1 Introduction

Microcontroller-based embedded systems are often programmed in low-level languages and are vulnerable to control-flow hijacking. But inline control-flow integrity (CFI) enforcement solutions increase the binary size and change the memory layout. Trace-based control-flow violation detection (CFVD) offers an alternative, but existing solutions are application-oriented, requiring kernel modifications to store and analyze the trace, limiting their use to monitor privileged codes.

2 System-oriented CFVD

Monitor control-flow transfers both within and among privileged and unprivileged components.

- **Interrupt-aware**
  - Interrupts and exceptions occur asynchronously and cannot be anticipated through static analysis or dynamic training. E.g., \( <c_i, t> \)

- **Scheduling-aware**
  - Scheduler may resume any running tasks’ execution. E.g., both \( (s_i, b_i) \) and \( (s_z, c_i) \) are legitimate control-flow transfers.

- **Secure hardware tracing**
  - Prevent-privileged but potentially compromised system to enable tracing

- **Secure trace storage and analysis**
  - Secure trace from the protected system

3 System and Threat Model

- The system features a hardware trace unit. Filtering capabilities are not required.
- The system supports TEE and secure boot for code integrity.
- Attackers can exploit privileged code bugs of the protected system via memory corruptions.

4 Design Overview

- **Secure trace storage and record**
  - SCFVD verifies each indirect control-flow transfer must match an edge in the interprocedural CFG \( G_i \) or the destination must match an address in the interrupt ISR address list \( (I_i) \) or the set of task entry or re-entry list \( (Y_i) \).

5 Runtime Detection Policy

The approach that SHERLOC takes for handling each type of dereferenced instruction in the trace, \( (s, d) \): a standard trace record. \( (s_1, d_1), (s_2, d_2) \): a pair of interrupt or exception return trace records. IBT: Indirect branch table. VT: non-secure state vector table. RCS: the current task- or kernel-specific reconstructed call stack. \( Y_1 \): task entry and re-entry address list.

6 Running Example on FreeRTOS

\([\cdot]\) represents RCS with the top on the right-hand side. Black \([\cdot]\) represents the active RCS, and gray \([\cdot]\) represents an inactive RCS.