Rule-Checking Logic Description

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

---

1 Overview

The Secure OTA Compiler extension operates on LLVM Intermediate Representation (IR) generated from C source code. It analyzes the `updateFirmware()` function using Control Flow Graph (CFG) construction and Dominator Tree analysis to enforce security invariants.

2 Verification Dominance Check

- Identify all `install(pkg)` calls within `updateFirmware()`.
- Compute the Dominator Tree.
- Verify that `verify_signature(pkg)` dominates each installation call.

If dominance is not satisfied, a compile-time error is generated.

3 Rollback Protection Check

- Analyze conditional branches controlling `install(pkg)`.
- Ensure installation is control-dependent on a predicate enforcing:

$$\texttt{pkg->version} > \texttt{current\_version()}$$

If installation is reachable without satisfying the version monotonicity condition, compilation fails.

4 Trusted Source Validation Check

- Verify that `source_is_trusted(pkg)` dominates `install(pkg)`, or
- Confirm that trusted source validation is part of the controlling condition for installation.

If trusted source validation is missing on any execution path, compilation is rejected.

5 Logging Safety Check

- Scan `updateFirmware()` for calls to logging APIs such as:
  - `printf`
  - `fprintf`
  - `log`
  - `syslog`

If detected, a compile-time error is raised.

6 Weak Crypto Ban Check

- Scan `updateFirmware()` for calls to banned cryptographic functions such as:
  - `MD5`
  - `SHA1`
  - `rand`

If any banned function is found, compilation fails.