MOBLERS DEVELOPER AI

Application Security Baseline

Every app we ship is secure by default. Here's how — and what your security team can audit.

> **Our pledge.** Every moblers-built application includes OS-keychain-backed secret storage, HTTPS-only networking, PII-aware logging, and a CI pipeline that scans every commit for leaked secrets and CVE-affected dependencies. Each safeguard is mapped to its OWASP Mobile Top 10 control so your security team can trace the protection to the standard.

Version: 1.0
Generated: 2026-06-10
Source: portal/docs/APP_SECURITY_BASELINE.md
Audience: Clients, security & compliance teams, auditors.

Application Security Baseline

This document describes the **security guarantees** every Flutter application generated by moblers developer ai ships with, and the **CI controls** that enforce them on every build. Use this as the canonical reference when explaining the offering to clients, auditors, or new engineers.

The baseline is split into two surfaces:

- **Domain A** — controls baked into the Flutter app itself (the code that ships to end-user devices).
- **Domain B** — controls enforced by the GitHub Actions workflow that builds and releases the app.

Every safeguard is cross-referenced to its OWASP Mobile Top 10 (2024) control so audits can trace the protection back to the standard.

Domain A — generated Flutter application

The codegen prompt (`portal/src/lib/flutter-codegen-anthropic.ts`) instructs Claude to emit the following files on **every** project. They are non-negotiable; if a file is missing in a generated zip the codegen sanitizer will surface the gap in the QA checklist.

Concern	File / Surface	OWASP M-id
Secret storage	`lib/core/storage/secure_storage.dart`	M9
HTTPS-only network client	`lib/core/api/api_client.dart`	M5
PII-aware logging	`lib/core/log/app_logger.dart`	M6
Input validation helpers	`lib/core/validators.dart`	M4
Cleartext-traffic disabled (A)	`android/app/src/main/res/xml/network_security_config.xml`	M5
Secret exclusions	`.gitignore`	M1, M8
Disclosure policy	`SECURITY.md`	M2
Stricter static analysis	`analysis_options.yaml` (`very_good_analysis` + errors)	M7
Release build commands	`README_MOBFLERS.md`	M7

Required runtime dependencies

The codegen pubspec.yaml MUST declare:

- `flutter_secure_storage` — OS-keychain-backed token storage

- `dio` — HTTP client with the HTTPS guard + redaction interceptor
- `logger` — backing implementation for `AppLogger`

...and these dev dependencies:

- `very_good_analysis` — stricter lint set than the default `flutter_lints`

Analyzer escalations

`analysis_options.yaml` escalates the following lints from `warning` to `error` so a CI run will fail before insecure code reaches a release:

- `avoid_print` — print statements leak data to release logs
- `unsafe_html` — `dart:html` API misuse
- `avoid_dynamic_calls` — type-erased calls bypass null safety
- `unrelated_type_equality_checks` — subtle bug class often used in auth
- `secure_pubspec_dependencies` — forces `secure` references in pubspec

OWASP Mobile Top 10 mapping

Every generated security file opens with a header comment of the form `// OWASP Mobile Top 10: M<n> — <title>`. This makes the safeguards machine-traceable for audits.

M-id	Title	Where addressed
M1	Improper Credential Usage	`.gitignore`, `secure_storage.dart`
M2	Inadequate Supply Chain Security	`SECURITY.md`, OSV-Scanner step (Domain B)
M3	Insecure Authentication / Authorization	`api_client.dart` (Bearer from secure storage only)
M4	Insufficient Input / Output Validation	`validators.dart`
M5	Insecure Communication	`api_client.dart` (HTTPS guard), `network_security_config.xml`
M6	Inadequate Privacy Controls	`app_logger.dart` (PII redaction in release)
M7	Insufficient Binary Protections	`analysis_options.yaml`, `MOBLERS_OBFUSCATE_RELEASE` flag (Domain B)
M8	Security Misconfiguration	`.gitignore`, top-level workflow `permissions: contents: read` (Domain B)
M9	Insecure Data Storage	`secure_storage.dart` + comment forbidding `SharedPreferences` for tokens
M10	Insufficient Cryptography	`flutter_secure_storage` uses OS keychain; comment notes "no homegrown crypto"

Regulated profile (HIPAA / PCI / GDPR-with-PII / banking)

When the structured spec contains regulated keywords (`HIPAA`, `PCI`, `SOC 2`, `GDPR with PII storage`, `biometric`, `cert pinning`, `jailbreak`), the codegen prompt additionally instructs Claude to scaffold:

- `lib/core/auth/biometric_gate.dart` (using `local_auth`) for step-up authentication on sensitive routes.
- A TODO with the code shape for certificate pinning via Dio's `badCertificateCallback` + a comment listing the SHA-256 fingerprints to populate.
- A `SECURITY.md` "Regulated profile" subsection enumerating which extra controls are present and which still need PM follow-up.

If the spec is silent on these keywords, the regulated files are **not** emitted, but `SECURITY.md` mentions the regulated baseline is available on request.

Domain B — hardened GitHub Actions workflow

The default workflow (`.github/workflows/moblers-build.yml`) is generated by both `portal/netlify/functions/run-codegen-background.js#buildDefaultWorkflow` and `portal/src/lib/flutter-ci.ts#buildDefaultWorkflowYaml` — kept in sync. A standalone example lives at `portal/docs/examples/github-flutter-production.yml`.

Workflow-level controls

- **Top-level least-privilege** — `permissions: contents: read` on the workflow; individual jobs widen explicitly (e.g. `pages: write` on the build job). A compromised step cannot write to the repo.
- **Concurrency** — `concurrency.group: moblers-build-${ref}-${target}` with `cancel-in-progress: true`. Stale runs are cancelled when newer commits land; saves runner time and keeps portal status pinned to the most recent code.

Per-step controls

Control	Step name	Behaviour
Secret scanning	`Secret scan (gitleaks)`	Runs on Linux. `continue-on-error` so a flaky scan can't block the build; findings surface in run log + Security tab.
SCA / CVE scanning	`SCA scan (OSV-Scanner)`	Downloads OSV-Scanner v1.7.4, runs against `pubspec.lock`. Linux-only.
Static analysis	`Flutter analyze`	`--fatal-warnings --no-fatal-infos` — warnings become errors; infos still tolerated.
Release obfuscation	`Flutter build`	When `MOBLERS_OBFUSCATE_RELEASE=true`, adds `--obfuscate --split-debug-info=build/symbols` to android / ios / macos / desktop. Off by default to avoid breaking legacy builds.
Symbol custody	`Upload debug symbols (obfuscation only)`	When obfuscation is on, uploads `build/symbols/` as a private 90-day artifact for crash de-obfuscation.
Artifact retention	All `Upload …` steps	Public artifacts pinned to `retention-days: 14` (vs GH's 90d default) to reduce the leak window.

Why `continue-on-error: true` everywhere?

The portal relies on the workflow's success / failure callbacks to drive the admin UI. If a security scan crashes (e.g. gitleaks rate-limits, OSV's CDN is down) we deliberately let the build continue so the callback still fires — the security finding still appears in the run log and the GitHub Security tab. **The scans are diagnostic, not blocking**, in line with the moblers "PM stays in control" UX. Promoting scans to blocking can be done per-client by removing the `continue-on-error: true` line on the relevant step.

Enabling release obfuscation

Per-client switch. In the client's GitHub repo:

1. `Settings → Secrets and variables → Actions → Variables`.
2. Add a new repository variable `MOBLERS_OBFUSCATE_RELEASE` with value `true`.
3. Trigger a production build from the moblers admin panel.
4. The build will emit native artifacts with `--obfuscate --split-debug-info=build/symbols`, and the private `debug-symbols-*` artifact is uploaded for crash de-obfuscation.

Recommended client-repo follow-ups

The workflow header comments these out as TODOs because they require repo settings that codegen can't safely script:

- **Pin third-party actions to commit SHAs** — `gitleaks/gitleaks-action`, `subosito/flutter-action`, `actions/upload-artifact`, etc. Then add a `.github/dependabot.yml` so Dependabot

keeps them current. Without SHA pinning, a re-tag of an action could push malicious code to client builds.
- **Enable GitHub Secret Scanning** + **Dependabot security updates** in repo Settings. These are GitHub-native services; they catch issues the workflow can't see (e.g. private keys in historical commits).
- **CODEOWNERS** + **Branch Protection** on the prod branch. Forces every change to the production code path through PR review and CI.

How the baseline is communicated to clients

Every formal proposal (`portal/src/lib/proposal-copy.ts`) and signed services agreement (`portal/src/lib/services-agreement-copy.ts`) should reference this baseline document so the client understands what they're buying. A future PR can add a "Security baseline" section to those copy files and surface a "Download security baseline" button next to the proposal download buttons — tracked separately.

Maintenance

When you add a new safeguard:

1. Update both workflow generators (`run-codegen-background.js` + `flutter-ci.ts`) — they MUST stay in lockstep.
2. Update the example workflow (`docs/examples/github-flutter-production.yml`).
3. Add a row to the appropriate table above with the new OWASP M-id.
4. Add the file to the codegen prompt's required list if Domain A.
5. Smoke-test by running a codegen job against a sample project and verifying the workflow YAML diff + generated files.