Speculation Control

Quite some CPUs have speculation-related misfeatures which are in fact vulnerabilities causing data leaks in various forms even across privilege domains.

The kernel provides mitigation for such vulnerabilities in various forms. Some of these mitigations are compile-time configurable and some can be supplied on the kernel command line.

There is also a class of mitigations which are very expensive, but they can be restricted to a certain set of processes or tasks in controlled environments. The mechanism to control these mitigations is via :manpage:`prctl(2)`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\(linux-master) (Documentation) (userspace-api) spec_ctrl.rst`, **line 13)**; ***backlink***

Unknown interpreted text role "manpage".

There are two prctl options which are related to this:

- PR_GET_SPECULATION_CTRL
- PR_SET_SPECULATION_CTRL

PR_GET_SPECULATION_CTRL

PR_GET_SPECULATION_CTRL returns the state of the speculation misfeature which is selected with arg2 of prctl(2). The return value uses bits 0-3 with the following meaning:

Bit	Define	Description
0	PR_SPEC_PRCTL	Mitigation can be controlled per task by PR_SET_SPECULATION_CTRL.
1	PR_SPEC_ENABLE	The speculation feature is enabled, mitigation is disabled.
2	PR_SPEC_DISABLE	The speculation feature is disabled, mitigation is enabled.
3	PR_SPEC_FORCE_DISABLE	Same as PR_SPEC_DISABLE, but cannot be undone. A subsequent prctl(..., PR_SPEC_ENABLE) will fail.
4	PR_SPEC_DISABLE_NOEXEC	Same as PR_SPEC_DISABLE, but the state will be cleared on :manpage:`execve(2)`. **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\(linux-master) (Documentation) (userspace-api) spec_ctrl.rst`, **line 43)**; ***backlink*** Unknown interpreted text role "manpage".

If all bits are 0 the CPU is not affected by the speculation misfeature.

If PR_SPEC_PRCTL is set, then the per-task control of the mitigation is available. If not set, prctl(PR_SET_SPECULATION_CTRL) for the speculation misfeature will fail.

PR_SET_SPECULATION_CTRL

PR_SET_SPECULATION_CTRL allows to control the speculation misfeature, which is selected by arg2 of :manpage:`prctl(2)` per task. arg3 is used to hand in the control value, i.e. either PR_SPEC_ENABLE or PR_SPEC_DISABLE or PR_SPEC_FORCE_DISABLE.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\(linux-master) (Documentation) (userspace-api) spec_ctrl.rst`, **line 57)**; ***backlink***

Unknown interpreted text role "manpage".

Common error codes

Value	Meaning
EINVAL	The prctl is not implemented by the architecture or unused prctl(2) arguments are not 0.

Value	Meaning
ENODEV	arg2 is selecting a not supported speculation misfeature.

PR_SET_SPECULATION_CTRL error codes

Value	Meaning
0	Success
ERANGE	arg3 is incorrect, i.e. it's neither PR_SPEC_ENABLE nor PR_SPEC_DISABLE nor PR_SPEC_FORCE_DISABLE.
ENXIO	Control of the selected speculation misfeature is not possible. See PR_GET_SPECULATION_CTRL.
EPERM	Speculation was disabled with PR_SPEC_FORCE_DISABLE and caller tried to enable it again.

Speculation misfeature controls

- PR_SPEC_STORE_BYPASS: Speculative Store Bypass

  Invocations:
    - prctl(PR_GET_SPECULATION_CTRL, PR_SPEC_STORE_BYPASS, 0, 0, 0);
    - prctl(PR_SET_SPECULATION_CTRL, PR_SPEC_STORE_BYPASS, PR_SPEC_ENABLE, 0, 0);
    - prctl(PR_SET_SPECULATION_CTRL, PR_SPEC_STORE_BYPASS, PR_SPEC_DISABLE, 0, 0);
    - prctl(PR_SET_SPECULATION_CTRL, PR_SPEC_STORE_BYPASS, PR_SPEC_FORCE_DISABLE, 0, 0);
    - prctl(PR_SET_SPECULATION_CTRL, PR_SPEC_STORE_BYPASS, PR_SPEC_DISABLE_NOEXEC, 0, 0);
- PR_SPEC_INDIR_BRANCH: Indirect Branch Speculation in User Processes
  (Mitigate Spectre V2 style attacks against user processes)

  Invocations:
    - prctl(PR_GET_SPECULATION_CTRL, PR_SPEC_INDIRECT_BRANCH, 0, 0, 0);
    - prctl(PR_SET_SPECULATION_CTRL, PR_SPEC_INDIRECT_BRANCH, PR_SPEC_ENABLE, 0, 0);
    - prctl(PR_SET_SPECULATION_CTRL, PR_SPEC_INDIRECT_BRANCH, PR_SPEC_DISABLE, 0, 0);
    - prctl(PR_SET_SPECULATION_CTRL, PR_SPEC_INDIRECT_BRANCH, PR_SPEC_FORCE_DISABLE, 0, 0);
- PR_SPEC_L1D_FLUSH: Flush L1D Cache on context switch out of the task
  (works only when tasks run on non SMT cores)

  Invocations:
    - prctl(PR_GET_SPECULATION_CTRL, PR_SPEC_L1D_FLUSH, 0, 0, 0);
    - prctl(PR_SET_SPECULATION_CTRL, PR_SPEC_L1D_FLUSH, PR_SPEC_ENABLE, 0, 0);
    - prctl(PR_SET_SPECULATION_CTRL, PR_SPEC_L1D_FLUSH, PR_SPEC_DISABLE, 0, 0);