Secure Software Development Lifecycle (SSDLC) Policy

Organization: Caliber Technologies Inc. (Calimatic Mail) **Document Version:** 1.0 **Effective Date:** February 14, 2026 **Last Reviewed:** February 14, 2026 **Classification:** Confidential

---

1. Purpose

This document defines the Secure Software Development Lifecycle (SSDLC) policy for Calimatic Mail, a multi-tenant email-as-a-service platform. The SSDLC ensures that security is integrated into every phase of software development, from design through deployment and maintenance.

2. Scope

This policy applies to all software developed and maintained as part of the Calimatic Mail platform, including:

- Backend API services (Node.js / Fastify)
- Frontend web applications (Next.js)
- Infrastructure configurations (Docker, Traefik)
- Third-party integrations (Zoom, Google OAuth)
- Database schemas and migrations

3. SSDLC Phases

3.1 Requirements and Design

- Security requirements are identified during the design phase for all new features.
- Threat modeling is performed for features involving authentication, authorization, data storage, and third-party integrations.
- OWASP Top 10 risks are evaluated for all user-facing functionality.
- Data flow diagrams are created for features handling sensitive data (OAuth tokens, credentials, PII).

3.2 Secure Coding Standards

The following secure coding practices are enforced:

Input Validation:

- All API inputs are validated using Zod schemas with strict type definitions.
- Query parameters, path parameters, and request bodies are validated at the route level.
- SQL injection is prevented through the use of Drizzle ORM with parameterized queries.

Authentication and Authorization:

- JWT-based authentication with HS256 signing and configurable expiration.
- Refresh token rotation for session management.
- Role-based access control (RBAC) with user, admin, and super-admin roles.
- API key authentication with scoped permissions for programmatic access.
- OAuth 2.0 implementation for third-party integrations (Zoom, Google) with PKCE and state verification.

Sensitive Data Protection:

- OAuth access tokens and refresh tokens are stored server-side only and never exposed in HTTP responses.
- An `onSend` hook strips sensitive fields (`access_token` , `refresh_token` , `client_secret`) from all JSON responses on integration endpoints.
- Passwords are hashed using bcrypt with a minimum cost factor.
- Database credentials, API keys, and secrets are managed through environment variables.

Security Headers:

- Content Security Policy (CSP) enforced at both application and reverse proxy levels.

- Strict-Transport-Security (HSTS) with 1-year max-age, includeSubDomains, and preload.
- X-Content-Type-Options: nosniff on all responses.
- X-Frame-Options: DENY to prevent clickjacking.
- Referrer-Policy: strict-origin-when-cross-origin.
- Permissions-Policy restricting camera, microphone, and geolocation.
- Cache-Control: no-store on all integration endpoints to prevent sensitive data caching.

Rate Limiting:

- Redis-backed distributed rate limiting per endpoint category.
- Configurable limits: 200 req/min (read), 50 req/min (write), 10 req/min (sensitive), 20 req/min (auth).
- Rate limit headers (`X-RateLimit-Limit` , `X-RateLimit-Remaining`) included in all responses.
- Brute-force protection with account lockout after repeated failures.

CORS:

- Origin whitelisting in production, restricted to known application domains.
- Credentials-based CORS with explicit allowed methods and headers.

3.3 Code Review

- All code changes undergo peer review before merging.
- Security-sensitive changes (authentication, authorization, token handling, integrations) receive additional scrutiny.
- Merge requests include a description of security implications where applicable.

3.4 Static Application Security Testing (SAST)

- `npm audit` is run against all packages to identify known vulnerabilities in dependencies.
- Dependency vulnerabilities are triaged based on severity (critical, high, moderate, low).
- Non-breaking fixes are applied immediately; breaking changes are scheduled for the next maintenance window.
- Audit results are documented in periodic SAST reports.

3.5 Deployment Security

- Applications are containerized using Docker with multi-stage builds.
- Production images use minimal base images (Node.js Alpine) to reduce attack surface.
- TLS 1.3 is enforced for all HTTPS traffic via Let's Encrypt certificates managed by Traefik.
- Container images are built from locked dependency files (`package-lock.json`) for reproducibility.
- Environment variables are used for all secrets; no secrets are hardcoded or committed to version control.

3.6 Runtime Security

- Fastify Helmet middleware provides defense-in-depth security headers.
- Request logging with structured JSON output for audit trails.
- Health check endpoints for container orchestration.
- Automated token refresh with 5-minute expiration buffer for OAuth integrations.
- Token revocation on integration disconnect with Zoom OAuth revoke endpoint.

3.7 Third-Party Integration Security

For OAuth-based integrations (Zoom, Google):

- **Principle of Least Privilege:** Only minimum required scopes are requested.
- **Token Storage:** Tokens stored in PostgreSQL, never sent to client browsers.
- **Token Lifecycle:** Automatic refresh, revocation on disconnect, and cleanup via deauthorization webhooks.
- **State Verification:** OAuth state parameter includes userId, provider, timestamp, and nonce to prevent CSRF.
- **Data Compliance:** Deauthorization webhook responds to Zoom's `app_deauthorized` event and submits data compliance confirmation to `https://api.zoom.us/oauth/data/compliance` .

4. Vulnerability Management

- Security advisories for dependencies are monitored via `npm audit` .

- Vulnerabilities are categorized by severity and triaged within:
  - **Critical/High:** 72 hours
  - **Moderate:** 14 days
  - **Low:** Next scheduled release
- Remediation actions are documented and tracked.

5. Incident Response

- Security incidents are escalated to the engineering team immediately.
- A post-incident review is conducted for any security breach.
- Affected users are notified as required by applicable regulations.
- See the separate Incident Management and Response Policy for detailed procedures.

6. Compliance

This SSDLC policy aligns with:

- **OWASP SAMM** (Software Assurance Maturity Model)
- **OWASP Top 10** risk mitigation
- **CWE/SANS Top 25** most dangerous software errors
- **Zoom Marketplace** security review requirements

7. Review Cycle

This policy is reviewed and updated at minimum annually, or when significant changes to the development process or technology stack occur.

Approved by: Engineering Team, Caliber Technologies Inc. **Contact:** security@calimatic.app