M-mode Secondary Context Extension

Mark Hill, Nick Kossifidis, Bicheng Yang, Dong Du, [Other Contributors], RISC-V TEE Task Group

Version 0.7.0, 10/2021: 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 secondary context extension is to offer a way for trusted OSes, services, and drivers to run on M-mode, which can be isolated from the secure monitor and firmware and from each other.

Having all those elements (OSes, services, and drivers) running at the highest privilege together with the secure monitor, without memory isolation between them, increases the attack surface of the system. Moreover, some or all of these elements may be third-party proprietary binaries outside the vendor's control and trust domain. On a system with S-mode available, we would have the secure monitor on M-mode and those elements on S-mode, but on an M/U-only system we are constrained.

We propose this proposal to provide primary and secondary contexts on M-mode, achieving isolation between different contexts. The extension targets M/U-only systems.

Chapter 2. M-Mode Secondary Context Extension

2.1. Requirements

The proposal currently has two requirements:

- ePMP must be implemented
- `mseccfg.MMWP` (i.e., Machine Mode Whitelist Policy) must be hardwired to 1

2.2. Proposal

The proposal introduces new fields in mseccfg and pmpcfg, which together implement the trusted context extension.

First, we add two new fields in mseccfg: SCP (Secondary context extension present) and PC (Primary context) fields, as shown in the following Figure.

TODO: figure-1 to show mseccfg extension

- `mseccfg.SCP` (Secondary context extension present): SCP is hardwired to 1 in case this extension is implemented.
- `mseccfg.PC` (Primary context): represents whether the current M-mode is in Primary context (its value is 1) or in Secondary Context (its value is 0).

Primary context has more permissions than secondary context (see next section for details). On hart reset, `mseccfg.PC` is set to 1. Any writes to mseccfg.PC are ignored (WARL).

Second, we add one new field in each pmpcfg, the S field (represents rules for the Secondary context), as shown in the following Figure.

7	6	5	4 3	2	1	0
L(**WARL**)	S(**WARL**)	**WIRL**	A(**WARL**)	X(**WARL**)	W(**WARL**)	R(**WARL**)
1	1	1	2	1	1	1

Figure 1. PMP configuration register format with secondary context extension.

- `pmpcfg.S`: the rule is applied to 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` and `pmpcfg.S` are not defined, the bits remain reserved for other uses.

2.3. Permission and isolation

We clarify the permission differences between primary context and secondary context here.

When M-mode is in secondary contexts (i.e., `mseccfg.PC` == 0):

- any writes to M-mode CSRs are ignored (read-only access)
- accessing a region with `pmpcfg.S` == 0 is denied

When M-mode is in the primary context (i.e., mseccfg.PC == 1):

- accessing a region with pmpcfg.S == 1 is denied

Others:

- On trap / interrupt (when we jump to mtvec), hardware sets mseccfg.PC to 1
- On mret, hardware sets mseccfg.PC to 0

2.4. Changes on enforcement of M-mode-only PMP rules

An M-mode-only PMP rule (i.e., pmpcfg.L == 1) is enforced on Machine mode and denied in Supervisor or User mode, as defined in ePMP.

- 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

This means 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, the standard PMP/ePMP priority is followed.

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

2.5. 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 and pmpcfg.L

2.6. Operations

We explain the usual operations with the extension. Assuming mseccfg.MML is set.

Firmware (or secure monitor) allocates a region for each trusted OS, services, and driver, verifies and unpacks them in memory. It can then context-switch between them on M-mode by dynamically adding or 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 or a 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 a service goes through the secure monitor. Establishing communication channels between trusted OSes, services, and drivers is up to the software implementation. Note that shared regions between the trusted OSes, services, drivers and the firmware are not available.

On context-switching, the firmware (secure monitor) sets mepc to a trusted region or 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 OSes, services, drivers, 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.