Cache Speculation Side-channels Linux kernel mitigations

These slides are based on information available in the Arm Whitepaper @ https://developer.arm.com/support/security-update

Date: 10 September 2018

Version : 1.7

Linux kernel mitigations

Variant	Exploit	Kernel Mitigation
1 Spectre	Bypassing software checking of untrusted values	Inhibit speculation of out-of-bounds-pointer at loads/pointer access Requires identification of (multiple) vulnerable locations Variant 1 mitigations are expected to be an ongoing process (cross-architecture)
2 Spectre	Speculation through training branch predictors	Invalidate branch predictors at key points in kernel corresponding to trust boundaries: Context switch, VM exit
3 Meltdown	Speculative reads of inaccessible data	Protection via Kernel Page Table Isolation (KPTI) Switch entire page tables between userspace and kernel (two sets of page tables) on OS entry/exit
3a Meltdown	Speculative reads of system register values (Arm specific)	Variant 3 mitigation provides protection for vector base address. Investigating if additional software mitigations are necessary.
4	Speculative load overtaking a store	Static: Memory disambiguation is permanently disabled in TF, at boot. Dynamic: Memory disambiguation can be disabled dynamically for specific userspace (EL0) workloads, at either firmware (SSBD) or kernel (SSBS) level

Arm Linux Kernel mitigations update

Arm has developed Linux kernel mitigations for Variants 1/2/3/4

Arm64 Variant 1/2/3 patches are now merged in **linux-4.16** (v1.1 patches merged in **4.19**)

Arm64 Variant 4 patches (SSBD) are now merged in **linux-4.18**.

Arm64 Variant4 patches (SSBS) are now posted on Linux Kernel mailing list.

Patches support Arm64 (Arm v8.x-A cores) and initial support for Arm32

Linaro are providing backports for Variants 1/2/3:

- **www.linaro.org/downloads/security/**
- git.linaro.org/kernel/speculation-fixes-staging (currently 4.4 / 4.9 / 4.14)

Arm has provided backports for Variant 4 to 4.9/4.14/4.17

Android backports to 3.18 / 4.4 / 4.9 / 4.14 available on https://android.googlesource.com/kernel/common.git

arm

Arm 'kpti' branch patch stack

Arm has produced a patch stack on top of linux-4.15:
https://git.kernel.org/pub/scm/linux/kernel/git/arm64/linux.git/log/?h=kpti

'kpti' branch is a staging integration which includes Variant 1/2/3 support
Full patch stack listed on next 3 slides [this is a reference for kernel backporting]

All 'kpti' branch patches are merged in 4.16 and later

[Variant */Spectre */Meltdown] arm64: **Add README describing the branch** security_fixes_20180208 kpti

README

[Variant 2/Spectre-v2] arm: KVM: Invalidate icache on guest exit for Cortex-A15
[Variant 2/Spectre-v2] arm: Invalidate icache on prefetch abort outside of us...
[Variant 2/Spectre-v2] arm: Add icache invalidation on switch_mm for Cortex-A15
[Variant 2/Spectre-v2] arm: KVM: Invalidate BTB on guest exit for Cortex-A12/A17
[Variant 2/Spectre-v2] arm: Invalidate BTB on prefetch abort outside of user ...
[Variant 2/Spectre-v2] arm: Add BTB invalidation on switch_mm for Cortex-A9, ...

Variant 2

arm support for v7-A cores using inline code sequences

[Variant 2/Spectre-v2] arm64: Kill PSCI_GET_VERSION as a variant-2 workaround arm64-spectre-meltdown-for-4.15-stable
[Variant 2/Spectre-v2] arm64: Add ARM_SMCCC_ARCH_WORKAROUND_1 BP hardening su...
[Variant 2/Spectre-v2] arm/arm64: smccc: Implement SMCCC v1.1 inline primitive
[Variant 2/Spectre-v2] arm/arm64: smccc: Make function identifiers an unsigne...
[Variant 2/Spectre-v2] firmware/psci: Expose SMCCC version through psci_ops
[Variant 2/Spectre-v2] firmware/psci: Expose PSCI conduit
[Variant 2/Spectre-v2] arm64: KVM: Add SMCCC_ARCH_WORKAROUND_1 fast handling
[Variant 2/Spectre-v2] arm64: KVM: Report SMCCC_ARCH_WORKAROUND_1 BP hardenin...
[Variant 2/Spectre-v2] arm/arm64: KVM: Turn kvm_psci_version into a static in...
[Variant 2/Spectre-v2] arm64: KVM: Make PSCI_VERSION a fast path
[Variant 2/Spectre-v2] arm/arm64: KVM: Advertise SMCCC v1.1
[Variant 2/Spectre-v2] arm/arm64: KVM: Implement PSCI 1.0 support
[Variant 2/Spectre-v2] arm/arm64: KVM: Add smccc accessors to PSCI code
[Variant 2/Spectre-v2] arm/arm64: KVM: Add PSCI_VERSION helper
[Variant 2/Spectre-v2] arm/arm64: KVM: Consolidate the PSCI include files
[Variant 2/Spectre-v2] arm64: KVM: Increment PC after handling an SMC trap
[Variant 2/Spectre-v2] arm64: Branch predictor hardening for Cavium ThunderX2
[Variant 2/Spectre-v2] arm64: Implement branch predictor hardening for Falkor
[Variant 2/Spectre-v2] arm64: Implement branch predictor hardening for affect...
[Variant 2/Spectre-v2] arm64: cputype: Add missing MIDR values for Cortex-A72...
[Variant 2/Spectre-v2] arm64: entry: Apply BP hardening for suspicious interr...
[Variant 2/Spectre-v2] arm64: entry: Apply BP hardening for high-priority syn...

arm64 support for v8-A cores using SMCCC

arm

[Variant 2/Spectre-v2] arm64: KVM: Use per-CPU vector when BP hardening is en...
[Variant 2/Spectre-v2] arm64: Move BP hardening to check_and_switch_context
[Variant 2/Spectre-v2] arm64: Add skeleton to harden the branch predictor aga...
[Variant 2/Spectre-v2] arm64: Move post_ttbr_update_workaround to C code
[Variant 2/Spectre-v2] drivers/firmware: Expose psci_get_version through psci...
[Variant 2/Spectre-v2] arm64: cpufeature: Pass capability structure to ->enab...
[Variant 2/Spectre-v2] arm64: Run enable method for errata work arounds on la...
[Variant 2/Spectre-v2] arm64: cpufeature: __this_cpu_has_cap() shouldn't stop...

Variant 2 – arm64

[Variant 1/Spectre-v1] arm64: futex: Mask __user pointers prior to dereference
[Variant 1/Spectre-v1] arm64: uaccess: Mask __user pointers for __arch_{clear...
[Variant 1/Spectre-v1] arm64: uaccess: Don't bother eliding access_ok checks ...
[Variant 1/Spectre-v1] arm64: uaccess: Prevent speculative use of the current...
[Variant 1/Spectre-v1] arm64: entry: Ensure branch through syscall table is b...
[Variant 1/Spectre-v1] arm64: Use pointer masking to limit uaccess speculation
[Variant 1/Spectre-v1] arm64: Make USER_DS an inclusive limit
[Variant 1/Spectre-v1] arm64: Implement array_index_mask_nospec()
[Variant 1/Spectre-v1] arm64: barrier: Add CSDB macros to control data-value ...
[Variant 1/Spectre-v1] array_index_nospec: Sanitize speculative array de-refe...
[Variant 1/Spectre-v1] Documentation: Document array_index_nospec

Variant 1 – arm64
Use data speculation barrier

[Variant 3/Meltdown] perf: arm_spe: Fail device probe when arm64_kernel_unmap...
[Variant 3/Meltdown] arm64: idmap: Use "awx" flags for .idmap.text .pushsecti...
[Variant 3/Meltdown] arm64: entry: Reword comment about post_ttbr_update_work...
[Variant 3/Meltdown] arm64: Force KPTI to be disabled on Cavium ThunderX
[Variant 3/Meltdown] arm64: kpti: Add ->enable callback to remap swapper usin...
[Variant 3/Meltdown] arm64: mm: Permit transitioning from Global to Non-Globa...
[Variant 3/Meltdown] arm64: kpti: Make use of nG dependent on arm64_kernel_un...
[Variant 3/Meltdown] arm64: Turn on KPTI only on CPUs that need it

Variant 3 – arm64
KPTI

arm

[Variant 3/Meltdown] arm64: cputype: Add MIDR values for Cavium ThunderX2 CPUs
[Variant 3/Meltdown] arm64: kpti: Fix the interaction between ASID switching ...
[Variant 3/Meltdown] arm64: mm: Introduce TTBR_ASID_MASK for getting at the A...
[Variant 3/Meltdown] arm64: capabilities: Handle duplicate entries for a capa...
[Variant 3/Meltdown] arm64: Take into account ID_AA64PFR0_EL1.CSV3
[Variant 3/Meltdown] arm64: Kconfig: Reword UNMAP_KERNEL_AT_EL0 kconfig entry
[Variant 3/Meltdown] arm64: Kconfig: Add CONFIG_UNMAP_KERNEL_AT_EL0
[Variant 3/Meltdown] arm64: use RET instruction for exiting the trampoline
[Variant 3/Meltdown] arm64: kaslr: Put kernel vectors address in separate dat...
[Variant 3/Meltdown] arm64: entry: Add fake CPU feature for unmapping the ker...
[Variant 3/Meltdown] arm64: tls: Avoid unconditional zeroing of tpidrro_el0 f...
[Variant 3/Meltdown] arm64: cpu_errata: Add Kryo to Falkor 1003 errata
[Variant 3/Meltdown] arm64: erratum: Work around Falkor erratum #E1003 in tra...
[Variant 3/Meltdown] arm64: entry: Hook up entry trampoline to exception vectors
[Variant 3/Meltdown] arm64: entry: Explicitly pass exception level to kernel_...
[Variant 3/Meltdown] arm64: mm: Map entry trampoline into trampoline and kern...
[Variant 3/Meltdown] arm64: entry: Add exception trampoline page for exceptio...
[Variant 3/Meltdown] arm64: mm: Invalidate both kernel and user ASIDs when pe...
[Variant 3/Meltdown] arm64: mm: Add arm64_kernel_unmapped_at_el0 helper
[Variant 3/Meltdown] arm64: mm: Allocate ASIDs in pairs
[Variant 3/Meltdown] arm64: mm: Fix and re-enable ARM64_SW_TTBR0_PAN
[Variant 3/Meltdown] arm64: mm: Rename post_ttbr0_update_workaround
[Variant 3/Meltdown] arm64: mm: Remove pre_ttbr0_update_workaround for Falkor...
[Variant 3/Meltdown] arm64: mm: Move ASID from TTBR0 to TTBR1
[Variant 3/Meltdown] arm64: mm: Temporarily disable ARM64_SW_TTBR0_PAN
[Variant 3/Meltdown] arm64: mm: Use non-global mappings for kernel space

Linux 4.15

Variant 3 – arm64

KPTI

arm

arm/arm64: smccc: Add SMCCC-specific return codes

arm64: Call ARCH_WORKAROUND_2 on transitions between EL0 and EL1

arm64: Add per-cpu infrastructure to call ARCH_WORKAROUND_2

arm64: Add ARCH_WORKAROUND_2 probing

arm64: Add 'ssbd' command-line option

arm64: ssbd: Add global mitigation state accessor

arm64: ssbd: Skip apply_ssbd if not using dynamic mitigation

arm64: ssbd: Restore mitigation status on CPU resume

arm64: ssbd: Introduce thread flag to control userspace mitigation

arm64: ssbd: Add prctl interface for per-thread mitigation

arm64: KVM: Add HYP per-cpu accessors

arm64: KVM: Add ARCH_WORKAROUND_2 support for guests

arm64: KVM: Handle guest's ARCH_WORKAROUND_2 requests

arm64: KVM: Add ARCH_WORKAROUND_2 discovery through ARCH_FEATURES_FUNC_ID

arm64: KVM: Move VCPU_WORKAROUND_2_FLAG macros to the top of the file

Variant 4 – arm64
ARCH_WORKAROUND_2
SSBD & prctl
Linux 4.18

arm64: fix possible spectre-v1 write in ptrace_hbp_set_event()

KVM: arm/arm64: vgic: Fix possible spectre-v1 write in vgic_mmio_write_apr()

Variant 1.1 – arm64

Armv8.5 SSBS mitigation to Variant 4 initially posted: http://lists.infradead.org/pipermail/linux-arm-kernel/2018-August/598923.html

arm

Overview of Linux kernel changes

Arm Cortex licensees should take changes from released kernels (from 4.16 on), or backports from Linaro

- Changes are in core and generic code and do not need to be further modified*
- Linaro providing backports to LTS (Long Term Support) kernels 4.4 / 4.9 / 4.14.
  See https://www.linaro.org/downloads/security/ for further details.

**Arm has been working with Arm architecture licensees (those companies who design their own CPUs that implement the Arm architecture. These CPUs do not use the Cortex brand in their model names. Small additional patches may be required, contact your silicon partner.*

It should not be necessary for silicon partners or OEMs using Arm designed Cortex-A CPUs to modify the content of the patches.

Variant 1: Kernel implementation detail

Aligned with Linux-4.16 mainline implementation for the new **array_index_mask_nospec()** macro

arm64 patches provide:

- An arm64 version of **array_index_mask_nospec()** to suppress any compiler optimisations that could introduce unwanted speculative paths
- Masking of the syscall number to restrict speculation through the syscall table
- Masking of __user pointers prior to deference in uaccess routines

Vulnerable sites in kernel need to be identified: active discussions on LKML about how to identify locations in the kernel. This is an architecture-agnostic issue.

Fixes for Variant1 in the kernel are expected to be an ongoing effort, perhaps many months.
Vendors should expect to issue further kernel updates.

Variant 2: Speculation through training branch predictors

Vulnerable points are ‘trust boundaries’ where level of trust changes:

- Entry to kernel: switch between processes, VM entry/exit
- Also affects userspace: JIT to app, etc.

Kernel mitigation relies on invalidation of Branch Predictor at the above trust boundaries

How this is done is subject to Implementation Defined sequences
(full detail here https://developer.arm.com/support/security-update)

- 64-bit uses new SMC call. **This requires support in firmware:** https://developer.arm.com/support/arm-security-updates/speculative-processor-vulnerability/downloads/firmware-interfaces-for-mitigating-cache-speculation-vulnerabilities
- 32-bit Armv7-A Cortex & Armv8 Cortex-A73/A75 use in-line BP-invalidate sequences (e.g. ARMv7-A ‘BPIALL’)
- 32-bit Armv8 Cortex-A57/A72 call into EL3 firmware (through SMCCC_ARCH_WORKAROUND_1) to invalidate the BP. **This requires support in firmware.**

arm

Variant 3: KPTI

Background: ‘Kaiser’ is a 2017 proposal to improve KASLR (Kernel Address Space Layout Randomization)

- Address-space layout randomization (ASLR) is a well-known technique to make exploits harder by placing various objects at random, rather than fixed, addresses. Linux has long had ASLR for user-space programs, but Kees Cook [Google security] would like to see it applied to the kernel itself as well
- Update: KASLR enabled for Android Common Kernel 4.4 / 4.9 / 4.14

For more details on Kaiser see Austria paper: **https://gruss.cc/files/kaiser.pdf**

Kaiser was public earlier in 2017 – it proposes **Kernel Page Table Isolation**

Basically **Kaiser == KPTI** and this is the complete Variant 3 mitigation

Variant 3: KPTI

The KPTI patches in the stack protect against Variant 3
(The only Arm designed core affected is Cortex-A75, but Variant 3 may affect other Arm Architecture licensee implementations)

KPTI is implemented for arm64 and is made the default in the Linux kernel as:

- Underlying KPTI support is compiled in by default (use of TTBR0/1 etc.)
- KPTI additionally hardens kernel KASLR (address randomization) which improves security
  (see Kaiser paper for details)
- **Arm64 is particularly efficient at KPTI** with two TTBR registers & ASID & VMID
  First benchmarking indicates minimal performance overhead
- It also provides Variant 3a protection for vector base register

Applying KPTI is not essential for Arm-designed cores, except for Cortex-A75 r0/r1/r2
However, Arm recommends KPTI is enabled by default if at least one CPU in the system is affected.

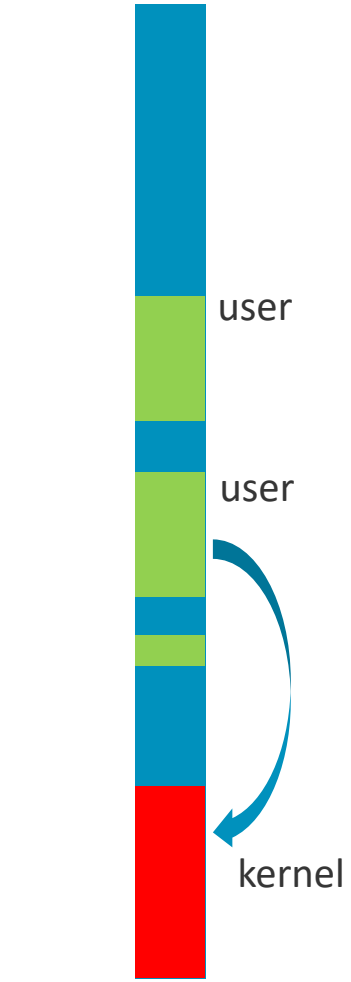
arm

Without KPTI

unmapped

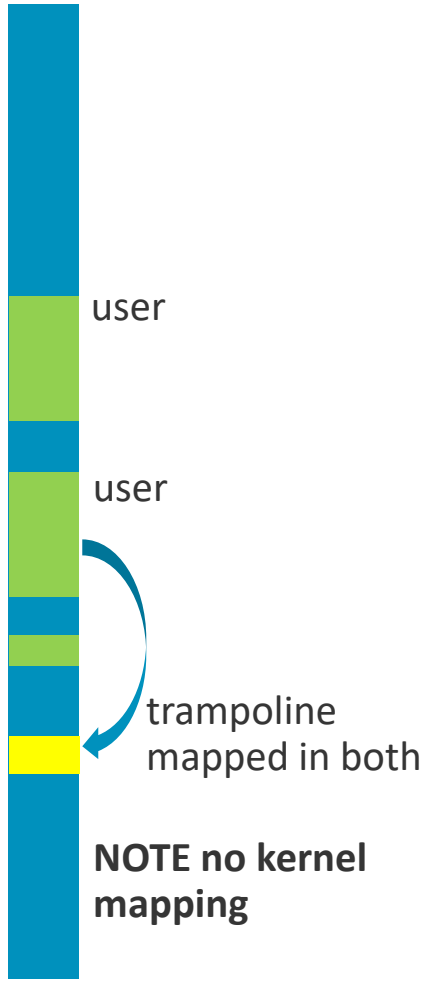
user mapping

kernel mapping

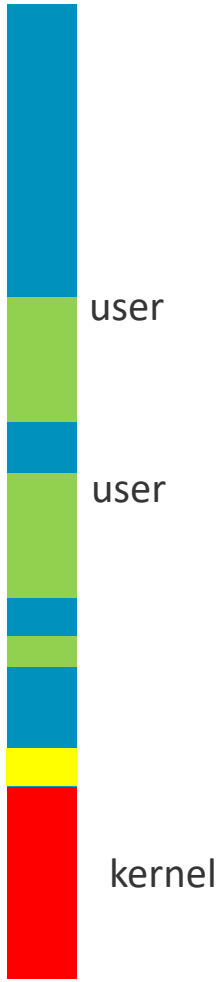


user & kernel mode

With KPTI (highly simplified)



user mode



kernel mode

arm

Variant 3a: System register read

This is not considered to be a security risk, as system registers do not hold ‘interesting’ information.

This exploit could obtain the vector base address (VBAR_EL1, VBAR_EL2...) – probably the most interesting value

When KPTI is enabled, VBAR_EL1 has a fixed value, and doesn't reflect the location of the kernel. This mitigates VBAR access on Cortex-A57 and Cortex-A72 arm64 kernels.

Cortex-A15 remains susceptible (no KPTI on arm32). However, this is not considered to be a security risk because arm32 does not support KASLR, and with a fixed kernel there are various ways to discover kernel addresses in addition to looking in VBAR registers – e.g. obtain the kernel image, etc.

Kernel configuration to enable mitigations

Variant 1	**Applied by default:** New Arm speculation barrier sequence used where needed. **64-bit:** Infrastructure support for arm64 speculation barrier & uaccess mitigations
Variant 2	**Applied by default:** **64-bit:** New SMCCC_ARCH_WORKAROUND_1 call must be implemented by firmware **32-bit: Applied by default:** uses 32-bit architectural BPIALL (most Arm v7-A cores) Can be disabled if **CONFIG_EXPERT=Y** and **CONFIG_HARDEN_BRANCH_PREDICTOR** is not set
Variant 3 (64-bit only)	**Applied by default:** KPTI is enabled for cores that need it, or when KASLR is enabled as it improves kernel KASLR and mitigates Variant 3a Can be disabled if **CONFIG_EXPERT=Y** and **CONFIG_UNMAP_KERNEL_AT_EL0** is not set Kernel command-line option 'kpti=off' allows KPTI to be turned off (no reason to do this)
Variant 4	Static: Firmware must turn off memory disambiguation and advertise via SMCCC_ARCH_FEATURES (existing cores) Dynamic: Firmware must implement full SMCCC_ARCH_WORKAROUND_2 (for future cores)

Summary: no special CONFIG options required to enable mitigations
Arm Linux kernel adopts a 'secure by default' policy

arm

Variant 4: SSBD (Speculative Store Bypass Disable) mitigation

New command-line option: ssbd=n

- n=force-on: force enable always
- n=kernel: enable workaround dynamically on kernel entry at EL1 (default)
- n=force-off: disable workaround (not recommended)

prctl interface to allow individual userspace tasks to enable workaround if they need protection (e.g. userspace JIT will want to enable protection for itself)

Each EL0 application that would like to be mitigated needs to explicitly opt-in for the mitigation by calling the prctl interface

prctl interface can be used to query Variant 4 mitigation status

Boot log message appears when speculative store bypass workaround active

Variant 4: SSBS (Speculative Store Bypass Safe) mitigation

Armv8.5 introduces a new PSTATE bit known as Speculative Store Bypass Safe (SSBS) which can be used as an architecturally defined control to enable/disable memory disambiguation.

- On CPUs with support for PSTATE.SSBS, the kernel can toggle the SSBD state without needing to call into firmware.
- Additionally, a CPU may provide instructions to manipulate PSTATE.SSBS directly, so that userspace can toggle the SSBS control without trapping to the kernel.
- Kernel probes for the existence of SSBS and advertise the new instructions to userspace if they exist (via HWCAP)

Note: Variant 4 SSBS patches under discussion on kernel mailing lists, so liable to change.

Arm 32-bit mitigations

Arm is committed to support of 32-bit Arm Linux

Variant 1: Generic mitigations exist from mainline 4.16 Variant 1 support.

Additional kernel support has been implemented into **linux-4.18** and **linux-4.19**

Fixes for Variant1 in the kernel are expected to be an ongoing effort, perhaps many months. Vendors should expect to issue further kernel updates

Variant 2:
For v7-A cores: Kernel mitigations are published and use direct BP-Invalidate sequences into kernel.
For v8.x-A cores: Arm32 kernel mitigations are published on Armv8-A Cortex cores, but support for big.LITTLE systems is still under development.

Cortex-A73/A75 apply BP-invalidation directly into kernel

Cortex-A57/A72 involve SMCC and trapping into EL3 Firmware, so they require a firmware update in addition to kernel patches

Variant 3: The only Arm Cortex core affected is Cortex-A75, no other Cortex-A cores are impacted

Arm expects 64-bit kernels (which include mitigations) to be used on Cortex-A75 systems, and no Arm32 KPTI support is necessary for other Arm Cortex cores.

There is no KASLR on Arm 32-bit Linux, so KPTI is not relevant.

Variant 4: All 32-bit cores have been statically mitigated into Firmware, so no kernel patches required so far.

Kernel patch status summary

	arm64	**Arm32**
Variant 1	arm64 speculation barrier Merged in **linux-4.16** v1.1: merged in **linux-4.19** Expect further updates	Generic mitigations available Further support merged in **linux-4.18** and **linux-4.19** v1.1: patches under public review Expect further updates
Variant 2	Uses SMCCC Merged in **linux-4.16**	v7-A cores: Uses BP-Invalidate code sequences. Merged in **linux-4.18** v8-A cores: Merged in **linux-4.18** (Cortex-A73/A75 BP invalidation) (Cortex-A57/A72 SMC call into Firmware) **big.LITTLE support In-development**
Variant 3	Merged in **linux-4.16**	No KPTI required
Variant 4	SSBD merged in **linux-4.18** SSBS Patches on LKML mailing list	v7-A cores: Not affected v8-A cores: Will follow arm64

arm

Summary

Arm has produced patchsets for Linux Kernel mitigations for arm64 and arm32

- Firstly, partners should investigate specific CPU susceptibility & device / application risk
- Mitigations for Variant 1/2/3 are now available for arm64 and are merged in linux-4.16 (var1.1 mitigation will be available in linux-4.19)
- Mitigations for Variant 4 (SSBD) is now available for arm64 and merged in linux-4.18
- Mitigations for Variant 4 (SSBS) is now under public review on LKML mailing list
- For Variant 2, a firmware update will be required to support SMCCC v1.1
- Backports available from Linaro & Google
- Opensource process means further updates may be required (particularly for Variant 1 where generic C code kernel mitigations will evolve over time)

arm

The information provided in this document is based on Arm's present understanding of the relevant vulnerabilities and mitigations, and is subject to change as additional information is revealed and as additional analyses are performed.