Threat Model Document

Dhruv Sunil Bhatia
24CSB0A19
CSE - A
Q119 - Secure OTA Update Compiler

1 Introduction

1.1 Purpose

This document defines the threat model for the Secure OTA Update Compiler. It identifies potential attackers, their capabilities and goals, key assets, trust assumptions, and security threats related to firmware over-the-air (OTA) update mechanisms. The threat model guides the definition and enforcement of compile-time security invariants.

1.2 Scope

The threat model focuses exclusively on firmware update logic implemented in software and analyzed at compile time. It does not cover hardware-level attacks, physical tampering, or post-deployment operational security beyond firmware installation.

2 System Overview

The system under analysis is a compiler-based static analysis and enforcement mechanism that inspects firmware update code before binary generation.

Analyzed Components:

- Firmware update functions (verify, download, install)
- Control-flow paths leading to firmware installation
- Cryptographic API usage within update logic
- Logging and debug statements related to updates

Out of Scope:

- Secure boot enforcement
- Key provisioning and management
- Network-layer protections
- Runtime monitoring

3 Assets

The following assets must be protected:

- Firmware integrity
- Firmware authenticity
- Version correctness
- Confidential update data
- System availability

4 Threat Actors

4.1 Remote Network Attacker

Capabilities: Intercept, modify, or replay firmware update payloads
Limitations: No physical device access

4.2 Malicious Firmware Provider

Capabilities: Craft malicious firmware images
Limitations: Cannot break strong cryptography

4.3 Compromised Update Server

Capabilities: Serve attacker-controlled firmware
Limitations: Cannot bypass compiler enforcement

4.4 Insider / Developer Error

Capabilities: Introduce insecure or incomplete update logic unintentionally
Limitations: No malicious intent

5 Attacker Goals

- Install malicious firmware
- Downgrade firmware to vulnerable versions
- Bypass signature or integrity checks
- Inject unauthorized update sources
- Extract sensitive data via logs

- Cause denial of service

6 Threat Enumeration

- **T1: Malicious Firmware Installation** – Installation without cryptographic verification
- **T2: Rollback Attack** – Forced installation of older vulnerable firmware
- **T3: Untrusted Update Source** – Acceptance of unauthenticated firmware
- **T4: Information Leakage via Logs** – Exposure of sensitive update data
- **T5: Weak Cryptographic Usage** – Use of deprecated or insecure crypto primitives

7 Threat Mitigations (Compiler-Enforced)

Threat ID	**Mitigation Strategy**
T1	Enforce signature verification dominance before installation
T2	Enforce firmware version monotonicity checks
T3	Enforce trusted source validation APIs
T4	Reject sensitive logging statements in update paths
T5	Enforce cryptographic policy at compile time

All mitigations are applied statically, and violations result in compilation failure.

8 Trust Assumptions

- The compiler is trusted and uncompromised
- Cryptographic primitives are secure when used correctly
- Secure boot mechanisms (if present) operate correctly
- Identified update APIs correctly represent firmware installation

9 Security Goals

The Secure OTA Update Compiler guarantees that:

- No firmware installation code exists without mandatory security checks
- All update-related execution paths satisfy defined invariants

- Insecure update logic is eliminated before deployment

10 Residual Risks

- Logic errors outside recognized update code
- Hardware-level or physical attacks
- Runtime vulnerabilities unrelated to update logic

These risks are explicitly outside the compiler's enforcement scope.

11 Conclusion

This threat model demonstrates that critical OTA update threats often arise from missing or incorrectly ordered software logic rather than cryptographic failure. By enforcing update security invariants at compile time, the Secure OTA Update Compiler mitigates high-impact attack classes before firmware deployment.