Securing Embedded Systems for AI-Driven IoT and Automotive Platforms: From Silicon to Cloud

Nikesh G. Gondchawar

EFY Expo Pune 2026 | Main Conference Hall

Why Embedded System Security is Critical Today

Three fundamental shifts have redefined how embedded system security must be designed.

Always-Connected Devices

- Permanent cloud connectivity and OTA updates are now **architectural requirements**
- Remote attack surface exposure demands continuous threat response

Software-Defined Functionality

- AI models and features deployed post-manufacturing, not at design time
- Intellectual property now embedded in firmware, requiring protection

Extended System Lifetimes

- Automotive and industrial systems operate 10-15+ years without hardware refresh
- Security must evolve across the **entire operational lifespan**

The Evolution: Four Generations of Embedded Architecture

Standalone MCUs

Isolated, firmware-only, No connectivity, Security through obscurity & physical isolation.

Networked Embedded Systems

Network-enabled sensors and actuators, limited remote access, Perimeter-style security.

Connected & OTA-Enabled Platform

Cloud backends, OTA updates, Security spans device, network, & update pipeline

AI-Driven, Software-Defined Platforms

On-device inference, Distributed computing, continuous updates, Security is lifecycle-wide & cross-layer

Each generation inherited vulnerabilities from the previous whilst introducing new attack surfaces. Modern platforms combine all four paradigms simultaneously.

What Fundamentally Changed

AI Models as Assets

Neural networks are now deployable intellectual property requiring protection from theft, reverse engineering, and tampering throughout their lifecycle.

OTA as Default

Over-the-air updates shifted from optional capability to architectural necessity, introducing both remedy and risk.

Open Source Dependency

Modern embedded platforms rely heavily on open-source components—powerful but requiring active supply chain security management.

Extended Lifecycles

Automotive and industrial systems operate for 10–15 years, demanding security architectures that evolve without hardware replacement.

The Expanded Threat Landscape

01

Silicon Layer

Hardware trojans, side-channel attacks, physical tampering, manufacturing compromises

02

Firmware Layer

Boot process attacks, firmware manipulation, debug interface exploitation

03

OS & Middleware

Kernel vulnerabilities, privilege escalation, supply chain compromise

04

AI Runtime

Model theft, adversarial inputs, inference manipulation, model poisoning

05

Connectivity & Cloud

Network interception, cloud API abuse, credential compromise, data exfiltration

Each layer presents distinct attack vectors. Adversaries exploit the weakest link—and increasingly, the boundaries between layers.

Why Siloed Security Fails

Security failures emerge at boundaries, not within components

The Critical Misconception

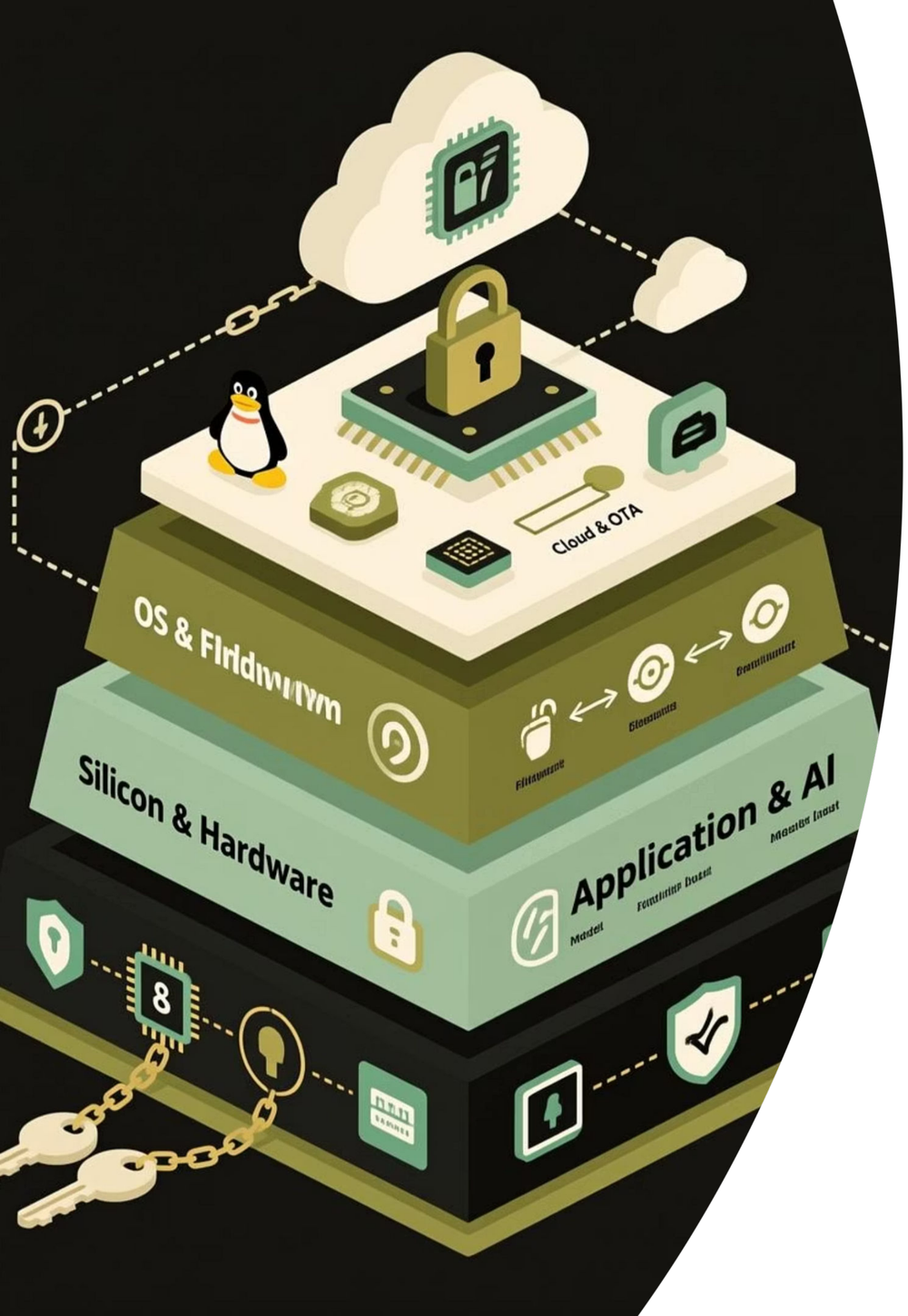
> *Common assumptions we hear: "We have a secure boot" • "We use encrypted storage" • "We run in a TEE"*

Organisations often assume that deploying secure components — a trusted execution environment here, encrypted storage there — automatically yields a secure system.

This represents a fundamental misunderstanding of system security.

Secure components ≠ secure systems

Security gaps emerge at layer boundaries: between hardware and firmware, firmware and OS, application and cloud. Adversaries don't attack your strongest component — they exploit the transitions and trust assumptions between them.



Core Security Framework: Silicon to Cloud

Replacing Siloed Controls with Continuous Trust

1

Establish Root of Trust

Hardware-based cryptographic root anchored in silicon. Immutable, unforgeable, and verified at manufacturing.

2

Propagate Trust Across Layers

Each layer cryptographically validates the next before transferring control. Secure boot, verified execution, attested runtime.

3

Maintain Continuous Trust

Trust doesn't restart—it extends. From power-on through runtime to updates and cloud communication.

This framework must be embedded in system architecture from initial design, not retrofitted during security audits.

Silicon & Firmware: Where Trust Begins & Often Breaks

What Fails in Real Systems

- Shared device keys across production batches
- Incomplete secure boot chains (ROM validates bootloader, but bootloader doesn't validate kernel)
- Exposed debug ports left enabled post-manufacturing
- Weak or missing firmware integrity verification

Why These Failures Matter

- Device cloning and fleet-scale impersonation
- Firmware extraction and reverse engineering
- Unauthorized modifications deployed across millions of units
- No detection of tampering until field compromise

What Correct Architecture Requires

- Hardware root of trust: cryptographic keys burned into silicon at manufacturing
- Per-device identity: unique, immutable credentials for each unit
- Secure & measured boot: each stage cryptographically validates the next
- Debug lifecycle control: disabled post-manufacturing, protected during development
- Authenticated firmware updates: signed, verified before execution

Why STQC Exists (India)

STQC mandates these architectural foundations for connected cameras and IoT devices—not as compliance overhead, but to enforce silicon-to-firmware trust at scale across the industry.

OS & Middleware: Where Trust Erodes

Model (What We Assume)

- Embedded Linux / RTOS provides isolation and security primitives
- Open source accelerates innovation and reuse
- Security updates naturally flow from upstream

Reality (What Happens in Practice)

- OSS components frozen at product release
- No ownership of CVEs post-deployment
- Forked kernels and BSPs without long-term support
- Devices deployed for 10–15 years with limited upstream updates

Failure (Why This Breaks)

- Known vulnerabilities persist in the field
- OS layer becomes the primary attack surface
- Patch availability does not equal patch deployment

Architectural Correction

- Explicit vulnerability lifecycle ownership
- Mandatory isolation and least privilege
- SBOM-driven dependency visibility
- Secure, cryptographically controlled update propagation

Why Standards Exist

Standards codify long-term OSS governance, not just secure configuration

- IEC 62443
- ISO 21434
- ETSI EN 303 645

> Open source is not insecure — unmanaged open source is.

AI Runtime: Where Trust Becomes Probabilistic

Model (What We Assume)

- Trained models behave predictably
- Secure boot + TEE = secure AI

Reality (What Happens in Practice)

- Inference is statistical, not deterministic
- Weights live in memory and evolve independently of firmware

Failure (Why Trust Degrades)

- Model theft via memory scraping or side channels
- Adversarial inputs causing silent misclassification
- No cryptographic binding between model, runtime, and device identity

Result: The system boots securely but reasons insecurely.

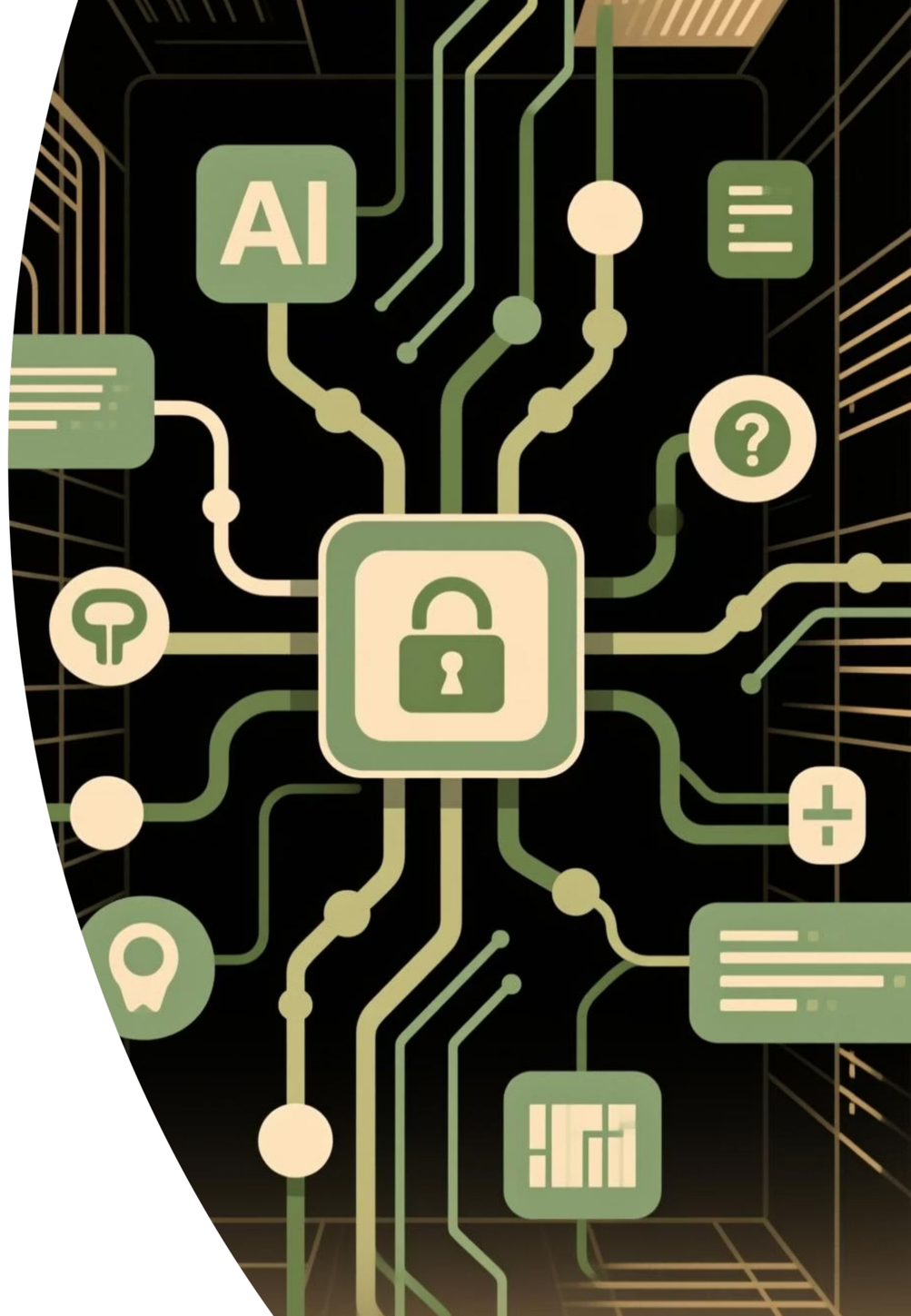
Architectural Correction (How to Contain Risk)

- Signed, versioned models bound to hardware identity
- Isolated AI runtime with input/output anomaly detection
- Secure OTA with rollback and runtime attestation

Why Standards Are Emerging

- ISO/IEC 24029 (AI robustness)
- NIST AI RMF
- ETSI AI security guidance (early-stage)

AI security is moving from model accuracy to model assurance.



Cloud & OTA: Where Trust Must Be Continuously Enforced

Security at scale is enforced through control loops, not point solutions.

Model (Assumption)

- Cloud platforms are secure by default
- OTA fixes issues post-deployment
- Device authentication is sufficient

Reality

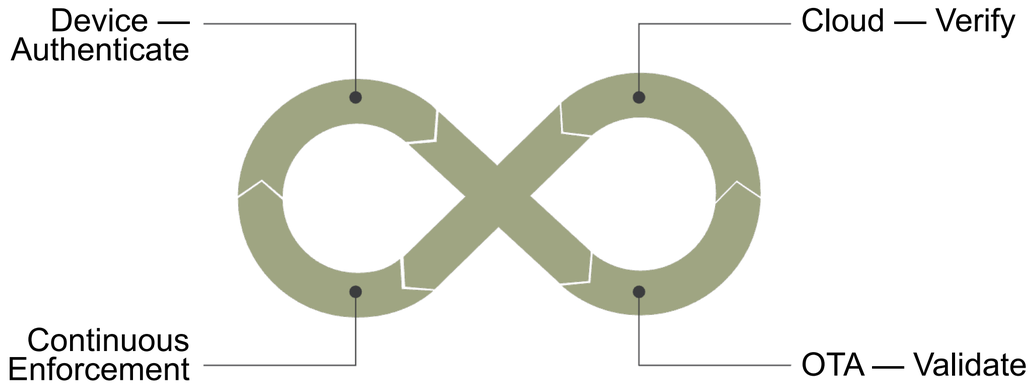
- Cloud becomes a permanent attack surface
- Highly automated OTA pipelines amplify failure impact
- Credential compromise scales across fleets

How Trust Breaks

- Fleet-wide compromise via OTA abuse
- Rollback / downgrade attacks
- Devices running unknown or vulnerable versions

At cloud scale, failures are systemic — not local.

Continuous Security Control Loop



How Trust Is Enforced

- Hardware-anchored device identity
- Mutual authentication (Device <> Cloud <> OTA)
- Signed, versioned, policy-controlled updates
- Continuous fleet attestation

Why Regulation Exists

- STQC (India): Secure OTA, identity, lifecycle
- ETSI EN 303 645: Baseline IoT security
- ISO 27001 / 27017: Cloud governance

Bottom Line:

In connected systems, trust is not a state — it is continuously enforced or silently lost

AI-Driven IoT & Automotive Platforms: Where the Same Trust Chain Is Stressed Differently

Silicon-to-Cloud security principles remain constant — failure modes do not.

AI-Driven IoT Platforms

- Massive device scale (millions to billions)
- Per-device identity and fleet-wide attestation
- Frequent AI model and firmware updates
- Cloud-centric inference and analytics
- Emerging regulation (STQC, ETSI, IoT security baselines)

Failures propagate horizontally across fleets.

AI-Driven Automotive Platforms

- Multiple ECUs and zonal architectures
- Safety-critical AI decisions (ADAS, autonomy)
- Long lifecycles (10–15+ years)
- Limited OTA windows and rollback constraints
- Strong regulatory enforcement (ISO 21434, UNECE R155/R156)

Failures propagate vertically across system layers.

The trust chain is identical — but IoT breaks at scale, while Automotive breaks at safety boundaries.

Security Standards Mapping: One Trust Chain, Multiple Enforcements

Architecture creates trust. Standards enforce it at scale.

Trust Layer	AI-Driven IoT	Automotive (AI-enabled)	AI / ML Systems
Silicon Identity	**STQC** (India), ETSI EN 303 645	**ISO 21434**, UNECE R155	—
Secure Boot & Firmware Integrity	**STQC**, ETSI EN 303 645	**ISO 21434**, UNECE R155	—
OS, Middleware & Open Source Control	ETSI EN 303 645, SBOM practices	**ISO 21434**	—
Device / ECU Isolation	**STQC**	**ISO 21434**, UNECE R155	—
AI Model Integrity	Emerging (**STQC** roadmap)	**ISO 21434** (ADAS scope)	ISO/IEC 24029, **NIST AI RMF**
Runtime Monitoring	ETSI baseline (partial)	**ISO 21434**	**NIST AI RMF**
OTA & Update Control	**STQC** (mandatory), ETSI	UNECE R156	Model versioning controls
Cloud & Fleet Governance	ISO 27001 / 27017	ISO 27001 + OEM policies	ISO 27001
Lifecycle & Compliance	**STQC**	UNECE R155/156	AI governance frameworks

Note: AI/ML standards primarily address model behavior and governance; hardware and OS trust is inherited from the underlying platform.

Standards do not create security — they codify the minimum architecture required to sustain trust at scale.

Architecture Creates Trust. Regulation Enforces It.

From Silicon to Cloud, accountability now spans design, deployment, and operation.

The Reality

AI-driven IoT and Automotive systems fail not because security is unknown — but because trust assumptions collapse at scale, over time, and under AI uncertainty.

Why Enforcement Exists

- India – STQC: Prevents insecure connected cameras and IoT from entering national infrastructure
- Automotive – UNECE R155 / R156: Cyber risk is now a safety and compliance issue
- AI – ISO / NIST: Probabilistic systems require continuous assurance, not static validation

(Regulation follows repeated architectural failure.)

The Architect's Responsibility

Security is no longer a feature, checklist, or certification.

It is a continuously enforced system property — designed in, verified at runtime, and governed across the full lifecycle.

Security Is Not a Feature. It Is a System Property.

- Secure components do not create secure systems.
- If trust is not designed into the architecture, it will fail at scale.
- When architecture fails, regulation enforces what was ignored.

From Silicon to Cloud — trust must be continuously engineered.