



PROCEDURE · PRODUCT SECURITY

Secure Development Procedure

Version 1 · Approved by Jacek Janczura · 18 October 2026

Taufolio · Jacek Janczura
Adama Mickiewicza 73, 01-625 Warszawa, Poland
EU VAT: PL5253008276 · support@taufolio.com

Contents

1. Objective

2. Scope

3. Engineering responsibilities

4. Process description

5. Document security classification

6. Non-compliance

7. Responsibilities

8. Schedule

9. Version history

1. Objective

This procedure describes the roles, the process flow and the records involved in Taufolio's software development life cycle. It turns the System Acquisition and Secure Development Lifecycle Policy into concrete steps and gives guidelines for secure development activities.

2. Scope

This procedure applies to the development of Taufolio's products. It must be followed by all staff members, contractors and partners involved in software product development.

3. Engineering responsibilities

Every person who changes Taufolio code or infrastructure configuration must:

- follow the guidelines in this procedure;
- track bugs, weaknesses and tooling issues in the Taufolio issue tracker;
- support external vulnerability assessments and penetration tests when they are carried out.

The Information Security Officer must:

- analyze new technology for security risks and known attack patterns before it is adopted;
- make sure everyone who works on the code has read and understood this procedure;
- keep the development environment separate from production.

4. Process description

4.1 Quality objectives

The process exists to deliver high-quality software on time while keeping it secure. Its effectiveness is judged by:

- **Quality:** the number and severity of defects found in testing and after release, tracked in the issue tracker and visible in the error monitoring data.
- **Timely delivery:** the ability to release planned work when intended, including the ability to ship a fix quickly.

4.2 Stages of the life cycle

Every change goes through the stages requirements, design, development, testing, release and maintenance. A small change may pass through several stages in one sitting, but none is skipped.

4.3 Requirements stage

- For each new feature or change to an existing feature, the high-level requirements must be written down, including any security, privacy and data-protection implications.
- Where needed, high-level requirements must be refined into functional requirements before design starts.
- A feature that handles customer credentials, brokerage data or personal data must be checked against the Encryption Policy, the Data Classification Policy and the privacy policy before work begins.

4.4 Design stage

- The user interface and user flows are specified for each requirement.
- The design must identify how the feature could be misused, which interfaces it exposes and which data it receives, stores and sends to third parties.
- The design must apply least privilege: server-side code only, no direct database access from browsers or mobile clients, and the narrowest provider scopes (for example read-only access to brokerage accounts).

4.5 Development stage

- Work is broken into tasks and each change is developed on a branch in the private GitHub repository. All development is managed through version control so that the latest code is always the basis for new work.
- Developers must follow secure coding practices, keep dependencies current and fix code-level vulnerabilities they find.
- Secrets must never be placed in the repository. A pre-commit secrets scan blocks them, and a blocked commit must be fixed, not bypassed. A secret that has been committed must be treated as compromised and rotated.
- Development uses its own environment and credentials, separate from production. Customer data is never used in development.
- Database schema changes are written as versioned migration files. A migration is applied deliberately by a person, development first and production only after that, and is never applied automatically by a deployment.

4.6 Testing stage

All changes must be tested before release. Testing runs at two levels.

On every change, before it is pushed, an automated gate runs:

1. formatting check;
2. lint;

3. type checking;
4. the unit tests related to the files changed.

On every production release, the following also runs:

1. the full unit test suite;
2. the repository's security guard tests, which check for example that every database table has row level security and that credential tables cannot be reached from the public data API;
3. the end-to-end test suite.

A release must not proceed while any of these checks is failing. A failing check that was not caused by the current change must still be fixed or recorded in the issue tracker with a reference to the commit where it first failed.

4.7 Review and release stage

- Every change must be reviewed before release, either by a person or by an independent automated code reviewer that reads the whole change and returns its findings. The author of a change must not be the only reviewer of that change. Where a human second reviewer is available, they should be used for changes to authentication, encryption, billing and brokerage connections.
- Findings from review must be fixed, or declined with a written reason, before release.
- Production deployments are explicit releases on Vercel. A release is tracked in version control with a commit that identifies it.
- Before a release the previous production deployment is retained. If a release causes a problem, the Information Security Officer or any authorized engineer must roll back to the previous deployment (instant rollback on Vercel) first and investigate second.
- Production environment variables are marked Sensitive in Vercel so they cannot be read back from the dashboard.

4.8 Maintenance stage

- Users can report problems through the in-app "Report a problem" form and by email to support@taufolio.com. Every error and bug must be recorded and tracked to closure.
- Error and performance monitoring supports detection. Alerts are reviewed by the Information Security Officer.
- Fixes follow the same change process, including the automated gate, review and release checks. A hotfix may shorten the waiting time between releases but must not skip the checks.
- Suspected security vulnerabilities are handled under the incident response procedures.

4.9 Change management and vulnerability management

- Every change is tested before it is implemented, and a record of changes, test results and approvals is kept in version control and the issue tracker.

- Updates to dependencies and to platform components used in production must be tested before release to confirm there is no adverse effect on operation, controls or security. An exception because of technical limits must be approved by the Information Security Officer and recorded.
- Where feasible, automated scanners must be used to find code-level and dependency vulnerabilities. Findings must be assessed and fixed in line with the vulnerability management procedure.
- After a platform change, application functions must be reviewed to confirm that they are not compromised.
- Previous versions are retained so that a rollback is always possible.
- Vendor-supplied packages must be modified only when unavoidable, only by the personnel who implement the change, and only after review.

4.10 Test data

Product, design and engineering work must never use customer data for testing. Tests must always be performed using synthetic or dummy data.

5. Document security classification

Public. This document is published in the Taufolio Trust Center. See the Data Classification Policy for the classification levels.

6. Non-compliance

Compliance with this procedure is verified through the automated change gate, code review, provider audit logs, periodic internal reviews and feedback to the policy owner. Any staff member found to have violated it may be subject to disciplinary action, up to and including termination of employment or of their contractual agreement. The action depends on the extent, intent and consequences of the violation.

7. Responsibilities

The Information Security Officer is responsible for approving and reviewing this procedure and the related policy. Staff members are responsible for applying the relevant sections of this procedure in their own work.

8. Schedule

This document is reviewed at least annually and whenever a significant change occurs in Taufolio's systems or organization.

End of Secure Development Procedure. For the version history, see the next page.

9. Version history

Version	Change	Approved by	Date
1 (current)	Taufolio procedure issued for the Taufolio service and published in the Taufolio Trust Center	Jacek Janczura, owner	18 October 2026