L1TF - L1 Terminal Fault

L1 Terminal Fault is a hardware vulnerability which allows unprivileged speculative access to data which is available in the Level 1 Data Cache when the page table entry controlling the virtual address, which is used for the access, has the Present bit cleared or other reserved bits set.

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
- A range of Intel ATOM processors (Cedarview, Cloverview, Lincroft, Penwell, Pineview, Silvermont, Airmont, Merrifield)
- The Intel XEON PHI family
- Intel processors which have the ARCH_CAP_RDCL_NO bit set in the IA32_ARCH_CAPABILITIES MSR. If the bit is set the CPU is not affected by the Meltdown vulnerability either. These CPUs should become available by end of 2018.

Whether a processor is affected or not can be read out from the L1TF vulnerability file in sysfs. See :ref:`l1tf_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]l1tf.rst, line 29); *backlink*

Unknown interpreted text role "ref".

Related CVEs

The following CVE entries are related to the L1TF vulnerability:

CVE-2018-3615	L1 Terminal Fault	SGX related aspects
CVE-2018-3620	L1 Terminal Fault	OS, SMM related aspects
CVE-2018-3646	L1 Terminal Fault	Virtualization related aspects

Problem

If an instruction accesses a virtual address for which the relevant page table entry (PTE) has the Present bit cleared or other reserved bits set, then speculative execution ignores the invalid PTE and loads the referenced data if it is present in the Level 1 Data Cache, as if the page referenced by the address bits in the PTE was still present and accessible.

While this is a purely speculative mechanism and the instruction will raise a page fault when it is retired eventually, the pure act of loading the data and making it available to other speculative instructions opens up the opportunity for side channel attacks to unprivileged malicious code, similar to the Meltdown attack.

While Meltdown breaks the user space to kernel space protection, L1TF allows to attack any physical memory address in the system and the attack works across all protection domains. It allows an attack of SGX and also works from inside virtual machines because the speculation bypasses the extended page table (EPT) protection mechanism.

Attack scenarios

1. Malicious user space

Operating Systems store arbitrary information in the address bits of a PTE which is marked non present. This allows a malicious user space application to attack the physical memory to which these PTEs resolve. In some cases user-space can maliciously influence the information encoded in the address bits of the PTE, thus making attacks more deterministic and more practical.

The Linux kernel contains a mitigation for this attack vector, PTE inversion, which is permanently enabled and has no performance impact. The kernel ensures that the address bits of PTEs, which are not marked present, never point to cacheable physical memory space.

A system with an up to date kernel is protected against attacks from malicious user space applications.

2. Malicious guest in a virtual machine

The fact that L1TF breaks all domain protections allows malicious guest OSes, which can control the PTEs directly, and malicious guest user space applications, which run on an unprotected guest kernel lacking the PTE inversion mitigation for L1TF, to attack physical host memory.

A special aspect of L1TF in the context of virtualization is symmetric multi threading (SMT). The Intel implementation of SMT is called HyperThreading. The fact that Hyperthreads on the affected processors share the L1 Data Cache (L1D) is important for this. As the flaw allows only to attack data which is present in L1D, a malicious guest running on one Hyperthread can attack the data which is brought into the L1D by the context which runs on the sibling Hyperthread of the same physical core. This context can be host OS, host user space or a different guest.

If the processor does not support Extended Page Tables, the attack is only possible, when the hypervisor does not sanitize the content of the effective (shadow) page tables.

While solutions exist to mitigate these attack vectors fully, these mitigations are not enabled by default in the Linux kernel because they can affect performance significantly. The kernel provides several mechanisms which can be utilized to address the problem depending on the deployment scenario. The mitigations, their protection scope and impact are described in the next sections.

The default mitigations and the rationale for choosing them are explained at the end of this document. See :ref:`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]l1tf.rst`, **line 114**); *backlink*
>
> Unknown interpreted text role "ref".

L1TF system information

The Linux kernel provides a sysfs interface to enumerate the current L1TF status of the system: whether the system is vulnerable, and which mitigations are active. The relevant sysfs file is:

/sys/devices/system/cpu/vulnerabilities/l1tf

The possible values in this file are:

'Not affected'	The processor is not vulnerable
'Mitigation: PTE Inversion'	The host protection is active

If KVM/VMX is enabled and the processor is vulnerable then the following information is appended to the 'Mitigation: PTE Inversion' part:

- SMT status:

'VMX: SMT vulnerable'	SMT is enabled
'VMX: SMT disabled'	SMT is disabled

- L1D Flush mode:

'L1D vulnerable'	L1D flushing is disabled
'L1D conditional cache flushes'	L1D flush is conditionally enabled
'L1D cache flushes'	L1D flush is unconditionally enabled

The resulting grade of protection is discussed in the following sections.

Host mitigation mechanism

The kernel is unconditionally protected against L1TF attacks from malicious user space running on the host.

Guest mitigation mechanisms

1. L1D flush on VMENTER

To make sure that a guest cannot attack data which is present in the L1D the hypervisor flushes the L1D before entering the guest.

Flushing the L1D evicts not only the data which should not be accessed by a potentially malicious guest, it also flushes the guest data. Flushing the L1D has a performance impact as the processor has to bring the flushed guest data back into the L1D. Depending on the frequency of VMEXIT/VMENTER and the type of computations in the guest performance

degradation in the range of 1% to 50% has been observed. For scenarios where guest VMEXIT/VMENTER are rare the performance impact is minimal. Virtio and mechanisms like posted interrupts are designed to confine the VMEXITs to a bare minimum, but specific configurations and application scenarios might still suffer from a high VMEXIT rate.

The kernel provides two L1D flush modes:

- conditional ('cond')
- unconditional ('always')

The conditional mode avoids L1D flushing after VMEXITs which execute only audited code paths before the corresponding VMENTER. These code paths have been verified that they cannot expose secrets or other interesting data to an attacker, but they can leak information about the address space layout of the hypervisor.

Unconditional mode flushes L1D on all VMENTER invocations and provides maximum protection. It has a higher overhead than the conditional mode. The overhead cannot be quantified correctly as it depends on the workload scenario and the resulting number of VMEXITs.

The general recommendation is to enable L1D flush on VMENTER. The kernel defaults to conditional mode on affected processors.

Note, that L1D flush does not prevent the SMT problem because the sibling thread will also bring back its data into the L1D which makes it attackable again.

L1D flush can be controlled by the administrator via the kernel command line and sysfs control files. See :ref:`mitigation_control_command_line` and :ref:`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]l1tf.rst, line 209); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\hw-vuln\[linux-master][Documentation][admin-guide][hw-vuln]l1tf.rst, line 209); *backlink*

Unknown interpreted text role "ref".

2. Guest VCPU confinement to dedicated physical cores

To address the SMT problem, it is possible to make a guest or a group of guests affine to one or more physical cores. The proper mechanism for that is to utilize exclusive cpusets to ensure that no other guest or host tasks can run on these cores.

If only a single guest or related guests run on sibling SMT threads on the same physical core then they can only attack their own memory and restricted parts of the host memory.

Host memory is attackable, when one of the sibling SMT threads runs in host OS (hypervisor) context and the other in guest context. The amount of valuable information from the host OS context depends on the context which the host OS executes, i.e. interrupts, soft interrupts and kernel threads. The amount of valuable data from these contexts cannot be declared as non-interesting for an attacker without deep inspection of the code.

Note, that assigning guests to a fixed set of physical cores affects the ability of the scheduler to do load balancing and might have negative effects on CPU utilization depending on the hosting scenario. Disabling SMT might be a viable alternative for particular scenarios.

For further information about confining guests to a single or to a group of cores consult the cpusets documentation:

https://www.kernel.org/doc/Documentation/admin-guide/cgroup-v1/cpusets.rst

3. Interrupt affinity

Interrupts can be made affine to logical CPUs. This is not universally true because there are types of interrupts which are truly per CPU interrupts, e.g. the local timer interrupt. Aside of that multi queue devices affine their interrupts to single CPUs or groups of CPUs per queue without allowing the administrator to control the affinities.

Moving the interrupts, which can be affinity controlled, away from CPUs which run untrusted guests, reduces the attack vector space.

Whether the interrupts with are affine to CPUs, which run untrusted guests, provide interesting data for an attacker depends on the system configuration and the scenarios which run on the system. While for some of the interrupts it can be assumed that they won't expose interesting information beyond exposing hints about the host OS memory layout, there is no way to make general assumptions.

Interrupt affinity can be controlled by the administrator via the /proc/irq/$NR/smp_affinity[_list] files. Limited

documentation is available at:

https://www.kernel.org/doc/Documentation/core-api/irq/irq-affinity.rst

4. SMT control

To prevent the SMT issues of L1TF it might be necessary to disable SMT completely. Disabling SMT can have a significant performance impact, but the impact depends on the hosting scenario and the type of workloads. The impact of disabling SMT needs also to be weighted against the impact of other mitigation solutions like confining guests to dedicated cores.

The kernel provides a sysfs interface to retrieve the status of SMT and to control it. It also provides a kernel command line interface to control SMT.

The kernel command line interface consists of the following options:

nosmt	Affects the bring up of the secondary CPUs during boot. The kernel tries to bring all present CPUs online during the boot process. "nosmt" makes sure that from each physical core only one - the so called primary (hyper) thread is activated. Due to a design flaw of Intel processors related to Machine Check Exceptions the non primary siblings have to be brought up at least partially and are then shut down again. "nosmt" can be undone via the sysfs interface.
nosmt=force	Has the same effect as "nosmt" but it does not allow to undo the SMT disable via the sysfs interface.

The sysfs interface provides two files:

- /sys/devices/system/cpu/smt/control
- /sys/devices/system/cpu/smt/active

/sys/devices/system/cpu/smt/control:

This file allows to read out the SMT control state and provides the ability to disable or (re)enable SMT. The possible states are:

on	SMT is supported by the CPU and enabled. All logical CPUs can be onlined and offlined without restrictions.
off	SMT is supported by the CPU and disabled. Only the so called primary SMT threads can be onlined and offlined without restrictions. An attempt to online a non-primary sibling is rejected
forceoff	Same as 'off' but the state cannot be controlled. Attempts to write to the control file are rejected.
notsupported	The processor does not support SMT. It's therefore not affected by the SMT implications of L1TF. Attempts to write to the control file are rejected.

The possible states which can be written into this file to control SMT state are:

- on
- off
- forceoff

/sys/devices/system/cpu/smt/active:

This file reports whether SMT is enabled and active, i.e. if on any physical core two or more sibling threads are online.

SMT control is also possible at boot time via the l1tf kernel command line parameter in combination with L1D flush control. See :ref:`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]l1tf.rst`, **line 344**); *backlink*

Unknown interpreted text role "ref".

5. Disabling EPT

Disabling EPT for virtual machines provides full mitigation for L1TF even with SMT enabled, because the effective page tables for guests are managed and sanitized by the hypervisor. Though disabling EPT has a significant performance impact

especially when the Meltdown mitigation KPTI is enabled.

EPT can be disabled in the hypervisor via the 'kvm-intel.ept' parameter.

There is ongoing research and development for new mitigation mechanisms to address the performance impact of disabling SMT or EPT.

Mitigation control on the kernel command line

The kernel command line allows to control the L1TF mitigations at boot time with the option "l1tf=". The valid arguments for this option are:

full	Provides all available mitigations for the L1TF vulnerability. Disables SMT and enables all mitigations in the hypervisors, i.e. unconditional L1D flushing SMT control and L1D flush control via the sysfs interface is still possible after boot. Hypervisors will issue a warning when the first VM is started in a potentially insecure configuration, i.e. SMT enabled or L1D flush disabled.
full,force	Same as 'full', but disables SMT and L1D flush runtime control. Implies the 'nosmt=force' command line option. (i.e. sysfs control of SMT is disabled.)
flush	Leaves SMT enabled and enables the default hypervisor mitigation, i.e. conditional L1D flushing SMT control and L1D flush control via the sysfs interface is still possible after boot. Hypervisors will issue a warning when the first VM is started in a potentially insecure configuration, i.e. SMT enabled or L1D flush disabled.
flush,nosmt	Disables SMT and enables the default hypervisor mitigation, i.e. conditional L1D flushing. SMT control and L1D flush control via the sysfs interface is still possible after boot. Hypervisors will issue a warning when the first VM is started in a potentially insecure configuration, i.e. SMT enabled or L1D flush disabled.
flush,nowarn	Same as 'flush', but hypervisors will not warn when a VM is started in a potentially insecure configuration.
off	Disables hypervisor mitigations and doesn't emit any warnings. It also drops the swap size and available RAM limit restrictions on both hypervisor and bare metal.

The default is 'flush'. For details about L1D flushing see :ref:`l1d_flush`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\hw-vuln\[linux-master][Documentation][admin-guide][hw-vuln]l1tf.rst`, line 413); *backlink*

Unknown interpreted text role "ref".

Mitigation control for KVM - module parameter

The KVM hypervisor mitigation mechanism, flushing the L1D cache when entering a guest, can be controlled with a module parameter.

The option/parameter is "kvm-intel.vmentry_l1d_flush=". It takes the following arguments:

always	L1D cache flush on every VMENTER.
cond	Flush L1D on VMENTER only when the code between VMEXIT and VMENTER can leak host memory which is considered interesting for an attacker. This still can leak host memory which allows e.g. to determine the hosts address space layout.
never	Disables the mitigation

The parameter can be provided on the kernel command line, as a module parameter when loading the modules and at runtime modified via the sysfs file:

/sys/module/kvm_intel/parameters/vmentry_l1d_flush

The default is 'cond'. If 'l1tf=full,force' is given on the kernel command line, then 'always' is enforced and the kvm-intel.vmentry_l1d_flush module parameter is ignored and writes to the sysfs file are rejected.

Mitigation selection guide

1. No virtualization in use

The system is protected by the kernel unconditionally and no further action is required.

2. Virtualization with trusted guests

If the guest comes from a trusted source and the guest OS kernel is guaranteed to have the L1TF mitigations in place the system is fully protected against L1TF and no further action is required.

To avoid the overhead of the default L1D flushing on VMENTER the administrator can disable the flushing via the kernel command line and sysfs control files. See :ref:`mitigation_control_command_line` and :ref:`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]l1tf.rst, line 466); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\hw-vuln\[linux-master][Documentation][admin-guide][hw-vuln]l1tf.rst, line 466); *backlink*

Unknown interpreted text role "ref".

3. Virtualization with untrusted guests

3.1. SMT not supported or disabled

If SMT is not supported by the processor or disabled in the BIOS or by the kernel, it's only required to enforce L1D flushing on VMENTER.

Conditional L1D flushing is the default behaviour and can be tuned. See :ref:`mitigation_control_command_line` and :ref:`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]l1tf.rst, line 481); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\hw-vuln\[linux-master][Documentation][admin-guide][hw-vuln]l1tf.rst, line 481); *backlink*

Unknown interpreted text role "ref".

3.2. EPT not supported or disabled

If EPT is not supported by the processor or disabled in the hypervisor, the system is fully protected. SMT can stay enabled and L1D flushing on VMENTER is not required.

EPT can be disabled in the hypervisor via the 'kvm-intel.ept' parameter.

3.3. SMT and EPT supported and active

If SMT and EPT are supported and active then various degrees of mitigations can be employed:

- L1D flushing on VMENTER:

  L1D flushing on VMENTER is the minimal protection requirement, but it is only potent in combination with other mitigation methods.

  Conditional L1D flushing is the default behaviour and can be tuned. See :ref:`mitigation_control_command_line` and :ref:`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]l1tf.rst, line 504); *backlink*

  Unknown interpreted text role "ref".

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

resources\linux-master\Documentation\admin-guide\hw-vuln\[linux-master] [Documentation] [admin-guide] [hw-vuln]l1tf.rst, line 504); *backlink*

Unknown interpreted text role "ref".

- Guest confinement:

  Confinement of guests to a single or a group of physical cores which are not running any other processes, can reduce the attack surface significantly, but interrupts, soft interrupts and kernel threads can still expose valuable data to a potential attacker. See :ref:`guest_confinement`.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\hw-vuln\[linux-master] [Documentation] [admin-guide] [hw-vuln]l1tf.rst, line 509); *backlink*

  Unknown interpreted text role "ref".

- Interrupt isolation:

  Isolating the guest CPUs from interrupts can reduce the attack surface further, but still allows a malicious guest to explore a limited amount of host physical memory. This can at least be used to gain knowledge about the host address space layout. The interrupts which have a fixed affinity to the CPUs which run the untrusted guests can depending on the scenario still trigger soft interrupts and schedule kernel threads which might expose valuable information. See :ref:`interrupt_isolation`.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\hw-vuln\[linux-master] [Documentation] [admin-guide] [hw-vuln]l1tf.rst, line 517); *backlink*

  Unknown interpreted text role "ref".

The above three mitigation methods combined can provide protection to a certain degree, but the risk of the remaining attack surface has to be carefully analyzed. For full protection the following methods are available:

- Disabling SMT:

  Disabling SMT and enforcing the L1D flushing provides the maximum amount of protection. This mitigation is not depending on any of the above mitigation methods.

  SMT control and L1D flushing can be tuned by the command line parameters 'nosmt', 'l1tf', 'kvm-intel.vmentry_l1d_flush' and at run time with the matching sysfs control files. See :ref:`smt_control`, :ref:`mitigation_control_command_line` and :ref:`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]l1tf.rst, line 537); *backlink*

  Unknown interpreted text role "ref".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\hw-vuln\[linux-master] [Documentation] [admin-guide] [hw-vuln]l1tf.rst, line 537); *backlink*

  Unknown interpreted text role "ref".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\hw-vuln\[linux-master] [Documentation] [admin-guide] [hw-vuln]l1tf.rst, line 537); *backlink*

  Unknown interpreted text role "ref".

- Disabling EPT:

  Disabling EPT provides the maximum amount of protection as well. It is not depending on any of the above mitigation methods. SMT can stay enabled and L1D flushing is not required, but the performance impact is significant.

  EPT can be disabled in the hypervisor via the 'kvm-intel.ept' parameter.

3.4. Nested virtual machines

When nested virtualization is in use, three operating systems are involved: the bare metal hypervisor, the nested hypervisor and the nested virtual machine. VMENTER operations from the nested hypervisor into the nested guest will always be processed by the bare metal hypervisor. If KVM is the bare metal hypervisor it will:

- Flush the L1D cache on every switch from the nested hypervisor to the nested virtual machine, so that the nested hypervisor's secrets are not exposed to the nested virtual machine;
- Flush the L1D cache on every switch from the nested virtual machine to the nested hypervisor; this is a complex operation, and flushing the L1D cache avoids that the bare metal hypervisor's secrets are exposed to the nested virtual machine;
- Instruct the nested hypervisor to not perform any L1D cache flush. This is an optimization to avoid double L1D flushing.

Default mitigations

The kernel default mitigations for vulnerable processors are:

- PTE inversion to protect against malicious user space. This is done unconditionally and cannot be controlled. The swap storage is limited to ~16TB.
- L1D conditional flushing on VMENTER when EPT is enabled for a guest.

The kernel does not by default enforce the disabling of SMT, which leaves SMT systems vulnerable when running untrusted guests with EPT enabled.

The rationale for this choice is:

- Force disabling SMT can break existing setups, especially with unattended updates.
- If regular users run untrusted guests on their machine, then L1TF is just an add on to other malware which might be embedded in an untrusted guest, e.g. spam-bots or attacks on the local network.

  There is no technical way to prevent a user from running untrusted code on their machines blindly.
- It's technically extremely unlikely and from today's knowledge even impossible that L1TF can be exploited via the most popular attack mechanisms like JavaScript because these mechanisms have no way to control PTEs. If this would be possible and not other mitigation would be possible, then the default might be different.
- The administrators of cloud and hosting setups have to carefully analyze the risk for their scenarios and make the appropriate mitigation choices, which might even vary across their deployed machines and also result in other changes of their overall setup. There is no way for the kernel to provide a sensible default for this kind of scenarios.