Secure Software Development Lifecycle Policy

Beyond

April 2026

Contents

Table 1: Control satisfaction

Standard	Controls Satisfied
TSC	CC8.1

Table 2: Document history

Date	Comment
Apr 9 2026	Initial document

1 Purpose and Scope

a. The purpose of this policy is to define requirements for secure software development practices across Beyond's engineering organization. It establishes controls that integrate security into every phase of the software development lifecycle, from design through deployment and operations.

b. This policy applies to all employees, contractors, and third parties who design, develop, test, deploy, or maintain software systems on behalf of Beyond.

c. This policy applies to all software systems developed or operated by Beyond, including production services, internal tools, and infrastructure-as-code.

2 Background

a. Beyond operates a continuous delivery model where developers deploy changes to production multiple times per day. This high-frequency, small-batch approach reduces risk by minimizing the scope and blast radius of each change, enabling rapid detection and recovery when issues arise.

b. This policy is designed for modern continuous delivery environments, not traditional waterfall or phase-gated models. Security is achieved through automated controls embedded in the development pipeline, peer review, deep observability, and a culture where security is everyone's responsibility.

c. This policy aligns with the OWASP DevSecOps Guideline, which promotes shift-left security: detecting security issues as early as possible in the development process. It is also informed by DORA research, which demonstrates that integrating security into daily development work, rather than treating it as a separate phase, results in both faster delivery and fewer security issues.

3 References

a. OWASP DevSecOps Guideline

b. DORA Capabilities: Pervasive Security, Streamlining Change Approval

c. Minimum Viable Secure Product (MVSP)

d. Change Management Policy (reference (a))

e. Risk Assessment and Mitigation Policy (reference (b))

f. Network and Application Vulnerability Management Procedure (reference (c))

4 Development Practices

a. Beyond follows a trunk-based development model. Developers commit directly to the main branch or merge short-lived feature branches. Long-lived branches are discouraged.

b. All source code is stored in version-controlled repositories with full audit history of changes, including author, timestamp, and commit message.

c. Developers must not commit secrets, credentials, API keys, or other sensitive material to source code repositories. Runtime secrets are managed through HashiCorp Vault, which provides centralized, auditable secrets management with dynamic credentials, lease-based access, and encryption as a service.

d. Production systems must not contain live data in non-production environments. Test data must be synthetic or appropriately anonymized.

e. Developers must follow secure coding practices, including input validation, output encoding, parameterized queries, and proper error handling, consistent with the OWASP Top 10.

5 Security Controls in the CI/CD Pipeline

a. Beyond uses Concourse CI for continuous integration automation. All CI pipelines are defined as code and version-controlled. Security controls are embedded into the pipeline and run on every commit, enforcing quality and security gates before code reaches production. The pipeline implements the following stages, aligned with the OWASP DevSecOps pipeline model:

   i. *Secrets Detection:* Gitleaks scans all commits to detect and block credentials, API keys, and other secrets from entering the codebase. This control is enforced as a blocking CI gate.

   ii. *Static Application Security Testing (SAST):* SonarQube performs static code analysis on every build, identifying vulnerabilities, security hotspots, and code quality issues including OWASP Top 10 and SANS Top 25 coverage. SonarQube findings are reviewed reactively and tracked for remediation per the Vulnerability Management Procedure (reference (c)).

   iii. *Software Composition Analysis (SCA):* Dependabot continuously monitors third-party dependencies for known vulnerabilities and automatically creates pull requests to update affected libraries. Critical and high-severity dependency vulnerabilities must be addressed per the remediation SLAs defined in reference (c).

   iv. *Automated Testing:* Unit tests, integration tests, and end-to-end tests run on every commit. CI must pass (green build) before code can be

merged or deployed. Test failures are blocking.

v. *Application Security Scanning:* GCP Web Security Scanner performs dynamic analysis of running web applications to identify vulnerabilities in deployed services.

vi. *Infrastructure Security Scanning:* GCP Security Command Center continuously monitors cloud configuration and network-level vulnerabilities. GCP Cloud IDS provides real-time intrusion detection.

b. The following controls are enforced as blocking CI gates, meaning code cannot proceed to deployment if they fail:

i. All automated tests (unit, integration, end-to-end)

ii. Gitleaks secrets detection

iii. Dependabot SCA (for critical/high findings)

c. SonarQube findings operate in a reporting and reactive model. Findings are tracked and remediated per the Vulnerability Management Procedure (reference (c)) but do not block deployment.

6 Change Classification

a. Not all changes carry the same risk. Beyond classifies changes to determine the appropriate level of review and the deployment path, as defined in the Change Management Policy (reference (a)):

i. *Standard Changes:* Bug fixes, configuration updates, minor UI changes, dependency patches, and other low-risk, well-understood changes. Standard changes may deploy directly to production after passing all automated CI gates. Peer review is at developer discretion.

ii. *Significant Changes:* New features, architectural changes, data model changes, security-sensitive changes, and changes to authentication or authorization logic. Significant changes must be deployed to a staging environment before production. Peer review via pull request is required.

iii. *Emergency Changes:* Critical production fixes for outages or active security vulnerabilities. Emergency changes follow an expedited path with sign-off from a Staff Engineer or the CTO. A post-deployment review must be conducted within 24 hours.

7 Feature Flags and Progressive Delivery

a. Beyond uses feature flags (PostHog, backed by an in-house Redis cache) to decouple deployment from release. This enables:

i. Progressive rollout of new features to a subset of users before full release.

ii. Instant rollback by disabling a flag, without requiring a new deployment.

iii. Safe experimentation in production with controlled blast radius.

b. Feature flags for significant changes should default to off in production and be explicitly enabled after validation.

c. Stale feature flags must be cleaned up. Feature flags that have been fully rolled out or abandoned should be removed within a reasonable timeframe.

8 Deployment

a. Beyond practices continuous deployment. Merges to the main branch trigger the CI pipeline, and successful builds are automatically deployed to production.

b. All deployments are zero-downtime, using rolling deployments orchestrated by HashiCorp Nomad across multiple job instances. Service discovery and configuration are managed through HashiCorp Consul.

c. All production deployments are logged with relevant details including author, timestamp, commit hash, and deployment status.

d. Deployments to staging environments are triggered by pushing to feature branches, enabling pre-production validation for significant changes.

9 Monitoring, Observability, and Incident Response

a. Beyond maintains deep observability across all production systems to enable rapid detection and response:

i. *Application Performance Monitoring:* Datadog APM provides real-time visibility into application performance, latency, errors, and throughput.

ii. *Error Tracking:* Sentry captures and aggregates application errors with full stack traces and contextual data.

iii. *Log Aggregation:* Mezmo aggregates logs from all production services for search, analysis, and alerting.

iv. *Infrastructure Monitoring:* GCP Cloud Monitoring tracks infrastructure-level metrics and triggers alerts on anomalies.

v. *Intrusion Detection:* GCP Cloud IDS monitors network traffic for malicious activity in real time.

vi. *Secrets Auditing:* HashiCorp Vault provides audit logging of all secrets access, enabling detection of unauthorized or anomalous credential usage.

b. Alerts are configured to notify the engineering team of anomalies, errors, and potential security incidents. Beyond maintains timezone coverage from Eurasia to the US West Coast, ensuring continuous monitoring without a formal on-call rotation.

c. When a production incident occurs, the engineer on duty initiates the response. Security incidents are escalated per the Security Incident Response Policy.

10 Rollback and Recovery

a. All changes must be revertible. Rollback strategies include, individually or in combination:

i. Reverting the commit and redeploying through the standard CI/CD pipeline.

ii. Disabling the change via feature flag.

iii. Rolling back to a previous container image or deployment version at the infrastructure level.

b. The choice of rollback strategy depends on the severity and nature of the issue. The goal is to restore service as quickly as possible.

11 AI-Assisted Development

a. Beyond permits the use of AI-assisted development tools (e.g., code completion, code generation, AI pair programming) to improve developer productivity.

b. AI-generated code is subject to the same security controls as human-written code. All automated CI gates, peer review expectations, and security scanning apply equally regardless of whether code was written by a human or generated by AI.

c. Developers are responsible for reviewing and understanding AI-generated code before committing it. AI tools must not be used to generate or handle secrets, credentials, or sensitive data.

d. AI tools that process source code must be evaluated and approved by the CTO or acting CISO to ensure they meet Beyond's data handling and

confidentiality requirements.

12 Security Training

a. All developers receive formal security training covering secure coding practices, common vulnerabilities (OWASP Top 10), and Beyond's security policies and procedures.

b. Security training is provided during onboarding and refreshed annually.

c. The engineering team conducts periodic security-focused sessions to share knowledge about emerging threats, new tools, and lessons learned from incidents.

13 Exceptions

a. Exceptions to this policy must be approved by a Staff Engineer or the CTO, acting CISO.

b. All exceptions must be documented, including the justification, the scope of the exception, and the compensating controls in place.

c. Exceptions are time-bound and must be reviewed for renewal or closure.

14 Compliance

a. All team members involved in software development and operations are required to adhere to this policy. Non-compliance may result in corrective actions.

b. This policy is reviewed annually and updated as needed to reflect changes in Beyond's technology stack, threat landscape, or regulatory requirements.