DOWNLOADABLE ENGINEERING REFERENCE / PDF EDITION

Mobile Release Hardening Checklist

This checklist is designed for teams shipping iOS and Android apps that have real production stakes. It is intentionally practical: it focuses on failure modes, trust boundaries, and release workflows that repeatedly break under pressure.

This is not legal advice and not a compliance framework. Use it to reduce operational and security risk.

1) Identity and Environment Separation

- Confirm production, staging, and dev environments are clearly separated at the API, identity, and telemetry layers.
- Verify mobile apps do not accidentally point at non-production endpoints in production builds.
- Ensure secrets and service credentials are not shared across environments.
- Confirm analytics and logging destinations are environment-specific (and do not leak production data into test sinks).

2) Signing and Artifact Integrity

- Verify signing identities are controlled, audited, and rotated when needed.
- Confirm build pipelines do not allow unsigned or unreviewed artifacts into release channels.
- Ensure build outputs are reproducible enough to investigate incidents (inputs, versions, and build metadata captured).
- Validate that debug symbols and source maps are handled safely and stored with access controls.

3) Release Validation (Pre-Launch)

- Run a pre-release validation pass covering:
  - auth flows and token refresh under failure conditions
  - offline/online transitions and retry behavior
  - critical purchase/admin/high-risk flows
  - deep links and universal/app links
  - push notification registration + token handling
- Confirm error handling does not leak tokens, secrets, or PII into logs or crash reports.
- Ensure “first run” and upgrade paths are tested (fresh install, upgrade from last stable, upgrade across multiple versions).

4) Rollout Discipline and Rollback Plan

- Prefer phased rollouts over “all users at once” releases when risk is non-trivial.
- Define what a rollback means:
  - app store rollback may be slow or impossible for already-updated devices
  - you may need server-side feature flags, kill switches, or compatibility layers

- Validate feature flags:
  - are evaluated server-side where appropriate
  - cannot be trivially bypassed by a modified client

5) Crash and Performance Guardrails

- Confirm crash reporting is enabled for production builds and tested in a non-sensitive way.
- Validate performance budgets:
  - startup time
  - memory behavior on low-end devices
  - network usage and retry storms
- Ensure background work is bounded (battery and network) and observable.

6) Telemetry and Incident Readiness

- Ensure logs are actionable and privacy-safe:
  - avoid logging tokens, raw credentials, or secrets
  - prefer correlation identifiers over raw payload dumps
- Confirm you can answer, during an incident:
  - “who is affected?”
  - “what changed?”
  - “what is the failure mode?”
  - “what is the safest mitigation?”
- Ensure you have a clear on-call/response path, even for small teams.

7) Security Checks That Matter for Releases

- Validate token lifecycle:
  - refresh behavior under clock skew, backgrounding, and network errors
  - token scope is least privilege
  - rotation does not accidentally create replay windows
- Confirm the backend does not treat the client as authoritative for sensitive decisions.
- Review third-party SDK updates before release:
  - permission usage
  - network destinations
  - supply chain provenance

8) Final “Stop Ship” Criteria

Define stop-ship criteria in advance so decisions are consistent:

- auth/login cannot be completed reliably
- critical user workflows fail under common conditions

- tokens/secrets appear in logs or crash reports
- rollout controls cannot mitigate a serious issue
- monitoring is insufficient to diagnose failures safely

Public engineering reference, not a completed assessment, legal advice, audit, certification, or customer deliverable.

Editable Markdown source: demonicbinary.com/downloads/mobile-release-hardening-checklist.md