M-mode Trusted Context Extension

Mark Hill, Nick Kossifidis, Bicheng Yang, Dong Du, RISC-V TEE Task Group

Version 0.7.0, 08/2021: This document is in the Development state. Assume anything can change. For more information see: https://riscv.org/spec-state

Table of Contents

Chapter 1. Motivation

The major goal of M-mode trusted context extension is to offer a way for trusted OSes/Services/Drivers to run on M-mode, isolated from the secure monitor/firmware and from each-other, on M/U-only systems.

Threat model: Having all those elements running at the highest privilege together with the secure monitor, without memory isolation between them, increases the attack surface of the system. Also some or all of these elements may be 3rd party proprietary binaries, outside the vendor's control / trust domain. On a system with S-mode available we'd have the secure monitor on M-mode and those elements on S-mode but on an M/U-only system we are constrained and have to come up with another solution.

Chapter 2. M-Mode Trusted Context Extension

2.1. Requirements

- ePMP must be implemented
- mseccfg.MMWP must be hardwired to 1

2.2. New fields

- mseccfg.SCP (Secondary context extension present → hardwired to 1 in case this extension is implemented)
- mseccfg.PC (Primary context → on hart reset its value is 1)
- pmpcfg.S (Region associated with a secondary context)

The combination pmpcfg.S == 1 && pmpcfg.L == 0 is reserved for future use.

If this extension is not implemented, mseccfg.SCP is hardwired to 0 and mseccfg.PC / pmpcfg.S are not defined, the bits remain reserved for other uses.

2.3. Changes on enforcement of M-mode-specific (pmpcfg.L == 1) PMP rules

- Rules with pmpcfg.S == 0 are processed first, ignoring rules with pmpcfg.S == 1 (backwards compatible - no changes there)
- Rules with pmpcfg.S == 1 are processed afterwards, with lower priority (meaning if there is a rule matching the same region with pmpcfg.S == 0, the rule with pmpcfg.S == 1 is ignored)

Within each PMP ruleset / pass the standard PMP priority is followed.

In other words it's as if we had 2 separate ePMP tables merged into one (1 table, 2 rulesets).

2.4. Changes on locking of PMP rules

- Rules with pmpcfg.S == 0 follow the locking rules as defined on ePMP (backwards compatible - no changes there)
- Rules with pmpcfg.S == 1 can be added/deleted/modified when mseccfg.PC == 1, regardless of mseccfg.RLB

2.5. Other changes

- When mseccfg.PC == 0, any writes to M-mode CSRs are ignored (read-only access)
- When mseccfg.PC == 0, accessing a region with pmpcfg.S == 0 is denied
- When mseccfg.PC == 1, accessing a region with pmpcfg.S == 1 is denied
- On trap / interrupt (when we jump to mtvec), hw sets mseccfg.PC to 1
- On mret, hw sets mseccfg.PC to 0

Any writes to mseccfg.PC are ignored (WARL)

2.6. Operations

Business as usual up to the point where mseccfg.MML is set.

Firmware / secure monitor allocates a region for each Trusted OS / Service / Driver, verifies and unpacks them in memory. It can then context-switch between them on M-mode by dynamically adding / removing PMP rules with pmpcfg.L == 1 && pmpcfg.S == 1. Depending on the security requirements there can be multiple rules active at the same time, allowing for example the OS to perform direct calls to a Driver / Service, or share data regions, without going through the secure monitor.

Alternatively only one region may be active at a time, and switching between the OS and e.g. a service or driver goes through the secure monitor. Establishing communication channels between Trusted OSes / Services / Drivers is up to the sw implementation (we 'll provide guidelines on the TEE arch doc). Note that shared regions between the Trusted OS / Service / Driver and the firmware are not available (see 5b,5c above).

On context-switch, the firmware / secure monitor sets mepc to a trusted region / user region and performs mret, setting mseccfg.PC to 0 and allowing access to the non-firmware regions, while blocking access to the firmware code. Once inside the Trusted OS / Service / Driver, it won't be possible to modify any PMP rules with pmpcfg.L == 1 (assuming mseccfg.RLB is disabled as it should), nor access firmware's memory, it can only manage U mode regions

2.7. Notes (3/2/2021):

- Primary/secondary context
- Only read-only access to M-mode CSRs while on secondary context (inc. PMP)
- PMP rules with S = 0 have a higher priority (so on non-SC you'll get an access fault before reaching a rule with S = 1 for the same region)
- Think about how mtval can be useful
- Multi-core scenarios
- Interrupt masking - use of timer interrupt to avoid being stuck on a service