principles of secure software development

- Importance of security-by-design and defense-in-depth
- Secure coding practices and vulnerability mitigation
- The role of security frameworks and best practices (e.g., OWASP, NIST)

1.3.2 Commercial advantage of secure software development

- Competitive edge through strong security practices
- Enhancing customer trust and brand reputation
- Differentiating products in security-sensitive markets (e.g., finance, healthcare)

1.3.3 Financial and legal consequences of security failures

- Impact of regulatory fines (e.g., GDPR, HIPAA, PCI-DSS non-compliance)
- Costs of security breaches, including data loss, lawsuits, and penalties
- Case studies of companies impacted by poor security practices (SolarWinds, CrowdStrike)

1.3.4 Enhancing software reliability and resilience

Copyright ©2026 by CertNexus Inc. (a division of Logical Operations).
All rights reserved.
Exam objectives are subject to change without notice.

- How security contributes to software stability, uptime, and availability (C.I.A)
- Importance of secure architecture and fault tolerance
- Proactive vs. reactive security measures for long-term resilience

1.3.5 Security as a Collaborative Effort in Development

- Integrating security into DevOps (DevSecOps) and agile workflow
- The role of cross-functional teams in securing applications
- Importance of security awareness training across departments

Objective 1.4 Understand common hacking concepts and terminology.

1.4.1 Black hat, gray hat, white hat

1.4.2 Vulnerabilities

- Broken access control
- Buffer overflow
- Exceptions/unintentional disclosure
- Improper certificate validation
- Improper configuration settings/protection
- Improper error handling
- Incomplete secure update mechanism
- Injection flaws
- Insecure data storage/transfer
- Insecure deserialization
- Insufficient jailbreak/root detection
- Insufficient logging and monitoring
- Lack of input validation
- Lack of output encoding
- Non-intended use (of a software platform or piece of code)
- Use of insecure/outdated components
- Weak cryptographic practices
- Weak passwords
- Weak/no encryption of data

1.4.3 Exploits and attacks

- Broken Object Level Authorization (BOLA)
- Brute force
- Credential stuffing
- Cross-Site Request Forgery (CSRF)
- Denial of Service (DoS) and Distributed Denial of Service (DDoS)
- Insecure Direct Object Reference (IDOR)
- Injection attacks
- SQL injection (SQLi)
- Cross-Site Scripting (XSS)
- Command injection (a.k.a. shell injection)
- Code injection
- Server-Side Template Injection (SSTI)
- HTTP header injection and CRLF injection
- LDAP injection
- Man-in-the-Middle (MitM)

Copyright ©2026 by CertNexus Inc. (a division of Logical Operations).
All rights reserved.
Exam objectives are subject to change without notice.

- Privilege escalation
- Remote Code Execution (RCE)
- Ransomware, spyware, malware, and rootkits
- Session hijacking
- Social engineering attacks
- Baiting
- Business email compromise (BEC)
- Phishing, spear phishing, smishing, vishing, and whaling
- Pretexting
- Scareware
- Spoofing
- Zero-day exploits

Domain 2.0 Explain the Secure Software Development Lifecycle [22%]

Objective 2.1 Explain the duality of the "builder/breaker" mindset and skillset in secure coding.

2.1.1 Builder: Understand how to build and engineer software

2.1.2 Breaker: Understand how to prevent, intercept, and recognize cyberattacks

- Lockheed Martin's Cyber Kill Chain
- MITRE ATT&CK framework

Objective 2.2 Explain the software development lifecycle.

2.2.1 Explain widely used SDLC methodologies

- Agile
- Waterfall
- DevOps
- Lean

2.2.2 Explain SDLC phases common across methodologies

- Requirements/Analysis
- Design
- Development
- Testing
- Deployment
- Maintenance

2.2.3 Explain frameworks and best practices for Secure SDLC (SSDLC)

- BSA Framework for Secure Software
- DevSecOps Shift Left Security
- Microsoft SDL (Security Development Lifecycle)
- NIST SSDF (Secure Software Development Framework)
- OWASP ASVS (Application Security Verification Standard)
- OWASP MAS (Mobile Application Security Project)
- OWASP SAMM (Software Assurance Maturity Model)

2.2.4 Explain key SSDLC best practices for AI applications

- Risk assessment and threat modeling specific to AI applications (e.g., data poisoning, adversarial attacks, model manipulation, and privacy concerns)
- DevSecOps integration in the AI development lifecycle
- Open-source component security
- Continuous security testing and monitoring

Copyright ©2026 by CertNexus Inc. (a division of Logical Operations).
All rights reserved.
Exam objectives are subject to change without notice.

- Compliance with relevant data privacy and security regulations

2.2.5 Explain secure coding best practices when GenAI is used to generate code

- Validate input to the GenAI model
- Review AI-generated code for potential vulnerabilities
- Conduct thorough code reviews of AI-generated code
- Ensure attribution/compliance with IP, copyright, and licensing agreements
- Use AI tools from reputable sources with strong security measures in place
- Sanitize outputs of the AI model

2.2.6 Explain secure coding best practices when GenAI is used across the SDLC

- Planning/Analysis phase: Identify potential threats and vulnerabilities early. Define security requirements alongside functional requirements. Ensure secure coding practices are incorporated from the outset.
- Architecture/Design: Review the security architecture to ensure the design is resilient to attacks. Integrate security controls into the design.
- Development: Apply secure coding practices. Conduct regular code reviews. Ensure AI-generated code adheres to security standards.
- Testing: Perform security testing, including static and dynamic analysis and penetration testing, to simulate real-world attacks and identify weaknesses.
- Deployment: The deployment process should include secure configuration and deployment of security patches. Continuous monitoring should also be implemented to detect and respond to security incidents in real time.
- Maintenance: Make regular updates to address new vulnerabilities and security threats. Have an incident response plan in place to quickly address any security breaches.

2.2.7 Using emerging technologies to improve secure coding practices

- AI vulnerability scanning
- Ethical hacking with AI

Objective 2.3 Understand key roles and their deliverables in the Secure SDLC (SSDLC).

2.3.1 Designer/Architect

- Security design and architecture
- Threat modeling
- Whiteboarding and solutions
- Compliance assurance and adherence to requirements

2.3.2 Developer

- Source code and code review (participation)
- Use of standard libraries and APIs
- Documentation
- Test and deployment scripts

2.3.3 Code Reviewer

- Code review
- Quality Assurance assessment and reporting
- Vulnerability assessment and reporting
- Use of static and dynamic code analysis tools
  - Fortify Static Code Analyzer, OWASP SonarQube, SAST, and SCA
  - Web Security Suite (IAST, RASP), Acunetix, Fortify WebInspect, Burp Suite

2.3.4 Tester

Copyright ©2026 by CertNexus Inc. (a division of Logical Operations).
All rights reserved.
Exam objectives are subject to change without notice.

- Test cases
- Quality Assurance assessment and reporting
- Vulnerability assessment and reporting
- Use of penetration testing methods and tools
  - Black box, white box, grey box
  - Metasploit, Nessus, ZAP, Burp Suite

Objective 2.4 Implement industry-standard SSDLC management practices.

2.4.1 Apply security-by-design principles

2.4.2 Identify security milestones and checkpoints

- Control gates
- Break/build criteria

Objective 2.5 Foster a security-first organizational culture.

2.5.1 Assess security awareness and train stakeholders

2.5.2 Communicate security effectively across teams

- Security Champions
- Security buy-in from stakeholders

2.5.3 Align security goals with business objectives (ROI in cybersecurity)

Domain 3.0 Develop Secure Code [33%]

Objective 3.1 Plan for secure software development.

3.1.1 Define the software requirements

- Determine the functional requirements
- Identify the assets
- Evaluate the security threats to the assets

3.1.2 Determine third-party dependencies

- Identify external dependencies
- Evaluate the security threats of the external dependencies

Objective 3.2 Perform analysis for secure software development.

3.2.1 Define the security requirements

- Map security objectives to functional requirements
- Document detailed security requirements guidelines
- Build a security requirements framework

3.2.2 Define the security requirements for third-party dependencies

- Evaluate the risks of external dependencies
- Determine security requirements for third-party suppliers
- Ensure proper security coverage for third-party code

Objective 3.3 Design a secure software.

3.3.1 Determine the software architecture

- Outline the data flow
- Define the use cases

3.3.2 Perform the threat modeling

- Identify abuse cases
- Determine mitigation strategies
- Analyze the effectiveness and the trade-offs of the mitigation strategies

Objective 3.4 Implement secure software.

3.4.1 Implement secure input and output management

Copyright ©2026 by CertNexus Inc. (a division of Logical Operations).
All rights reserved.
Exam objectives are subject to change without notice.

- Input validation techniques
  - Escape input
  - Client-side and server-side validation
  - Regular expressions
  - Whitelisting/Blacklisting
  - Elimination of string concatenation for database queries
- Output security techniques
  - Output encoding
  - Randomizing UUID
  - Sanitization of user-facing error messages
  - Masking and tokenization

3.4.2 Implement cryptography

- Cryptographic algorithm and libraries (AES, RSA, Weak ciphers)
- Key management
- Secure communications (WireGuard, SSH, IPsec)
- Considerations for mobile and IoT apps and web services
- Cryptography at various levels

3.4.3 Implement authentication and access control

- Password management
- Authentication factors
- Account management lifecycle
- Secure session management (JWT, PKI)
- Authorization and access control (RBAC, DAC, MAC, Zero Trust)
- Trust zones
- Function permissions

3.4.4 Implement error handling and logging

- Error message logging
- Security exception logging
- Log management
- Secure logging & auditing (confidentiality, privacy)

3.4.5 Implement application security parameters and configure security settings

- Parameterizing security properties and settings
- Configuration and protection of security settings
  - Baseline security configuration
  - Credentials management
- Secrets management
  - Hard-coded secrets prevention
  - Secrets in code
  - Secrets detection
  - Environment-specific secrets
  - Secrets rotation
  - Secrets storage
  - Secrets auditing

3.4.6 Implement secure database access

- Database connection access control

Copyright ©2026 by CertNexus Inc. (a division of Logical Operations).
All rights reserved.
Exam objectives are subject to change without notice.

    - Parameterized queries
    - Stored procedures
- Encryption of data at rest and in transit
- Encrypt data stored in databases with strong algorithms
- Secure database connections using TLS to protect data in transit

Objective 3.5 Version Control, Testing, Integration, and Deployment.

3.5.1 Enforce security in the SSDLC

- Manage secure source code, build, and deployment
    - Secure coding standards
    - Code quality
    - Source code integrity
    - Trusted dependencies
    - Build management
    - Automation
- Securing environments
    - Isolation between development, test, and production environments
    - Environment hardening
    - Harden server and services
    - Secure containers and VMs used in development and production
- Secure development tools and CI/CD pipelines
    - IDE security
    - Code confidentiality
    - Risks of AI-generated code
    - Securing built-in CI/CD tools
- Security testing
    - IAST (Interactive Application Security Testing)
    - SAST (Static Application Security Testing)
    - DAST (Dynamic Application Security Testing)
    - SCA (Software Composition Analysis)
    - Manual code review/peer review

3.5.2 Implement cloud and DevSecOps security practices

- Cloud frameworks (CIS Benchmarks, NIST CSF, ISO 27017, CSA Guidance)
- Deploying immutable infrastructure
- Site Reliability Engineering (SRE)
- DevSecOps best practices

3.5.3 Ensure security in manufacturing and device environments

- Equipment security
- Physical security
- Firmware integrity
- Hardware root of trust (TPMs or HSMs)
- Secure boot
- Secure element

Objective 3.6 Maintenance.

3.6.1 Monitor security and functional requirements

- Monitor and respond to incidents

Copyright ©2026 by CertNexus Inc. (a division of Logical Operations).
All rights reserved.
Exam objectives are subject to change without notice.

- Fix bugs and vulnerabilities
- Develop and install bug fixes and vulnerability patches

3.6.2 Continually improve the requirements

- Collect feedback
- Determine possible feature enhancements

3.6.3 Decommission legacy assets

- Perform code coverage
- Identify and remove unused assets
- Securely decommission unused assets

Domain 4.0 Defending Against Cyberattacks [15%]

Objective 4.1 Explain the 7 stages of the Cyber Kill Chain model.

4.1.1 Reconnaissance in Cyber Kill Chain model

- Passive and active information gathering
- Identifying security weaknesses in software and infrastructure
- Automation & AI-driven reconnaissance
- Social engineering & pretexting

4.1.2 Weaponization in Cyber Kill Chain model

- Exploit delivery mechanisms
- Customized malware and payloads
- Automation of payload generation
- Malicious intent with legitimate tools

4.1.3 Delivery in Cyber Kill Chain model

- Exploit distribution mechanisms
- Customized payload distributions
- Automated delivery and propagation
- Weaponizing legitimate channels for malicious payload delivery

4.1.4 Exploitation in Cyber Kill Chain model

- Software & system vulnerability exploitation
- Exploiting weak authentication & misconfigurations
- Client-side exploitation & user-interaction attacks
- Living Off the Land (LotL) exploitation

4.1.5 Installation in Cyber Kill Chain model

- Persistent malware & backdoor installation
- Exploiting privilege escalation for stealthy persistence
- Deploying fileless & polymorphic malware

4.1.6 Command & Control (C2) in Cyber Kill Chain model

- Establishing encrypted & covert communication channels
- Leveraging cloud services & social media for C2
- Using malware & botnets for large-scale C2 operations

4.1.7 Actions on objectives in Cyber Kill Chain model

- Theft & data exfiltration
- System disruption & ransomware deployment
- Lateral movement & privilege escalation
- Sabotage & system integrity compromise

Objective 4.2 Explain key steps to implement and their rationale: Reconnaissance.

Copyright ©2026 by CertNexus Inc. (a division of Logical Operations).
All rights reserved.
Exam objectives are subject to change without notice.

4.2.1 Regularly monitor network traffic for unusual activity

- Deploy Network Traffic Analysis (NTA) & Intrusion Detection and Prevention Systems (IDS/IPS)
- Establish baseline traffic patterns & set anomaly alerts
- Deep Packet Inspection (DPI) & Encrypted Traffic Analysis (ETA)

4.2.2 Leverage threat intelligence to identify potential threats and vulnerabilities

- Integrate threat intelligence feeds into security infrastructure
- Conduct threat intelligence-based vulnerability management
- Utilize dark web & OSINT monitoring for early threat detection
- Foster collaboration through information sharing

4.2.3 Perform regular security audits to identify potential entry points

- Comprehensive Vulnerability Assessments & Penetration Testing (VAPT)
- Account privilege escalation review
- Audit security logs and patch schedule
- Check compliance with current regulations and laws

Objective 4.3 Explain key steps to implement and their rationale: Weaponization.

4.3.1 Conduct malware analysis to understand the nature of potential threats

- Static malware analysis
- Dynamic malware analysis
- Malware classification and documentation

4.3.2 Update all security measures to counter new types of malware

- Maintain up-to-date antivirus & endpoint protection
- Maintain patches and system updates
- Automate incident response & malware containment

4.3.3 Use strong access controls to prevent unauthorized access to critical systems

- Enforce Multi-Factor Authentication (MFA) & Privilege Access Management (PAM)
- Role-Based Access Control (RBAC) & Least Privilege Accessing
- Network segmentation and Zero Trust implementation

Objective 4.4 Explain key steps to implement and their rationale: Delivery.

4.4.1 Implement robust email filtering to block phishing attempts

- Implement advanced email filtering and spam detection
- DMARC, DKIM, SPF configuration

4.4.2 Use network segmentation to limit the spread of any potential breach

- Utilization of VLAN segmentation
- Zero Trust implementation

4.4.3 Train users to recognize and report suspicious emails and links

- Phishing simulation and user training
- Simple reporting mechanisms & gamification

Objective 4.5 Explain key steps to implement and their rationale: Exploitation.

4.5.1 Regularly update and patch all software and systems to fix vulnerabilities

- Establish a mature patch management process
- Automation of patch deployment
- Test patches in staging before deployment
- Establish beta group for QA

4.5.2 Deploy intrusion detection systems to identify and respond to attacks

Copyright ©2026 by CertNexus Inc. (a division of Logical Operations).
All rights reserved.
Exam objectives are subject to change without notice.

- Signature-based and Anomaly-based detection
- Integrations with IDS and SIEM
- IDS tuning and threat hunting

4.5.3 Have a well-defined incident response plan to address any breaches quickly

- Establish Incident Response Team (IRT) & roles
- Develop and implement Incident Response Plan (IRP)
- Implement incident detection and monitoring tools
- Implement tabletop and incident response drills
- Incident response and security orchestration (SOAR)

Objective 4.6 Explain key steps to implement and their rationale: Installation.

4.6.1 Use endpoint protection solutions to detect and remove malicious software

- EDR and IDS implementation
- MDM and endpoint security audits

4.6.2 Conduct regular scans of systems to detect and remove any installed malware

- Automation virus and malware scanning

4.6.3 Maintain regular backups and recovery plan to restore systems in case of attack

- Implement & automate backup and recovery strategies
- Implement DR and BCP strategies
- Perform regular tests, audits, and retrospectives

Objective 4.7 Explain key steps to implement and their rationale: Command and Control.

4.7.1 Continuously monitor network traffic for signs of C2 activity

- Network Detection and Response (NDR)
- SIEM implementation with anomaly detection

4.7.2 Use firewalls and IDS/IPS to block unauthorized access

- Defense in depth
- Anomaly detection and SOC

4.7.3 Analyze network traffic to identify and block C2 communications

- NTA with threat intelligence integration
- Automation of blacklist and malicious URLs/IPs
- Implementation of DNS and proxy filtering

Objective 4.8 Explain key steps to implement and their rationale: Actions on Objectives.

4.8.1 Implement strict access controls to sensitive data and systems

- Role-Based Access Control and least privilege
- Zero Trust implementation & MFA
- Privilege Access Management and timeout

4.8.2 Continuously monitor critical systems for any unusual activity

- SIEM event monitoring with automated threat detection and response
- EDR implementation with fail close

4.8.3 Have a rapid incident response plan to mitigate the impact of any successful attack

- Automate incident response playbook
- Establish clear procedures and processes
- Development of a comprehensive communication strategy

Objective 4.9 Continuously enhance understanding of adversarial behaviors.

4.9.1 Study real-world attack cases

- Analyze documented cases of cyberattacks

Copyright ©2026 by CertNexus Inc. (a division of Logical Operations).
All rights reserved.
Exam objectives are subject to change without notice.

- Learn from post-mortem reports and incident response case studies

4.9.2 Engage with threat intelligence feeds

- Stay updated on the latest threats and attack vectors
- Participate in cybersecurity communities and forums
- Connect to automated threat intelligence feeds

4.9.3 Refer to industry knowledge bases (e.g., MITRE ATT&CK framework)

- Stay informed about known adversary TTPs
- Map observed behaviors in security logs/alerts to the ATT&CK framework

4.9.4 Participate in red team/blue team exercises

- Experience both offensive and defensive security in cyber operations
- Simulate attacks to test defenses, identify weaknesses in systems/software

Objective 4.10 Continuously improve defensive coding skills and understanding of resilient systems.

4.10.1 Reference real-world adversarial tactics and techniques to help anticipate potential threats and prepare appropriate defenses

- Utilize MITRE ATT&CK framework
- Conduct threat modeling and attack simulation
- Utilize red team and purple team exercises

4.10.2 Identify which parts of your codebase are susceptible to adversarial techniques to highlight areas that need stronger security measures

- Implement Static Application Security Testing (SAST) for code analysis
- Implement Dynamic Application Security Testing (DAST) for code analysis
- Review open-source libraries
- Enforce code review and coding standards

4.10.3 Incorporate insights from real-world adversarial tactics and techniques in SSDLC

- Monitor current attack trends
- Utilize findings in red team and purple team engagements
- Utilize threat intelligence information
- Continuous secure development training

4.10.4 Ensure security testing activities cover known adversary techniques

- Utilize OWASP Top 10 and MITRE ATT&CK framework

4.10.5 Develop code review checklists that (a) are informed by real-world adversarial tactics and (b) ensure reviewers look for vulnerabilities that could be exploited using specific techniques

- SDLC alignment with MITRE ATT&CK and OWASP
- Utilize automated code review and vulnerability scanning

4.10.6 Configure static analysis tools to look for code patterns that may be vulnerable to real-world adversarial techniques

- Implement automatic static application testing & threat detection

4.10.7 Enforce defensive coding practices that defend against specific threats

- Secure coding principles
- Automate secure code enforcement with CI/CD pipeline integration (e.g., GitHub)

4.10.8 Ensure all dependencies and components are regularly updated to mitigate known vulnerabilities

- Automate code and update to dependency using Software Composition Analysis (SCA)
- Continuous patch and vulnerability scans

4.10.9 Train teams on real-world adversarial tactics and techniques to build awareness, promote understanding, and upskill periodically

Copyright ©2026 by CertNexus Inc. (a division of Logical Operations).
All rights reserved.
Exam objectives are subject to change without notice.

- Continuous learning and development
- Perform third-party audits and lessons learned sessions

4.10.10 Simulate attacks and adversarial techniques to help coders see potential real-world impacts and reinforce the importance of defensive coding

- Red team and purple team engagements
- Automated threat simulations attack
- Internal capture the flags

Domain 5.0 Engage in Governance, Risk Management, and Compliance [15%]

Objective 5.1 Understand key concepts and terminology of GRC.

5.1.1 Governance, Risk Management, and Compliance (GRC)

- Define Governance
- Explain the need for Governance
- Define Risk Management
- Explain the need for Risk Management
- Define Compliance
- Explain the need for Compliance

5.1.2 Software Supply Chain, Software Supply Chain of Integrity, Software Chain of Custody

- Define Software Supply Chain
- Define Software Supply Chain of Integrity
- Define Software Chain of Custody
- Difference between Chain of Custody versus Chain of Integrity

5.1.3 Software Bill of Materials (SBOM)

- Define Software Bill of Materials
- Explain the need for SBOM
- Define SBOM provenance
- Explain the need for SBOM -provenance

5.1.4 Responsible Vulnerability Disclosure Program

- Define VDP
- Explain the need for VDP
- Benefits of VDP

5.1.5 Trusted dependencies

- Define OSS
- Explain the need for reusing OSS
- Know the two main groups of licenses of OSS
- Identify supply chain risk
- Define Software Composition Analysis (SCA) tool
- Explain how SCA tool works

5.1.6 Third-Party Risk Management

- Define TPRM
- Explain why TPRM is important

Objective 5.2 Understanding key concepts and terminology of Common Compliance Frameworks.

5.2.1 Explain NIST Risk Management Framework (RMF) vs. NIST Cybersecurity Framework (CSF) Core

- CSF
  - Identify
  - Protect

Copyright ©2026 by CertNexus Inc. (a division of Logical Operations).
All rights reserved.
Exam objectives are subject to change without notice.

    - Detect
    - Respond
    - Recover
    - Govern
- RMF
    - Prepare
    - Categorize
    - Select
    - Implement
    - Assess
    - Authorize
    - Monitor

5.2.2 Using security frameworks like NIST SSDF (Secure Software Development Framework)

- Define SSDF
- Explain structure of SSDF

5.2.3 Cybersecurity Compliance Certifications (SOC 2, ISO 2700x)

- Define SOC 2 compliance
- SOC 2 Type 1 versus Type 2
- Define ISO 2700x
- SOC 2 versus ISO 2700x

Copyright ©2026 by CertNexus Inc. (a division of Logical Operations).
All rights reserved.
Exam objectives are subject to change without notice.

CertNexus Cyber Secure Software Developer™ (CSSD) Acronyms

Acronym	Expanded Form
AAA	Authentication, authorization, and accounting
CIA	Confidentiality, integrity, and availability
DoS	Denial of Service
GDPR	General Data Protection Regulation
HIPAA	Health Insurance Portability and Accountability Act
IoT	Internet of Things
IPsec	Internet Protocol security
ISO	International Organization for Standardization
LDAP	Lightweight Directory Access Protocol
OS	operating system
OWASP	Open Worldwide Application Security Project
PASTA	Process for Attack Simulation and Threat Analysis
PCI DSS	Payment Card Industry Data Security Standard
SDLC	software development lifecycle
SOX	Sarbanes-Oxley Act
SSH	Secure Shell
SSL/TLS	Secure Sockets Layer/Transport Layer Security
SQL	Structured Query Language
VPN	virtual private network
XML	Extensible Markup Language
XSS	cross-site scripting
XXE	XML External Entity

Copyright ©2026 by CertNexus Inc. (a division of Logical Operations).
All rights reserved.
Exam objectives are subject to change without notice.

CertNexus offers personnel certifications and micro-credentials in a variety of emerging technology skills including Cybersecurity, Cyber Secure Coding, the Internet of Things (IoT), IoT Security, Data Science, Artificial Intelligence, and Data Ethics. For a complete list of our credentials visit https://certnexus.com/certification/.

CERTNEXUS®

1150 University Ave, Suite 20, Rochester, NY 14607
1-800-326-8724 | info@certnexus.com
certnexus.com

Copyright ©2026 by CertNexus Inc. (a division of Logical Operations).
All rights reserved.
Exam objectives are subject to change without notice.