# Firmware Security Audit Checklist

This checklist helps engineering managers, firmware leads and product security owners review the security of embedded firmware already shipping or close to release. It covers boot and update integrity, key management, debug access, communication, code quality and vulnerability handling, so the team can see which gaps carry the highest risk and fix those first.

Technically reviewed by the EmbedCrest engineering team  Updated November 16, 2026

## How to use this checklist

Answer each item with evidence from the code, build system or a test on real hardware, not from memory. Mark items as met, partly met, not met or not applicable. Rank every gap by impact and ease of exploitation, then plan fixes for deployed units and new production separately.

## Boot integrity

- [ ] Does the bootloader verify a cryptographic signature on the application before every boot?
- [ ] Is the root of trust stored in immutable memory or hardware protected storage?
- [ ] Does a failed verification stop boot or fall back to a known good image?
- [ ] Is the bootloader itself protected against modification from the application?
- [ ] Are anti rollback version counters used to block installation of older vulnerable images?

## Firmware updates

- [ ] Are update images signed in a controlled pipeline, never on a developer laptop?
- [ ] Is the running image kept intact until the new image is verified, using A/B slots or equivalent?
- [ ] Does a new image need to confirm itself after boot, with automatic rollback otherwise?
- [ ] Is hardware compatibility checked on the device before an image is installed?
- [ ] Has power loss at every stage of an update been tested on real hardware?
- [ ] Is staged rollout with halt criteria used for fleet updates?

## Keys and credentials

- [ ] Are signing keys held in a hardware security module (HSM) or managed key service with logged access?
- [ ] Does each device have unique credentials rather than a shared fleet secret?
- [ ] Are device private keys stored in a secure element or protected flash, not in plain firmware?
- [ ] Is there a documented process to rotate and revoke keys and certificates?
- [ ] Are default passwords absent or forced to change on first use?
- [ ] Are secrets excluded from source control, logs and debug output?

## Debug and physical access

- [ ] Are JTAG (Joint Test Action Group) and SWD (Serial Wire Debug) debug ports locked or disabled on production units?
- [ ] Is flash readout protection enabled in production programming?
- [ ] Are UART (universal asynchronous receiver transmitter) consoles disabled or protected by authentication in production builds?
- [ ] Are test and manufacturing commands removed or locked after production test?
- [ ] Is it documented what an attacker gains with physical access to one device?

## Communication and code quality

- [ ] Is network traffic protected with Transport Layer Security (TLS) or Datagram TLS (DTLS) using current cipher suites?
- [ ] Does the device validate server certificates rather than accept any certificate?
- [ ] Is every external input, including radio and serial frames, length checked and validated?
- [ ] Is static analysis run on every build with high severity findings tracked to closure?
- [ ] Are compiler warnings enabled and treated as errors for security relevant modules?
- [ ] Are stack protection and memory protection unit features used where the hardware supports them?

## Vulnerability management

- [ ] Is a software bill of materials maintained for every released firmware version?
- [ ] Are third party libraries and real time operating system (RTOS) versions monitored for published vulnerabilities?
- [ ] Is there a public way for researchers to report vulnerabilities?
- [ ] Is a target time defined from vulnerability report to patched release?
- [ ] Is the security support period for the product stated to customers?