Spectre Side Channels

Spectre is a class of side channel attacks that exploit branch prediction and speculative execution on modern CPUs to read memory, possibly bypassing access controls. Speculative execution side channel exploits do not modify memory but attempt to infer privileged data in the memory.

This document covers Spectre variant 1 and Spectre variant 2.

Affected processors

Speculative execution side channel methods affect a wide range of modern high performance processors, since most modern high speed processors use branch prediction and speculative execution.

The following CPUs are vulnerable:

- Intel Core, Atom, Pentium, and Xeon processors
- AMD Phenom, EPYC, and Zen processors
- IBM POWER and zSeries processors
- Higher end ARM processors
- Apple CPUs
- Higher end MIPS CPUs
- Likely most other high performance CPUs. Contact your CPU vendor for details.

Whether a processor is affected or not can be read out from the Spectre vulnerability files in sysfs. See :ref:`spectre_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)spectre.rst`, line 36); *backlink*
>
> Unknown interpreted text role "ref".

Related CVEs

The following CVE entries describe Spectre variants:

CVE-2017-5753	Bounds check bypass	Spectre variant 1
CVE-2017-5715	Branch target injection	Spectre variant 2
CVE-2019-1125	Spectre v1 swapgs	Spectre variant 1 (swapgs)

Problem

CPUs use speculative operations to improve performance. That may leave traces of memory accesses or computations in the processor's caches, buffers, and branch predictors. Malicious software may be able to influence the speculative execution paths, and then use the side effects of the speculative execution in the CPUs' caches and buffers to infer privileged data touched during the speculative execution.

Spectre variant 1 attacks take advantage of speculative execution of conditional branches, while Spectre variant 2 attacks use speculative execution of indirect branches to leak privileged memory. See :ref:`[1] <spec_ref1>` :ref:`[5] <spec_ref5>` :ref:`[6] <spec_ref6>` :ref:`[7] <spec_ref7>` :ref:`[10] <spec_ref10>` :ref:`[11] <spec_ref11>`.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\hw-vuln\(linux-master) (Documentation) (admin-guide) (hw-vuln)spectre.rst`, line 60); *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)spectre.rst`, line 60); *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)spectre.rst, line 60); *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)spectre.rst`, line 60); *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)spectre.rst`, line 60); *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)spectre.rst`, line 60); *backlink*

Unknown interpreted text role "ref".

Spectre variant 1 (Bounds Check Bypass)

The bounds check bypass attack :ref:`[2] <spec_ref2>` takes advantage of speculative execution that bypasses conditional branch instructions used for memory access bounds check (e.g. checking if the index of an array results in memory access within a valid range). This results in memory accesses to invalid memory (with out-of-bound index) that are done speculatively before validation checks resolve. Such speculative memory accesses can leave side effects, creating side channels which leak information to the attacker.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\hw-vuln\(linux-master)(Documentation)(admin-guide)(hw-vuln)spectre.rst`, line 69); *backlink*

Unknown interpreted text role "ref".

There are some extensions of Spectre variant 1 attacks for reading data over the network, see :ref:`[12] <spec_ref12>`. However such attacks are difficult, low bandwidth, fragile, and are considered low risk.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\hw-vuln\(linux-master)(Documentation)(admin-guide)(hw-vuln)spectre.rst`, line 78); *backlink*

Unknown interpreted text role "ref".

Note that, despite "Bounds Check Bypass" name, Spectre variant 1 is not only about user-controlled array bounds checks. It can affect any conditional checks. The kernel entry code interrupt, exception, and NMI handlers all have conditional swapgs checks. Those may be problematic in the context of Spectre v1, as kernel code can speculatively run with a user GS.

Spectre variant 2 (Branch Target Injection)

The branch target injection attack takes advantage of speculative execution of indirect branches :ref:`[3] <spec_ref3>`. The indirect branch predictors inside the processor used to guess the target of indirect branches can be influenced by an attacker, causing gadget code to be speculatively executed, thus exposing sensitive data touched by the victim. The side effects left in the CPU's caches during speculative execution can be measured to infer data values.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\hw-vuln\(linux-master)(Documentation)(admin-guide)(hw-vuln)spectre.rst`, line 92); *backlink*

Unknown interpreted text role "ref".

In Spectre variant 2 attacks, the attacker can steer speculative indirect branches in the victim to gadget code by poisoning the branch target buffer of a CPU used for predicting indirect branch addresses. Such poisoning could be done by indirect branching into existing code, with the address offset of the indirect branch under the attacker's control. Since the branch prediction on impacted hardware does not fully disambiguate branch address and uses the offset for prediction, this could cause privileged code's indirect branch to jump to a gadget code with the same offset.

The most useful gadgets take an attacker-controlled input parameter (such as a register value) so that the memory read can be controlled. Gadgets without input parameters might be possible, but the attacker would have very little control over what memory can be read, reducing the risk of the attack revealing useful data.

One other variant 2 attack vector is for the attacker to poison the return stack buffer (RSB) :ref:`[13] <spec_ref13>` to cause speculative subroutine return instruction execution to go to a gadget. An attacker's imbalanced subroutine call instructions might "poison" entries in the return stack buffer which are later consumed by a victim's subroutine return instructions. This attack can be mitigated by flushing the return stack buffer on context switch, or virtual machine (VM) exit.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\hw-vuln\(linux-master) (Documentation) (admin-guide) (hw-vuln)spectre.rst`, line 118); *backlink*

Unknown interpreted text role "ref".

On systems with simultaneous multi-threading (SMT), attacks are possible from the sibling thread, as level 1 cache and branch target buffer (BTB) may be shared between hardware threads in a CPU core. A malicious program running on the sibling thread may influence its peer's BTB to steer its indirect branch speculations to gadget code, and measure the speculative execution's side effects left in level 1 cache to infer the victim's data.

Yet another variant 2 attack vector is for the attacker to poison the Branch History Buffer (BHB) to speculatively steer an indirect branch to a specific Branch Target Buffer (BTB) entry, even if the entry isn't associated with the source address of the indirect branch. Specifically, the BHB might be shared across privilege levels even in the presence of Enhanced IBRS.

Currently the only known real-world BHB attack vector is via unprivileged eBPF. Therefore, it's highly recommended to not enable unprivileged eBPF, especially when eIBRS is used (without retpolines). For a full mitigation against BHB attacks, it's recommended to use retpolines (or eIBRS combined with retpolines).

Attack scenarios

The following list of attack scenarios have been anticipated, but may not cover all possible attack vectors.

1. A user process attacking the kernel

Spectre variant 1

The attacker passes a parameter to the kernel via a register or via a known address in memory during a syscall. Such parameter may be used later by the kernel as an index to an array or to derive a pointer for a Spectre variant 1 attack. The index or pointer is invalid, but bound checks are bypassed in the code branch taken for speculative execution. This could cause privileged memory to be accessed and leaked.

For kernel code that has been identified where data pointers could potentially be influenced for Spectre attacks, new "nospec" accessor macros are used to prevent speculative loading of data.

Spectre variant 1 (swapgs)

An attacker can train the branch predictor to speculatively skip the swapgs path for an interrupt or exception. If they initialize the GS register to a user-space value, if the swapgs is speculatively skipped, subsequent GS-related percpu accesses in the speculation window will be done with the attacker-controlled GS value. This could cause privileged memory to be accessed and leaked.

For example:

```
if (coming from user space)
    swapgs
mov %gs:<percpu_offset>, %reg
mov (%reg), %reg1
```

When coming from user space, the CPU can speculatively skip the swapgs, and then do a speculative percpu load using the user GS value. So the user can speculatively force a read of any kernel value. If a gadget exists which uses the percpu value as an address in another load/store, then the contents of the kernel value may become visible via an L1 side channel attack.

A similar attack exists when coming from kernel space. The CPU can speculatively do the swapgs, causing the user GS to get used for the rest of the speculative window.

Spectre variant 2

A spectre variant 2 attacker can :ref:`poison <poison_btb>` the branch target buffer (BTB) before issuing syscall to launch an attack. After entering the kernel, the kernel could use the poisoned branch target buffer on indirect jump and jump to gadget code in speculative execution.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\hw-vuln\(linux-master) (Documentation) (admin-guide) (hw-vuln) spectre.rst`, line 204); *backlink*
>
> Unknown interpreted text role "ref".

If an attacker tries to control the memory addresses leaked during speculative execution, he would also need to pass a parameter to the gadget, either through a register or a known address in memory. After the gadget has executed, he can measure the side effect.

The kernel can protect itself against consuming poisoned branch target buffer entries by using return trampolines (also known as "retpoline") :ref:`[3] <spec_ref3>` :ref:`[9] <spec_ref9>` for all indirect branches. Return trampolines trap speculative execution paths to prevent jumping to gadget code during speculative execution. x86 CPUs with Enhanced Indirect Branch Restricted Speculation (Enhanced IBRS) available in hardware should use the feature to mitigate Spectre variant 2 instead of retpoline. Enhanced IBRS is more efficient than retpoline.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\hw-vuln\(linux-master) (Documentation) (admin-guide) (hw-vuln) spectre.rst`, line 215); *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) spectre.rst`, line 215); *backlink*
>
> Unknown interpreted text role "ref".

There may be gadget code in firmware which could be exploited with Spectre variant 2 attack by a rogue user process. To mitigate such attacks on x86, Indirect Branch Restricted Speculation (IBRS) feature is turned on before the kernel invokes any firmware code.

2. A user process attacking another user process

A malicious user process can try to attack another user process, either via a context switch on the same hardware thread, or from the sibling hyperthread sharing a physical processor core on simultaneous multi-threading (SMT) system.

Spectre variant 1 attacks generally require passing parameters between the processes, which needs a data passing relationship, such as remote procedure calls (RPC). Those parameters are used in gadget code to derive invalid data pointers accessing privileged memory in the attacked process.

Spectre variant 2 attacks can be launched from a rogue process by :ref:`poisoning <poison_btb>` the branch target buffer. This can influence the indirect branch targets for a victim process that either runs later on the same hardware thread, or running concurrently on a sibling hardware thread sharing the same physical core.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\hw-vuln\(linux-master) (Documentation) (admin-guide) (hw-vuln) spectre.rst`, line 244); *backlink*
>
> Unknown interpreted text role "ref".

A user process can protect itself against Spectre variant 2 attacks by using the prctl() syscall to disable indirect branch speculation for itself. An administrator can also cordon off an unsafe process from polluting the branch target buffer by disabling the process's indirect branch speculation. This comes with a performance cost from not using indirect branch speculation and clearing the branch target buffer. When SMT is enabled on x86, for a process that has indirect branch speculation disabled, Single Threaded Indirect Branch Predictors (STIBP) :ref:`[4] <spec_ref4>` are turned on to prevent the sibling thread from controlling branch target buffer. In addition, the Indirect Branch Prediction Barrier (IBPB) is issued to clear the branch target buffer when context switching to and from such process.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\hw-vuln\(linux-master) (Documentation) (admin-guide) (hw-vuln) spectre.rst`, line 250); *backlink*

Unknown interpreted text role "ref".

On x86, the return stack buffer is stuffed on context switch. This prevents the branch target buffer from being used for branch prediction when the return stack buffer underflows while switching to a deeper call stack. Any poisoned entries in the return stack buffer left by the previous process will also be cleared.

User programs should use address space randomization to make attacks more difficult (Set /proc/sys/kernel/randomize_va_space = 1 or 2).

3. A virtualized guest attacking the host

The attack mechanism is similar to how user processes attack the kernel. The kernel is entered via hyper-calls or other virtualization exit paths.

For Spectre variant 1 attacks, rogue guests can pass parameters (e.g. in registers) via hyper-calls to derive invalid pointers to speculate into privileged memory after entering the kernel. For places where such kernel code has been identified, nospec accessor macros are used to stop speculative memory access.

For Spectre variant 2 attacks, rogue guests can :ref:`poison <poison_btb>` the branch target buffer or return stack buffer, causing the kernel to jump to gadget code in the speculative execution paths.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\hw-vuln\(linux-master)(Documentation)(admin-guide)(hw-vuln)spectre.rst`, line 285); *backlink*

Unknown interpreted text role "ref".

To mitigate variant 2, the host kernel can use return trampolines for indirect branches to bypass the poisoned branch target buffer, and flushing the return stack buffer on VM exit. This prevents rogue guests from affecting indirect branching in the host kernel.

To protect host processes from rogue guests, host processes can have indirect branch speculation disabled via prctl(). The branch target buffer is cleared before context switching to such processes.

4. A virtualized guest attacking other guest

A rogue guest may attack another guest to get data accessible by the other guest.

Spectre variant 1 attacks are possible if parameters can be passed between guests. This may be done via mechanisms such as shared memory or message passing. Such parameters could be used to derive data pointers to privileged data in guest. The privileged data could be accessed by gadget code in the victim's speculation paths.

Spectre variant 2 attacks can be launched from a rogue guest by :ref:`poisoning <poison_btb>` the branch target buffer or the return stack buffer. Such poisoned entries could be used to influence speculation execution paths in the victim guest.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\hw-vuln\(linux-master)(Documentation)(admin-guide)(hw-vuln)spectre.rst`, line 310); *backlink*

Unknown interpreted text role "ref".

Linux kernel mitigates attacks to other guests running in the same CPU hardware thread by flushing the return stack buffer on VM exit, and clearing the branch target buffer before switching to a new guest.

If SMT is used, Spectre variant 2 attacks from an untrusted guest in the sibling hyperthread can be mitigated by the administrator, by turning off the unsafe guest's indirect branch speculation via prctl(). A guest can also protect itself by turning on microcode based mitigations (such as IBPB or STIBP on x86) within the guest.

Spectre system information

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

The sysfs file showing Spectre variant 1 mitigation status is:

/sys/devices/system/cpu/vulnerabilities/spectre_v1

The possible values in this file are:

'Not affected'	The processor is not vulnerable.

'Vulnerable: __user pointer sanitization and usercopy barriers only; no swapgs barriers'	The swapgs protections are disabled; otherwise it has protection in the kernel on a case by case base with explicit pointer sanitation and usercopy LFENCE barriers.
'Mitigation: usercopy/swapgs barriers and __user pointer sanitization'	Protection in the kernel on a case by case base with explicit pointer sanitation, usercopy LFENCE barriers, and swapgs LFENCE barriers.

However, the protections are put in place on a case by case basis, and there is no guarantee that all possible attack vectors for Spectre variant 1 are covered.

The spectre_v2 kernel file reports if the kernel has been compiled with retpoline mitigation or if the CPU has hardware mitigation, and if the CPU has support for additional process-specific mitigation.

This file also reports CPU features enabled by microcode to mitigate attack between user processes:

1. Indirect Branch Prediction Barrier (IBPB) to add additional isolation between processes of different users.
2. Single Thread Indirect Branch Predictors (STIBP) to add additional isolation between CPU threads running on the same core.

These CPU features may impact performance when used and can be enabled per process on a case-by-case base.

The sysfs file showing Spectre variant 2 mitigation status is:

```
/sys/devices/system/cpu/vulnerabilities/spectre_v2
```

The possible values in this file are:

- Kernel status:

'Not affected'	The processor is not vulnerable
'Mitigation: None'	Vulnerable, no mitigation
'Mitigation: Retpolines'	Use Retpoline thunks
'Mitigation: LFENCE'	Use LFENCE instructions
'Mitigation: Enhanced IBRS'	Hardware-focused mitigation
'Mitigation: Enhanced IBRS + Retpolines'	Hardware-focused + Retpolines
'Mitigation: Enhanced IBRS + LFENCE'	Hardware-focused + LFENCE

- Firmware status: Show if Indirect Branch Restricted Speculation (IBRS) is used to protect against Spectre variant 2 attacks when calling firmware (x86 only).

'IBRS_FW'	Protection against user program attacks when calling firmware

- Indirect branch prediction barrier (IBPB) status for protection between processes of different users. This feature can be controlled through prctl() per process, or through kernel command line options. This is an x86 only feature. For more details see below.

'IBPB: disabled'	IBPB unused
'IBPB: always-on'	Use IBPB on all tasks
'IBPB: conditional'	Use IBPB on SECCOMP or indirect branch restricted tasks

- Single threaded indirect branch prediction (STIBP) status for protection between different hyper threads. This feature can be controlled through prctl per process, or through kernel command line options. This is x86 only feature. For more details see below.

'STIBP: disabled'	STIBP unused
'STIBP: forced'	Use STIBP on all tasks
'STIBP: conditional'	Use STIBP on SECCOMP or indirect branch restricted tasks

- Return stack buffer (RSB) protection status:

'RSB filling'	Protection of RSB on context switch enabled

Full mitigation might require a microcode update from the CPU vendor. When the necessary microcode is not available, the kernel will report vulnerability.

Turning on mitigation for Spectre variant 1 and Spectre variant 2

1. Kernel mitigation

Spectre variant 1

For the Spectre variant 1, vulnerable kernel code (as determined by code audit or scanning tools) is annotated on a case by case basis to use nospec accessor macros for bounds clipping :ref:`[2] <spec_ref2>` to avoid any usable disclosure gadgets. However, it may not cover all attack vectors for Spectre variant 1.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\hw-vuln\(linux-master) (Documentation) (admin-guide) (hw-vuln)spectre.rst`, **line 438**); ***backlink***

Unknown interpreted text role "ref".

Copy-from-user code has an LFENCE barrier to prevent the access_ok() check from being mis-speculated. The barrier is done by the barrier_nospec() macro.

For the swapgs variant of Spectre variant 1, LFENCE barriers are added to interrupt, exception and NMI entry where needed. These barriers are done by the FENCE_SWAPGS_KERNEL_ENTRY and FENCE_SWAPGS_USER_ENTRY macros.

Spectre variant 2

For Spectre variant 2 mitigation, the compiler turns indirect calls or jumps in the kernel into equivalent return trampolines (retpolines) :ref:`[3] <spec_ref3>` :ref:`[9] <spec_ref9>` to go to the target addresses. Speculative execution paths under retpolines are trapped in an infinite loop to prevent any speculative execution jumping to a gadget.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\hw-vuln\(linux-master) (Documentation) (admin-guide) (hw-vuln)spectre.rst`, **line 456**); ***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)spectre.rst`, **line 456**); ***backlink***

Unknown interpreted text role "ref".

To turn on retpoline mitigation on a vulnerable CPU, the kernel needs to be compiled with a gcc compiler that supports the -mindirect-branch=thunk-extern -mindirect-branch-register options. If the kernel is compiled with a Clang compiler, the compiler needs to support -mretpoline-external-thunk option. The kernel config CONFIG_RETPOLINE needs to be turned on, and the CPU needs to run with the latest updated microcode.

On Intel Skylake-era systems the mitigation covers most, but not all, cases. See :ref:`[3] <spec_ref3>` for more details.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\hw-vuln\(linux-master) (Documentation) (admin-guide) (hw-vuln)spectre.rst`, **line 471**); ***backlink***

Unknown interpreted text role "ref".

On CPUs with hardware mitigation for Spectre variant 2 (e.g. Enhanced IBRS on x86), retpoline is automatically disabled at run time.

The retpoline mitigation is turned on by default on vulnerable CPUs. It can be forced on or off by the administrator via the kernel command line and sysfs control files. See :ref:`spectre_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)spectre.rst`, **line 477**); ***backlink***

Unknown interpreted text role "ref".

On x86, indirect branch restricted speculation is turned on by default before invoking any firmware code to prevent Spectre variant 2 exploits using the firmware.

Using kernel address space randomization (CONFIG_RANDOMIZE_BASE=y and CONFIG_SLAB_FREELIST_RANDOM=y in the kernel configuration) makes attacks on the kernel generally more difficult.

2. User program mitigation

User programs can mitigate Spectre variant 1 using LFENCE or "bounds clipping". For more details see :ref:`[2] <spec_ref2>`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\hw-vuln\(linux-master) (Documentation) (admin-guide) (hw-vuln) spectre.rst`, **line 493**); ***backlink***

Unknown interpreted text role "ref".

For Spectre variant 2 mitigation, individual user programs can be compiled with return trampolines for indirect branches. This protects them from consuming poisoned entries in the branch target buffer left by malicious software. Alternatively, the programs can disable their indirect branch speculation via prctl() (See :ref:`Documentation/userspace-api/spec_ctrl.rst <set_spec_ctrl>`). On x86, this will turn on STIBP to guard against attacks from the sibling thread when the user program is running, and use IBPB to flush the branch target buffer when switching to/from the program.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\hw-vuln\(linux-master) (Documentation) (admin-guide) (hw-vuln) spectre.rst`, **line 496**); ***backlink***

Unknown interpreted text role "ref".

Restricting indirect branch speculation on a user program will also prevent the program from launching a variant 2 attack on x86. Administrators can change that behavior via the kernel command line and sysfs control files. See :ref:`spectre_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) spectre.rst`, **line 506**); ***backlink***

Unknown interpreted text role "ref".

Programs that disable their indirect branch speculation will have more overhead and run slower.

User programs should use address space randomization (/proc/sys/kernel/randomize_va_space = 1 or 2) to make attacks more difficult.

3. VM mitigation

Within the kernel, Spectre variant 1 attacks from rogue guests are mitigated on a case by case basis in VM exit paths. Vulnerable code uses nospec accessor macros for "bounds clipping", to avoid any usable disclosure gadgets. However, this may not cover all variant 1 attack vectors.

For Spectre variant 2 attacks from rogue guests to the kernel, the Linux kernel uses retpoline or Enhanced IBRS to prevent consumption of poisoned entries in branch target buffer left by rogue guests. It also flushes the return stack buffer on every VM exit to prevent a return stack buffer underflow so poisoned branch target buffer could be used, or attacker guests leaving poisoned entries in the return stack buffer.

To mitigate guest-to-guest attacks in the same CPU hardware thread, the branch target buffer is sanitized by flushing before switching to a new guest on a CPU.

The above mitigations are turned on by default on vulnerable CPUs.

To mitigate guest-to-guest attacks from sibling thread when SMT is in use, an untrusted guest running in the sibling thread can have its indirect branch speculation disabled by administrator via prctl().

The kernel also allows guests to use any microcode based mitigation they choose to use (such as IBPB or STIBP on x86) to protect themselves.

Mitigation control on the kernel command line

Spectre variant 2 mitigation can be disabled or force enabled at the kernel command line.

nospectre_v1

[X86,PPC] Disable mitigations for Spectre Variant 1 (bounds check bypass). With this option data leaks are possible in the system.

nospectre_v2

[X86] Disable all mitigations for the Spectre variant 2 (indirect branch prediction) vulnerability. System may allow data leaks with this option, which is equivalent to spectre_v2=off.

spectre_v2=

[X86] Control mitigation of Spectre variant 2 (indirect branch speculation) vulnerability. The default operation protects the kernel from user space attacks.

on
unconditionally enable, implies spectre_v2_user=on

off
unconditionally disable, implies spectre_v2_user=off

auto
kernel detects whether your CPU model is vulnerable

Selecting 'on' will, and 'auto' may, choose a mitigation method at run time according to the CPU, the available microcode, the setting of the CONFIG_RETPOLINE configuration option, and the compiler with which the kernel was built.

Selecting 'on' will also enable the mitigation against user space to user space task attacks.

Selecting 'off' will disable both the kernel and the user space protections.

Specific mitigations can also be selected manually:

retpoline auto pick between generic,lfence retpoline,generic Retpolines retpoline,lfence LFENCE; indirect branch retpoline,amd alias for retpoline,lfence eibrs enhanced IBRS eibrs,retpoline enhanced IBRS + Retpolines eibrs,lfence enhanced IBRS + LFENCE

Not specifying this option is equivalent to spectre_v2=auto.

In general the kernel by default selects reasonable mitigations for the current CPU. To disable Spectre variant 2 mitigations, boot with spectre_v2=off. Spectre variant 1 mitigations cannot be disabled.

For spectre_v2_user see Documentation/admin-guide/kernel-parameters.txt

Mitigation selection guide

1. Trusted userspace

If all userspace applications are from trusted sources and do not execute externally supplied untrusted code, then the mitigations can be disabled.

2. Protect sensitive programs

For security-sensitive programs that have secrets (e.g. crypto keys), protection against Spectre variant 2 can be put in place by disabling indirect branch speculation when the program is running (See :ref:`Documentation/userspace-api/spec_ctrl.rst <set_spec_ctrl>`).

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\hw-vuln\(linux-master) (Documentation) (admin-guide) (hw-vuln) spectre.rst`, line 633); *backlink*

Unknown interpreted text role "ref".

3. Sandbox untrusted programs

Untrusted programs that could be a source of attacks can be cordoned off by disabling their indirect branch speculation when they are run (See :ref:`Documentation/userspace-api/spec_ctrl.rst <set_spec_ctrl>`). This prevents untrusted programs from polluting the branch target buffer. This behavior can be changed via the kernel command line and sysfs control files. See :ref:`spectre_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) spectre.rst`, line 641); *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) spectre.rst`, line 641); *backlink*

Unknown interpreted text role "ref".

3. High security mode

All Spectre variant 2 mitigations can be forced on at boot time for all programs (See the "on" option in :ref:`spectre_mitigation_control_command_line`). This will add overhead as indirect branch speculations for all programs will be restricted.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\hw-vuln\(linux-master) (Documentation) (admin-guide) (hw-vuln) spectre.rst`, line 652); ***backlink***

Unknown interpreted text role "ref".

On x86, branch target buffer will be flushed with IBPB when switching to a new program. STIBP is left on all the time to protect programs against variant 2 attacks originating from programs running on sibling threads.

Alternatively, STIBP can be used only when running programs whose indirect branch speculation is explicitly disabled, while IBPB is still used all the time when switching to a new program to clear the branch target buffer (See "ibpb" option in :ref:`spectre_mitigation_control_command_line`). This "ibpb" option has less performance cost than the "on" option, which leaves STIBP on all the time.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\hw-vuln\(linux-master) (Documentation) (admin-guide) (hw-vuln) spectre.rst`, line 663); ***backlink***

Unknown interpreted text role "ref".

References on Spectre

Intel white papers:

[1] Intel analysis of speculative execution side channels.

[2] Bounds check bypass.

[3] Deep dive: Retpoline: A branch target injection mitigation.

[4] Deep Dive: Single Thread Indirect Branch Predictors.

AMD white papers:

[5] AMD64 technology indirect branch control extension.

[6] Software techniques for managing speculation on AMD processors.

ARM white papers:

[7] Cache speculation side-channels.

[8] Cache speculation issues update.

Google white paper:

[9] Retpoline: a software construct for preventing branch-target-injection.

MIPS white paper:

[10] MIPS: response on speculative execution and side channel vulnerabilities.

Academic papers:

[11] Spectre Attacks: Exploiting Speculative Execution.

[12] NetSpectre: Read Arbitrary Memory over Network.

[13] Spectre Returns! Speculation Attacks using the Return Stack Buffer.