Update Logic Specification

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

1 Overview

Firmware update logic refers to all code segments responsible for receiving, validating, and installing firmware images on an embedded or IoT device.

2 Scope of Update Logic

This includes functions that perform:

- Cryptographic signature verification
- Firmware version checks and rollback prevention
- Trusted source validation
- Firmware installation procedures

3 Update Logic Identification

Update logic is identified based on the presence of:

- Firmware installation calls (e.g., `install_firmware`, `apply_update`)
- Cryptographic verification functions (e.g., `verify_signature`)
- Version comparison or rollback prevention logic
- Network or storage interfaces used to retrieve firmware images

The compiler assumes that any function directly or indirectly invoking firmware installation APIs is part of the update logic.

4 Security Invariant Enforcement

All control-flow paths leading to firmware installation must satisfy mandatory security invariants enforced by the compiler.

5 Out of Scope

The compiler does not enforce:

- Runtime security policies such as secure boot
- Hardware trust anchors
- Cryptographic key provisioning

The compiler's responsibility is limited to static analysis and compile-time enforcement of firmware update security correctness.

Secure vs Insecure Firmware Snippets

Dhruv Sunil Bhatia
24CSB0A19
CSE - A

Q119 - Secure OTA Update Compiler

1 Insecure Firmware Update Example

The following example demonstrates insecure firmware update logic where mandatory security checks are missing. Firmware is installed without signature verification or version validation.

```
void update_firmware(Firmware *fw) {
    download_firmware(fw);
    install_firmware(fw); // No verification or version check
}
```

2 Secure Firmware Update Example

This example illustrates secure firmware update logic where all required security checks are enforced before firmware installation.

```
void update_firmware(Firmware *fw) {
    download_firmware(fw);

    if (!verify_signature(fw)) {
        return;
    }

    if (fw->version <= current_version()) {
        return; // Prevent rollback
    }

    if (!is_trusted_source(fw)) {
        return;
    }

    install_firmware(fw);
}
```

Defined Enforcement Rules (Informal)

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

1 Overview

This section informally defines the security enforcement rules applied by the Secure OTA Update Compiler. These rules specify mandatory conditions that must hold on all control-flow paths leading to firmware installation. Violation of any rule results in compile-time failure.

2 Enforcement Rules

Rule 1: Signature Verification Before Installation

Every execution path that reaches a firmware installation function must be preceded by successful cryptographic signature verification.

Rule 2: Rollback Prevention via Version Monotonicity

Firmware installation is permitted only if the incoming firmware version is strictly greater than the currently installed firmware version.

Rule 3: Trusted Update Source Validation

Firmware updates must originate from a trusted and authenticated source. Any update retrieved from an unverified or unauthenticated source is rejected.

Rule 4: No Sensitive Information Leakage

Firmware update logic must not log or expose sensitive data such as cryptographic keys, firmware contents, or verification results during the update process.

Rule 5: Approved Cryptographic API Usage

Only approved cryptographic primitives and APIs may be used for signature verification. Weak or deprecated algorithms are explicitly disallowed.