Arm v8.5-A CPU updates

Version 1.3

Version 1.3

May 21. 2019

© 2019 Arm Limited

Summary

- As part of the Arm v8.5-A Architecture update, Arm has added some features meant to address security issues disclosed by the Google Project Zero team and elsewhere.
- That full architecture specification update is confidential, but is scheduled to be included in the full v8-A Arm ARM update later this year.
  - The update has been provided under NDA to Arm licensees.
- Some of those features have been implemented in maintenance releases of affected Cortex-A CPUs, and have also been implemented in Neoverse CPUs.
- Arm has also developed Arm Trusted Firmware (ATF) and Linux kernel patches that will take advantage of those implemented updates.
- While the features are described in the documentation for the affected CPUs, or in the Security Whitepaper, this slide deck will provide additional information about those features.

© 2019 Arm Limited

Implemented features list:

- Barrier instruction to mitigate variant 1.
- New ID fields in system registers that identify whether the CPU has had hardware mitigations for variant 2 or variant 3 (and 3A) applied.
- Barrier instructions to mitigate variant 4.
- New PSTATE bit to turn off memory disambiguation (which allows variant 4).
  - Replaces implementation defined control with architecturally defined control.
- New instruction to access new PSTATE bit.
- New ID field in system register to identify whether new PSTATE bit has been implemented.
- Note: not all features will be implemented on all existing Cortex-A CPUs.

© 2019 Arm Limited

Note on updated register fields

- The updated ID fields use bits that were reserved as zero (RES0) in previous implementations of the architecture.
- If software accesses those fields on versions of the CPUs that have not been updated, there will not be a problem.
- Reading the fields as zero does not necessarily mean that the CPU is vulnerable; for CPUs not affected by the variants in question, they are (still) not vulnerable.
  - There is a complete list of CPUs available in the Security Update pages at Developer.Arm.com
- The proper way for software to process those fields is:

  IF ((CPU is vulnerable) AND (ID field is zero))

  THEN apply software mitigation;

© 2019 Arm Limited

Variant 1 barrier

- There is a barrier instruction which allows mitigation of code that could be vulnerable to variant 1 attacks
- CSDB: No instruction, other than a branch instruction, appearing in program order after the CSDB can be speculatively executed using the results of any data value predictions of any instructions, or comparison results of any instructions (other than conditional branch instructions)
- This instruction is implemented using existing options for the DSB instruction in both AArch32 and AArch64, and is available on all affected Cortex-A CPUs
- More details about this instruction are found in the Cache Speculation White Paper available on the Arm website.

© 2019 Arm Limited

Variant 2 updates

- There is a new CSV2 field defined in bits [59:56] of the ID_AA64PFR0_EL1 register
  - Also implemented in bits [19:16] of the ID_PFR0 register in Aarch32
- This field indicates whether the CPU allows a branch prediction result trained in one context to be practically used in another context.
- Essentially, the cause of variant 2.
- Values:
  - 0000: the CPU does not indicate whether mitigations have been applied
  - 0001: the CPU does not allow branch targets created in one hardware context to be easily used in a different hardware context
- Implemented on maintenance releases of Cortex-A72, -A73, -A75, and -A76, and on Neoverse N1.

© 2019 Arm Limited

Variant 3 and 3A updates

- There is a new CSV3 field defined in bits [63:60] of the ID_AA64PFR0_EL1 register
  - Also implemented in bits [3:0] of the ID_PFR0 register in Aarch32
- That field indicates whether the CPU causes data loaded under speculation with a permission or domain fault to be used to form an address that can be used to cause cache allocation
  - Essentially, the cause of variant 3 and 3A
- Values:
  - 0000: the CPU does not indicate whether mitigations have been applied
  - 0001: the CPU does not allow the behavior that permits Variant 3
- Implemented on maintenance releases of Cortex-A72, -A73, -A75, and -A76, and on the Neoverse N1.

© 2019 Arm Limited

Variant 4 barriers

- There are two barrier instructions which allow easier mitigation of code that could be vulnerable to variant 4 attacks
- SSBB: Any stores before the SSBB using a virtual address will not be bypassed by any speculative executions of a load after the SSBB to the same virtual address.
- PSSBB: Any stores before the PSSBB using a physical address will not be bypassed by any speculative executions of a load after the PSSBB to the same physical address.
- These instructions are implemented using existing options for the DSB instruction in both AArch32 and AArch64, and is available on all affected Cortex-A CPUs.
- More details about these instructions are found in the Cache Speculation White Paper available on the Arm website.
- An optimized version of this instruction has been implemented on a maintenance release of the Cortex-A76, and on the Neoverse N1

© 2019 Arm Limited

Variant 4 memory disambiguation control (1)

- A new PSTATE/CPSR bit has been added: SSBS (Speculative Store Bypass Safe).
  - Also supported in SPSR for exception entry/return
- Provides an architecturally defined control to enable/disable memory disambiguation.
  - Current controls are unique (per CPU) implementation defined in system registers
- When set, enables memory disambiguation on speculative stores and loads.
- AArch64 adds MSR and MRS variants to read and write the SSBS bit in PSTATE.
  - If those new variants are not present, then the bit is set by setting the appropriate bit in SPSR, and then performing an ERET instruction (which copies the bit into PSTATE)
- Controls for setting/clearing bit on entry/exit to debug state also implemented

© 2019 Arm Limited

Variant 4 memory disambiguation control (2)

- There is a new SSBS field defined in bits [7:4] of the ID_AA64PFR1_EL1 register
  - Also implemented in bits [7:4] of the ID_PFR2 register in Aarch32
- That field indicates whether the new SSBS bit and associated instructions have been implemented
- Values:
  - 0000: SSBS bit has not been added to PSTATE/CPSR
  - 0001: SSBS bit is present in PSTATE, but MRS/MSR instructions to access directly (in AArch64) are not implemented
  - 0010: SSBS bit and associated MRS/MSR instructions are implemented
- Neoverse N1 r3p0 implements the PSTATE.SSBS bit, as well as the MRS/MSR instructions
- Maintenance releases of Cortex-A55 and Cortex-A76 include the bit in PSTATE, but not the MRS/MSR instructions to access the bit directly
  - Included on Cortex-A55 for software compatibility only (CPU is not vulnerable to variant 4)

© 2019 Arm Limited

CPU summary

CPU	Version number	Variant 2 ID field	Variant 3/3A ID field	Variant 4 SSBS control	Notes
Cortex-A55	r2p0 – also as ECO	No (implemented as 0)	No (implemented as 0)	PSTATE.SSBS bit only – no MSR/MRS	SSBS bit added for software compatibility with other v8.2 CPUs.
Cortex-A72	r1p0	Yes	Yes	No	
Cortex-A73	r1p0	Yes	Yes	No	
Cortex-A75	r3p0	Yes	Yes	No	
Cortex-A76	r3p0	Yes	Yes	PSTATE.SSBS bit only – no MSR/MRS	Bit must be set by (1) setting bit in SPSR and (2) doing an ERET to set PSTATE.
Neoverse N1	r3p0	Yes	Yes	PSTATE.SSBS bit, and MSR/MRS instructions	

© 2019 Arm Limited
arm

The Arm trademarks featured in this presentation are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. All other marks featured may be trademarks of their respective owners.

www.arm.com/company/policies/trademarks

© 2019 Arm Limited