Zetlan Technologies

Certified Secure Software Lifecycle Professional

Online Course

Learn without leaving home!

ZETLAN TECHNOLOGIES
www.zetlantech.com

Certified Secure Software Lifecycle Professional

Course Modules

Secure Software Concepts

1. **Understand core concepts**
   - 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), (SSO), etc,)
   - 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)

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, etc)
   - Component reuse (e.g., common controls, libraries)

Secure Software Lifecycle Management

Manage security within a software development methodology (e.g., Agile, waterfall)
Identify and adopt security standards (e.g., implementing security frameworks, promoting security awareness)

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

Define and develop security documentation

Certified Secure Software Lifecycle Professional

Define security metrics (e.g., criticality level, average remediation time, Key Performance Indicators (KPI), etc.,)

4. **Decommission applications**
   - End of Life (EOL) policies (e.g., credential removal, configuration removal, license cancellation, archiving, etc.,)
   - Data disposition (e.g., retention, destruction, dependencies)

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

5. **Incorporate integrated risk management methods**
   - Regulations, standards and guidelines (e.g., (ISO), Payment Card Industry (PCI), (NIST), (OWASP), etc.,)
   - Legal (e.g., intellectual property, breach notification)
   - Risk management (e.g., risk assessment, risk analysis)
   - Technical risk vs. business risk

6. **Implement secure operation practices**
   - Change management process
   - Incident response plan
   - Verification and validation
   - Assessment and Authorization (A&A) process

Secure Software Requirements

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

8. **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)

9. **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)

10. 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, etc.)
- Data retention (e.g., how long, where, what)
- Cross-border requirements (e.g., data residency, jurisdiction, multi-national data processing boundaries)

11. Define data access provisioning

- User provisioning
- Service accounts
- Reapproval process

12. Develop misuse and abuse

- Mitigating control identification

Develop security requirement traceability matrix

Define third-party vendor security requirements

Secure Software Architecture and Design

13. Define the security architecture

- Secure architecture and design patterns (e.g., Sherwood Applied Business Security Architecture (SABSA), etc.,)
- 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, etc.,)
- 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 Servc (IaaS))
- Mobile applications (e.g., implicit data collection privacy)
- Hardware platform concerns (e.g., side-channel mitigation, speculative execution mitigation, 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)

14. 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)

Certified Secure Software Lifecycle Professional

15. 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 (VM), Python, PowerShell)
- Operating system (OS) controls and services
- Secure backup and restoration planning
- Secure data retention, retrieval, and destruction

16. Perform threat modeling

- Threat modeling methodologies (e.g., Spoofing, Tampering, Repudiation, Information Disclosure, etc.,)
- 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)

Perform architectural risk assessment and design reviews

Model (non-functional) security properties and constraints

Define secure operational architecture (e.g., deployment topology, operational interfaces, Continuous Integration.

Secure Software Implementation

17. 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), (DAC), (MAC))
- Processor microarchitecture security extensions

18. Analyze code for security risks

- Secure code reuse
- Vulnerability databases/lists (e.g., Open Web Application Security Project (OWASP) Top 10, etc.,)
- 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)

Implement security controls (e.g., watchdogs, file integrity monitoring, anti-malware)
Address the identified security risks (e.g., risk strategy)

19. 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)

20. Apply security during the build process

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

Secure Software Testing

21. Develop security testing strategy & plan

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

22. Develop security test cases

- Attack surface validation
- Automated vulnerability testing (e.g., dynamic application security testing (DAST), (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

Verify and validate documentation (e.g., installation & 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)

23. Classify and track security errors

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

24. 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 (e.g., independent/internal verification and validation, acceptance test)
Secure Software Deployment, Operations, Maintenance

25. 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

26. Secure configuration and version control

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

27. 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)

28. Store and manage security data

- Credentials
- Secrets
- Keys/certificates
- Configurations

29. 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

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

30. 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))

31. Execute the incident response plan

- Incident triage
- Forensics
- Remediation
- Root cause analysis

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

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

Incorporate runtime protection (e.g., (RASP), (WAF), (ASLR), dynamic execution prevention)

32. 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)

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

Secure Software Supply Chain

33. Implmnt software supply chain risk management (e.g., International Organization for Standardization (ISO), (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

34. Analyze security of third-party software

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

35. 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

36. 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)

37. Support contractual requirements

- Intellectual property ownership
- Code escrow
- liability
- Warranty,
- End-User License Agreement (EULA)
- Service-level agreements (SLA))

Zetlan Technologies

LEARN REMOTELY!!

The efficiency of online learning in terms of time management, flexibility, and the ability to access resources anytime, anywhere can be compelling.

ZETLAN TECHNOLOGIES

www.zetlantech.com

For contact: +91 8680961847
+91 9600579474