Secure Development Policy

1. Purpose

This policy sets out how Subcontext designs, builds, tests and releases software so that security is built in from the start. It applies the principle that security problems are cheapest and safest to fix before code is written, and gives engineers clear standards for everything that follows.

2. Scope

This policy applies to everyone who writes, reviews or releases code for Subcontext, and covers:

- The web application, background processing, REST API and MCP server
- The embeddable widgets that run on customers' websites
- The iOS, Android and Mac apps
- Infrastructure code
- AI agent features, including the prompts, tools and guardrails they use

3. Roles and Responsibilities

Role	Responsibility
Engineers	Follow this policy, write tests, flag security-relevant changes and keep their secure coding knowledge current
Reviewers	Check changes for security and data safety as well as correctness
Security Officer	Owns this policy, provides guidance on security questions and reviews high-risk designs
Founding team	Approves significant changes and production releases

4. Design

- Security and privacy are considered when features are designed, including what data is needed and who may see it
- New features that handle personal data, payments, authentication or permissions receive additional review
- Every customer organisation's data is kept separate, and every query is scoped to the organisation making it

4.1 Questions asked at design time

For each new feature, the engineer considers and, where relevant, records:

- What data does it collect or use, and is all of it needed? How is it classified under the Data Classification Policy?
- Who can see or change it? Does it respect the owner, manager and user roles, and organisation boundaries?
- How long is the data kept, and is the feature covered by the Data Retention & Deletion Policy?
- Should actions be recorded in the audit log, and do they need a notification?
- Does it receive input from outside the platform, such as webhooks, messages or uploaded files? If so, how is that input verified and treated?
- Does it introduce a new sub-processor or send data to a new place? If so, the Supplier Risk Management Policy applies, and a data protection impact assessment may be needed.
- Can an AI agent take this action? If so, does it need human approval, and how are the conduct and compliance rules applied?

4.2 Worked example

A feature that lets an agent send a document to a customer's customer would be reviewed for: whether the document could contain another person's data, whether the recipient's contact details are verified, whether sending is recorded in the audit log, whether the action needs approval, and whether content read from the document is treated as untrusted when the agent reads it.

5. Coding Standards

- Code is version-controlled in Git, and changes are made on branches
- Database access uses the framework's parameterised queries, never hand-built SQL from user input
- Output is escaped by default, and a strict Content Security Policy limits what pages can load and run
- Forms are protected against cross-site request forgery
- Secrets are kept in encrypted credentials, never in source code
- Content from outside the platform that AI models read (customer messages, web pages, documents) is marked as untrusted and separated from instructions
- Sensitive values are filtered from logs

In addition, engineers must:

- Check permissions on the server for every request, never relying on what the interface hides
- Verify the signature of every incoming webhook and reject it if verification fails
- Use the platform's existing encryption for sensitive fields such as credentials, secrets and bank details, rather than inventing their own
- Avoid writing personal data or secrets into logs, error reports or AI tool parameters that are stored
- Prefer well-maintained, widely used dependencies, and add new ones only when they are clearly needed

6. Automated Checks

The full check suite runs before code is released:

- Code style and quality checks (RuboCop)

- Known-vulnerability checks on Ruby gems (Bundler Audit) and browser JavaScript packages (importmap audit)
- Static security analysis (Brakeman), failing on any warning
- The full automated test suite, which must cover 100% of lines and branches
- Browser tests of complete customer journeys
- Tests for the embeddable widgets and mobile apps

Dependabot raises weekly updates for dependencies.

Tests for the embeddable widgets also run automatically when changes are pushed. Tests must cover the failure paths of security controls, for example that a forged webhook is rejected or that a user cannot reach another organisation's records, not only the successful path. A failing check is fixed, not skipped. Findings from these checks are handled under the Vulnerability Management Policy.

7. Review

- Changes are reviewed before release, with security-relevant changes flagged for closer review
- Database migrations are reviewed for data safety

Security-relevant changes include anything touching authentication, sessions, permissions, encryption, payments, data export or deletion, webhooks, the API, and AI guardrails or tool approvals. Reviewers check that these changes include tests for misuse as well as normal use.

8. Release

- Every release is built as an immutable, versioned container image, scanned for vulnerabilities when published
- Releases to production require approval in the deployment pipeline
- Database migrations run before the new release goes live, and a failed migration stops the release
- A previous release can be restored if a problem is found

Releases follow the Change Management Policy. Production containers run as a non-root user.

9. Environments

- Development and test environments are separate from production
- Production data is not copied into development or test environments

Development and test environments use seeded, made-up data. Production credentials are not used outside production.

10. Training

Engineers are expected to keep up to date with secure coding practice, including the OWASP Top 10.

Secure development is covered when an engineer joins, and refreshed at least annually alongside the security awareness training described in the Information Security Policy. Lessons from incidents and vulnerability findings are shared with the team.

11. Exceptions

Any departure from this policy, for example a temporary exemption from a check, must be approved by the Security Officer, recorded with its reason and an end date, and removed by that date.

12. Monitoring Compliance

The Security Officer periodically reviews a sample of recent changes against this policy, and reviews the results of the automated checks and any recurring issues. Gaps are tracked and corrected.

13. Related Documents

- Change Management Policy
- Vulnerability Management Policy
- Encryption & Key Management Policy
- Information Security Policy
- Access Control Policy
- Data Classification Policy
- AI Ethics & Responsible Use Policy

14. Document Control

Item	Detail
Owner	Security Officer
Approved by	The directors of Subcontext Ltd
Review	At least annually
Effective	October 2026

Subcontext Ltd, registered in England and Wales, Company No. 17019645. Registered office: 124 City Road, London, EC1V 2NX. The current version of this document is published at subcontext.com.