TAA - TSX Asynchronous Abort

TAA is a hardware vulnerability that allows unprivileged speculative access to data which is available in various CPU internal buffers by using asynchronous aborts within an Intel TSX transactional region.

Affected processors

This vulnerability only affects Intel processors that support Intel Transactional Synchronization Extensions (TSX) when the TAA_NO bit (bit 8) is 0 in the IA32_ARCH_CAPABILITIES MSR. On processors where the MDS_NO bit (bit 5) is 0 in the IA32_ARCH_CAPABILITIES MSR, the existing MDS mitigations also mitigate against TAA.

Whether a processor is affected or not can be read out from the TAA vulnerability file in sysfs. See :ref:`tsx_async_abort_sys_info`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\hw-vuln\(linux-master) (Documentation) (admin-guide) (hw-vuln) tsx_async_abort.rst`, **line 19**); ***backlink***

Unknown interpreted text role "ref".

Related CVEs

The following CVE entry is related to this TAA issue:

CVE-2019-11135	TAA	TSX Asynchronous Abort (TAA) condition on some microprocessors utilizing speculative execution may allow an authenticated user to potentially enable information disclosure via a side channel with local access.

Problem

When performing store, load or L1 refill operations, processors write data into temporary microarchitectural structures (buffers). The data in those buffers can be forwarded to load operations as an optimization.

Intel TSX is an extension to the x86 instruction set architecture that adds hardware transactional memory support to improve performance of multi-threaded software. TSX lets the processor expose and exploit concurrency hidden in an application due to dynamically avoiding unnecessary synchronization.

TSX supports atomic memory transactions that are either committed (success) or aborted. During an abort, operations that happened within the transactional region are rolled back. An asynchronous abort takes place, among other options, when a different thread accesses a cache line that is also used within the transactional region when that access might lead to a data race.

Immediately after an uncompleted asynchronous abort, certain speculatively executed loads may read data from those internal buffers and pass it to dependent operations. This can be then used to infer the value via a cache side channel attack.

Because the buffers are potentially shared between Hyper-Threads cross Hyper-Thread attacks are possible.

The victim of a malicious actor does not need to make use of TSX. Only the attacker needs to begin a TSX transaction and raise an asynchronous abort which in turn potentially leaks data stored in the buffers.

More detailed technical information is available in the TAA specific x86 architecture section: :ref:`Documentation/x86/tsx_async_abort.rst <tsx_async_abort>`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\hw-vuln\(linux-master) (Documentation) (admin-guide) (hw-vuln) tsx_async_abort.rst`, **line 65**); ***backlink***

Unknown interpreted text role "ref".

Attack scenarios

Attacks against the TAA vulnerability can be implemented from unprivileged applications running on hosts or guests.

As for MDS, the attacker has no control over the memory addresses that can be leaked. Only the victim is responsible for bringing data to the CPU. As a result, the malicious actor has to sample as much data as possible and then postprocess it to try to infer any useful information from it.

A potential attacker only has read access to the data. Also, there is no direct privilege escalation by using this technique.

TAA system information

The Linux kernel provides a sysfs interface to enumerate the current TAA status of mitigated systems. The relevant sysfs file is:

/sys/devices/system/cpu/vulnerabilities/tsx_async_abort

The possible values in this file are:

'Vulnerable'	The CPU is affected by this vulnerability and the microcode and kernel mitigation are not applied.
'Vulnerable: Clear CPU buffers attempted, no microcode'	The system tries to clear the buffers but the microcode might not support the operation.
'Mitigation: Clear CPU buffers'	The microcode has been updated to clear the buffers. TSX is still enabled.
'Mitigation: TSX disabled'	TSX is disabled.
'Not affected'	The CPU is not affected by this issue.

Best effort mitigation mode

If the processor is vulnerable, but the availability of the microcode-based mitigation mechanism is not advertised via CPUID the kernel selects a best effort mitigation mode. This mode invokes the mitigation instructions without a guarantee that they clear the CPU buffers.

This is done to address virtualization scenarios where the host has the microcode update applied, but the hypervisor is not yet updated to expose the CPUID to the guest. If the host has updated microcode the protection takes effect; otherwise a few CPU cycles are wasted pointlessly.

The state in the tsx_async_abort sysfs file reflects this situation accordingly.

Mitigation mechanism

The kernel detects the affected CPUs and the presence of the microcode which is required. If a CPU is affected and the microcode is available, then the kernel enables the mitigation by default.

The mitigation can be controlled at boot time via a kernel command line option. See :ref:`taa_mitigation_control_command_line`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\hw-vuln\(linux-master) (Documentation) (admin-guide) (hw-vuln) tsx_async_abort.rst`, line 136); *backlink*

Unknown interpreted text role "ref".

Virtualization mitigation

Affected systems where the host has TAA microcode and TAA is mitigated by having disabled TSX previously, are not vulnerable regardless of the status of the VMs.

In all other cases, if the host either does not have the TAA microcode or the kernel is not mitigated, the system might be vulnerable.

Mitigation control on the kernel command line

The kernel command line allows to control the TAA mitigations at boot time with the option "tsx_async_abort=". The valid arguments for this option are:

off	This option disables the TAA mitigation on affected platforms. If the system has TSX enabled (see next parameter) and the CPU is affected, the system is vulnerable.
full	TAA mitigation is enabled. If TSX is enabled, on an affected system it will clear CPU buffers on ring transitions. On systems which are MDS-affected and deploy MDS mitigation, TAA is also mitigated. Specifying this option on those systems will have no effect.
full,nosmt	The same as tsx_async_abort=full, with SMT disabled on vulnerable CPUs that have TSX enabled. This is the complete mitigation. When TSX is disabled, SMT is not disabled because CPU is not vulnerable to cross-thread TAA attacks.

Not specifying this option is equivalent to "tsx_async_abort=full". For processors that are affected by both TAA and MDS, specifying just "tsx_async_abort=off" without an accompanying "mds=off" will have no effect as the same mitigation is used for both vulnerabilities.

The kernel command line also allows to control the TSX feature using the parameter "tsx=" on CPUs which support TSX control. MSR_IA32_TSX_CTRL is used to control the TSX feature and the enumeration of the TSX feature bits (RTM and HLE) in CPUID.

The valid options are:

off	Disables TSX on the system. Note that this option takes effect only on newer CPUs which are not vulnerable to MDS, i.e., have MSR_IA32_ARCH_CAPABILITIES.MDS_NO=1 and which get the new IA32_TSX_CTRL MSR through a microcode update. This new MSR allows for the reliable deactivation of the TSX functionality.
on	Enables TSX. Although there are mitigations for all known security vulnerabilities, TSX has been known to be an accelerator for several previous speculation-related CVEs, and so there may be unknown security risks associated with leaving it enabled.
auto	Disables TSX if X86_BUG_TAA is present, otherwise enables TSX on the system.

Not specifying this option is equivalent to "tsx=off".

The following combinations of the "tsx_async_abort" and "tsx" are possible. For affected platforms tsx=auto is equivalent to tsx=off and the result will be:

tsx=on	tsx_async_abort=full	The system will use VERW to clear CPU buffers. Cross-thread attacks are still possible on SMT machines.
tsx=on	tsx_async_abort=full,nosmt	As above, cross-thread attacks on SMT mitigated.
tsx=on	tsx_async_abort=off	The system is vulnerable.
tsx=off	tsx_async_abort=full	TSX might be disabled if microcode provides a TSX control MSR. If so, system is not vulnerable.
tsx=off	tsx_async_abort=full,nosmt	Ditto
tsx=off	tsx_async_abort=off	ditto

For unaffected platforms "tsx=on" and "tsx_async_abort=full" does not clear CPU buffers. For platforms without TSX control (MSR_IA32_ARCH_CAPABILITIES.MDS_NO=0) "tsx" command line argument has no effect.

For the affected platforms below table indicates the mitigation status for the combinations of CPUID bit MD_CLEAR and IA32_ARCH_CAPABILITIES MSR bits MDS_NO and TSX_CTRL_MSR.

MDS_NO	MD_CLEAR	TSX_CTRL_MSR	Status
0	0	0	Vulnerable (needs microcode)
0	1	0	MDS and TAA mitigated via VERW
1	1	0	MDS fixed, TAA vulnerable if TSX enabled because MD_CLEAR has no meaning and VERW is not guaranteed to clear buffers
1	X	1	MDS fixed, TAA can be mitigated by VERW or TSX_CTRL_MSR

Mitigation selection guide

1. Trusted userspace and guests

If all user space applications are from a trusted source and do not execute untrusted code which is supplied externally, then the mitigation can be disabled. The same applies to virtualized environments with trusted guests.

2. Untrusted userspace and guests

If there are untrusted applications or guests on the system, enabling TSX might allow a malicious actor to leak data from the host or from other processes running on the same physical core.

If the microcode is available and the TSX is disabled on the host, attacks are prevented in a virtualized environment as well, even if the VMs do not explicitly enable the mitigation.

Default mitigations

The kernel's default action for vulnerable processors is:

- Deploy TSX disable mitigation (tsx_async_abort=full tsx=off).