# **Secure Agile and DevOps SDLC Practices**

## **Document Control**

|  |  |
| --- | --- |
| Document Title: | Secure Agile and DevOps SDLC Practices |
| Version: | 1.0 |
| Document ID: | SD-SDLC-2024-[Unique Identifier] |
| Approval Date: |  |
| Owner: | Software Development and Security Group |
| Approver: | Chief Technology Officer & Chief Information Security Officer |

Approved by: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## 

This template is developed by KLC Consulting, Inc. <https://klcconsulting.net>

You are free to use this template under the MIT license.

**1. Introduction**

This document provides a comprehensive guide to implementing Secure Agile and DevOps Software Development Lifecycle (SDLC) practices within [Company Name]. It is designed to integrate security seamlessly into our fast-paced and collaborative development and operations environment, ensuring the production of secure and resilient software.

## **2. Scope**

This policy applies to all individuals involved in the software development, operational processes, and security within [Company Name], promoting a unified approach to secure software delivery.

## **3. Detailed Secure Agile and DevOps SDLC Practices**

### **3.1 Agile Development Practices with Security**

* **Security in User Stories:** Explicitly include security requirements when drafting user stories. For example, a user story might state, "As a user, I want to securely log in to the application so that my personal information is protected against unauthorized access," emphasizing the need for secure authentication mechanisms.
* **Agile Security Planning:** At the beginning of each sprint, conduct a dedicated session to identify and assess potential security tasks. Utilize tools like OWASP Top 10 to guide these discussions and ensure comprehensive coverage of security concerns.
* **Continuous Security Engagement:** Assign a security specialist to each Agile team. Their role is to provide ongoing security insights, conduct mini-threat assessments during story refinement, and ensure security tasks are considered alongside functional tasks.

### **3.2 DevOps Operational Practices with Security**

* **Integrated CI/CD Security:** Within the CI/CD pipeline, incorporate tools like SonarQube for static code analysis and OWASP ZAP for dynamic analysis to automate security testing. Set up the pipeline to halt deployments if critical vulnerabilities are detected, requiring manual review.
* **Secure Infrastructure Management:** Utilize Terraform or CloudFormation for infrastructure provisioning, ensuring all infrastructure code is stored in version control (e.g., Git) and peer-reviewed. Implement pre-commit hooks to check for misconfigurations or non-compliance with security policies.
* **Proactive Security Monitoring:** Deploy tools like Splunk or ELK Stack for logging and Prometheus or Grafana for monitoring. Set up alerts for suspicious activities, such as multiple failed login attempts or unexpected changes in outbound traffic, to ensure real-time threat detection.

### **3.3 Collaborative Security Practices**

* **Cross-functional Security Teams:** Encourage an open-door policy between development, operations, and security teams. Organize monthly cross-functional meetings to discuss ongoing security concerns, share knowledge, and brainstorm on improving security practices.
* **Security Champions Program:** Identify and train volunteers from development and operations teams to act as Security Champions. These individuals should receive additional training in secure coding practices, threat modeling, and security testing to serve as the first line of defense and peer mentors.
* **Security Awareness and Training:** Develop a security training curriculum tailored to different roles. For developers, focus on secure coding and common vulnerabilities; for operations, emphasize secure deployment practices and incident response; for security teams, ensure they are up-to-date with the latest threat intelligence and defensive technologies.

### **3.4 Secure Development and Deployment**

* **Secure Coding Standards:** Adopt standards like the OWASP Secure Coding Practices. Conduct regular workshops and code walkthroughs to familiarize the development team with these practices. Utilize linters and pre-commit hooks to enforce coding standards automatically.
* **Security Code Reviews:** Integrate automated security review tools into the pull request process. In addition, conduct manual security reviews for critical features, involving at least one peer and a Security Champion, focusing on areas like authentication, data validation, and secure data handling.
* **Pre-Deployment Security Assessments:** Before any deployment, run a full security assessment using tools like Nessus or Qualys to scan for vulnerabilities within the application and the underlying infrastructure. Require a sign-off from the security team for any high or critical findings.

### **3.5 Continuous Security Improvement**

* **Feedback Loops for Security:** Implement a system for developers and operators to easily report security concerns or suggestions, such as a dedicated Slack channel or JIRA board. Regularly review these inputs in security meetings and integrate actionable items into the development backlog.
* **Security Retrospectives:** Include a "security moment" in each retrospective to discuss any security issues encountered, lessons learned from recent security incidents, or updates to security practices. Document these discussions and follow up on action items.
* **Regular Security Audits and Penetration Testing:** Schedule annual third-party security audits and bi-annual penetration testing. Share the results with the relevant teams and create user stories to address any findings, prioritizing them based on risk.

## **4. Compliance and Enforcement**

Compliance with these Secure Agile and DevOps SDLC Practices is essential for maintaining the security and integrity of [Company Name]'s software products and services. Non-compliance will be addressed through a structured review process and may lead to corrective actions.

## **5. Governance**

The Software Development and Security Governance Committee oversees the enforcement, review, and update of this document. Changes require formal approval, and feedback for improvement is encouraged from all stakeholders. Direct suggestions and comments to [Contact Information].