

Secure Development Lifecycle Policy

Purpose

Your Organization builds, configures, and operates software that handles customer and company information. This policy sets the security activities that must happen at each stage of the development lifecycle so that defects are prevented early, found before release, and fixed on a known timeline, rather than discovered in production.

Scope

This policy applies to all software that Your Organization develops, customizes, or commissions from a third party, including applications, APIs, infrastructure as code, build and deployment pipelines, scripts that touch production data, and internally developed tooling. It applies to every engineer, contractor, and vendor who contributes code. Off-the-shelf software that Your Organization only configures is governed by the Vendor Risk Management Policy and the Change Management Policy.

Roles and responsibilities

- **Engineering leadership** owns this policy, funds the required tooling, and approves exceptions.
- **Product and engineering managers** make sure security requirements are part of the definition of done for each work item.
- **Developers** write code to the secure coding standard, request reviews, and fix findings assigned to them.
- **Reviewers** do not approve a change they cannot understand, and never approve their own change.
- **The security team** maintains the standards, the tooling, and the severity model, and reports on findings and fix times.

Requirements

1. Planning and design

- Each new system or significant change states its security and privacy requirements: data handled, trust boundaries, authentication, authorization, logging, and availability needs.
- A lightweight threat model is produced for new services, new external interfaces, and changes to authentication, authorization, or data storage. The model is stored with the design record.
- Security-relevant design decisions are recorded and reviewed by someone other than the author.

2. Secure coding

- Developers follow the Your Organization secure coding standard, which covers input validation, output encoding, parameterized queries, secrets handling, error handling that does not leak internals, and safe use of cryptography.

- Secrets, keys, and credentials are never committed to source control. A secret scanner runs on every push, and any exposed secret is treated as compromised and rotated.
- Approved languages, frameworks, and base images are listed by engineering leadership. Additions require review.

3. Code review and source control

- All production code lives in the approved source control system with branch protection on release branches.
- Every change is peer reviewed and approved by at least one qualified person other than the author before merge. Changes to security-sensitive areas (authentication, authorization, cryptography, payment, tenant isolation) require review by a designated security-aware reviewer.
- Direct commits to protected branches and force pushes are disabled.

4. Security testing

- Automated checks run in the pipeline on every change: static analysis, dependency and license scanning, secret scanning, and container or infrastructure-as-code scanning where applicable.
- Dynamic or manual security testing (for example, a penetration test) is performed before launch of a new internet-facing system and at least annually for production systems that handle sensitive data.
- Findings are rated by severity and tracked to closure. Critical and high findings block release unless an exception is approved. The target fix times are set in the Risk Assessment and Vulnerability Management Policy.

5. Dependencies and supply chain

- Third-party libraries are obtained from trusted registries, pinned to specific versions, and kept in a lockfile.
- Known vulnerable dependencies are patched or replaced within the timelines in the vulnerability management policy.
- A software bill of materials is generated for each release of customer-facing software where tooling supports it.
- Build pipelines run with least privilege, and build artifacts are produced only from reviewed source.

6. Environment separation and test data

- Development, test, and production environments are separate, with separate credentials and access lists.
- Production data is not used in non-production environments unless it has been masked or approved by the data owner, and the environment then carries production-level protections.
- Developers do not have standing write access to production.

7. Release and change

- Releases follow the Change Management Policy: approved, tested, reversible, and recorded.
- Release approval confirms that required tests passed and that no unapproved critical or high finding remains open.
- Emergency changes follow the emergency path in the Change Management Policy, including its after-the-fact review.

8. Outsourced development

- Contractors and vendors who write code for Your Organization are bound by this policy by contract, work in Your Organization-controlled repositories where possible, and deliver source and documentation. Delivered code is reviewed and scanned before acceptance.

9. Retirement

- Decommissioned systems have their access revoked, secrets rotated, data disposed of under the Data Retention Policy, and their repositories archived.

Exceptions

An exception must state the requirement, the reason, the compensating controls, the owner, and an expiry date no later than twelve months out. Engineering leadership approves it, and the security team records it. Expired exceptions are re-approved or closed.

Enforcement

Failure to follow this policy may result in blocked releases, revoked repository access, and disciplinary action in line with the Acceptable Use Policy.

Evidence

Evidence of operation includes branch protection settings, pull request approval records, pipeline scan results, threat models, penetration test reports, and the exception register.

Review cadence

This policy is reviewed annually by engineering leadership and the security team, or sooner after a significant incident, a change in architecture, or a change in applicable requirements.

Effective date: [the date this policy takes effect]