MDS - Microarchitectural Data Sampling

Microarchitectural Data Sampling is a hardware vulnerability which allows unprivileged speculative access to data which is available in various CPU internal buffers.

Affected processors

This vulnerability affects a wide range of Intel processors. The vulnerability is not present on:

- Processors from AMD, Centaur and other non Intel vendors
- Older processor models, where the CPU family is < 6
- Some Atoms (Bonnell, Saltwell, Goldmont, GoldmontPlus)
- Intel processors which have the ARCH_CAP_MDS_NO bit set in the IA32_ARCH_CAPABILITIES MSR.

Whether a processor is affected or not can be read out from the MDS vulnerability file in sysfs. See :ref:`mds_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)mds.rst`, **line 23**); *backlink*
>
> Unknown interpreted text role "ref".

Not all processors are affected by all variants of MDS, but the mitigation is identical for all of them so the kernel treats them as a single vulnerability.

Related CVEs

The following CVE entries are related to the MDS vulnerability:

CVE-2018-12126	MSBDS	Microarchitectural Store Buffer Data Sampling
CVE-2018-12130	MFBDS	Microarchitectural Fill Buffer Data Sampling
CVE-2018-12127	MLPDS	Microarchitectural Load Port Data Sampling
CVE-2019-11091	MDSUM	Microarchitectural Data Sampling Uncacheable Memory

Problem

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

Under certain conditions, usually a fault/assist caused by a load operation, data unrelated to the load memory address can be speculatively forwarded from the buffers. Because the load operation causes a fault or assist and its result will be discarded, the forwarded data will not cause incorrect program execution or state changes. But a malicious operation may be able to forward this speculative data to a disclosure gadget which allows in turn 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.

Deeper technical information is available in the MDS specific x86 architecture section: :ref:`Documentation/x86/mds.rst <mds>`.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\hw-vuln\(linux-master)(Documentation)(admin-guide)(hw-vuln)mds.rst`, **line 60**); *backlink*
>
> Unknown interpreted text role "ref".

Attack scenarios

Attacks against the MDS vulnerabilities can be mounted from malicious non priviledged user space applications running on hosts or guest. Malicious guest OSes can obviously mount attacks as well.

Contrary to other speculation based vulnerabilities the MDS vulnerability does not allow the attacker to control the memory target address. As a consequence the attacks are purely sampling based, but as demonstrated with the TLBleed attack samples can be postprocessed successfully.

Web-Browsers

It's unclear whether attacks through Web-Browsers are possible at all. The exploitation through Java-Script is considered very unlikely, but other widely used web technologies like Webassembly could possibly be abused.

MDS system information

The Linux kernel provides a sysfs interface to enumerate the current MDS status of the system: whether the system is vulnerable, and which mitigations are active. The relevant sysfs file is:

/sys/devices/system/cpu/vulnerabilities/mds

The possible values in this file are:

'Not affected'	The processor is not vulnerable
'Vulnerable'	The processor is vulnerable, but no mitigation enabled
'Vulnerable: Clear CPU buffers attempted, no microcode'	The processor is vulnerable but microcode is not updated. The mitigation is enabled on a best effort basis. See :ref:`vmwerv` **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\hw-vuln\(linux-master)(Documentation)(admin-guide)(hw-vuln)mds.rst, **line 107**); *backlink* Unknown interpreted text role "ref".
'Mitigation: Clear CPU buffers'	The processor is vulnerable and the CPU buffer clearing mitigation is enabled.

If the processor is vulnerable then the following information is appended to the above information:

'SMT vulnerable'	SMT is enabled
'SMT mitigated'	SMT is enabled and mitigated
'SMT disabled'	SMT is disabled
'SMT Host state unknown'	Kernel runs in a VM, Host SMT state unknown

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 mds 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:`mds_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)mds.rst, **line 146**); *backlink*

Unknown interpreted text role "ref".

CPU buffer clearing

The mitigation for MDS clears the affected CPU buffers on return to user space and when entering a guest.

If SMT is enabled it also clears the buffers on idle entry when the CPU is only affected by MSBDS and not any other

MDS variant, because the other variants cannot be protected against cross Hyper-Thread attacks.

For CPUs which are only affected by MSBDS the user space, guest and idle transition mitigations are sufficient and SMT is not affected.

Virtualization mitigation

The protection for host to guest transition depends on the L1TF vulnerability of the CPU:

- CPU is affected by L1TF:

  If the L1D flush mitigation is enabled and up to date microcode is available, the L1D flush mitigation is automatically protecting the guest transition.

  If the L1D flush mitigation is disabled then the MDS mitigation is invoked explicit when the host MDS mitigation is enabled.

  For details on L1TF and virtualization see: :ref:`Documentation/admin-guide/hw-vuln//l1tf.rst <mitigation_control_kvm>`.

  > **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\hw-vuln\(linux-master)(Documentation)(admin-guide)(hw-vuln)mds.rst`, **line 183**); *backlink*
  >
  > Unknown interpreted text role "ref".

- CPU is not affected by L1TF:

  CPU buffers are flushed before entering the guest when the host MDS mitigation is enabled.

The resulting MDS protection matrix for the host to guest transition:

L1TF	MDS	VMX-L1FLUSH	Host MDS	MDS-State
Don't care	No	Don't care	N/A	Not affected
Yes	Yes	Disabled	Off	Vulnerable
Yes	Yes	Disabled	Full	Mitigated
Yes	Yes	Enabled	Don't care	Mitigated
No	Yes	N/A	Off	Vulnerable
No	Yes	N/A	Full	Mitigated

This only covers the host to guest transition, i.e. prevents leakage from host to guest, but does not protect the guest internally. Guests need to have their own protections.

XEON PHI specific considerations

The XEON PHI processor family is affected by MSBDS which can be exploited cross Hyper-Threads when entering idle states. Some XEON PHI variants allow to use MWAIT in user space (Ring 3) which opens an potential attack vector for malicious user space. The exposure can be disabled on the kernel command line with the 'ring3mwait=disable' command line option.

XEON PHI is not affected by the other MDS variants and MSBDS is mitigated before the CPU enters a idle state. As XEON PHI is not affected by L1TF either disabling SMT is not required for full protection.

SMT control

All MDS variants except MSBDS can be attacked cross Hyper-Threads. That means on CPUs which are affected by MFBDS or MLPDS it is necessary to disable SMT for full protection. These are most of the affected CPUs; the exception is XEON PHI, see :ref:`xeon_phi`.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\hw-vuln\(linux-master)(Documentation)(admin-guide)(hw-vuln)mds.rst`, **line 233**); *backlink*
>
> Unknown interpreted text role "ref".

Disabling SMT can have a significant performance impact, but the impact depends on the type of workloads.

See the relevant chapter in the L1TF mitigation documentation for details: :ref:`Documentation/admin-guide/hw-vuln/l1tf.rst <smt_control>`.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\hw-vuln\(linux-master)(Documentation)(admin-guide)(hw-vuln)mds.rst, line 241); *backlink*

Unknown interpreted text role "ref".

Mitigation control on the kernel command line

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

full	If the CPU is vulnerable, enable all available mitigations for the MDS vulnerability, CPU buffer clearing on exit to userspace and when entering a VM. Idle transitions are protected as well if SMT is enabled. It does not automatically disable SMT.
full,nosmt	The same as mds=full, with SMT disabled on vulnerable CPUs. This is the complete mitigation.
off	Disables MDS mitigations completely.

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

Mitigation selection guide

1. Trusted userspace

If all userspace applications are from a trusted source and do not execute untrusted code which is supplied externally, then the mitigation can be disabled.

2. Virtualization with trusted guests

The same considerations as above versus trusted user space apply.

3. Virtualization with untrusted guests

The protection depends on the state of the L1TF mitigations. See :ref:`virt_mechanism`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\hw-vuln\(linux-master)(Documentation)(admin-guide)(hw-vuln)mds.rst, line 293); *backlink*

Unknown interpreted text role "ref".

If the MDS mitigation is enabled and SMT is disabled, guest to host and guest to guest attacks are prevented.

Default mitigations

The kernel default mitigations for vulnerable processors are:

- Enable CPU buffer clearing

The kernel does not by default enforce the disabling of SMT, which leaves SMT systems vulnerable when running untrusted code. The same rationale as for L1TF applies. See :ref:`Documentation/admin-guide/hw-vuln//l1tf.rst <default_mitigations>`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\hw-vuln\(linux-master)(Documentation)(admin-guide)(hw-vuln)mds.rst, line 308); *backlink*

Unknown interpreted text role "ref".