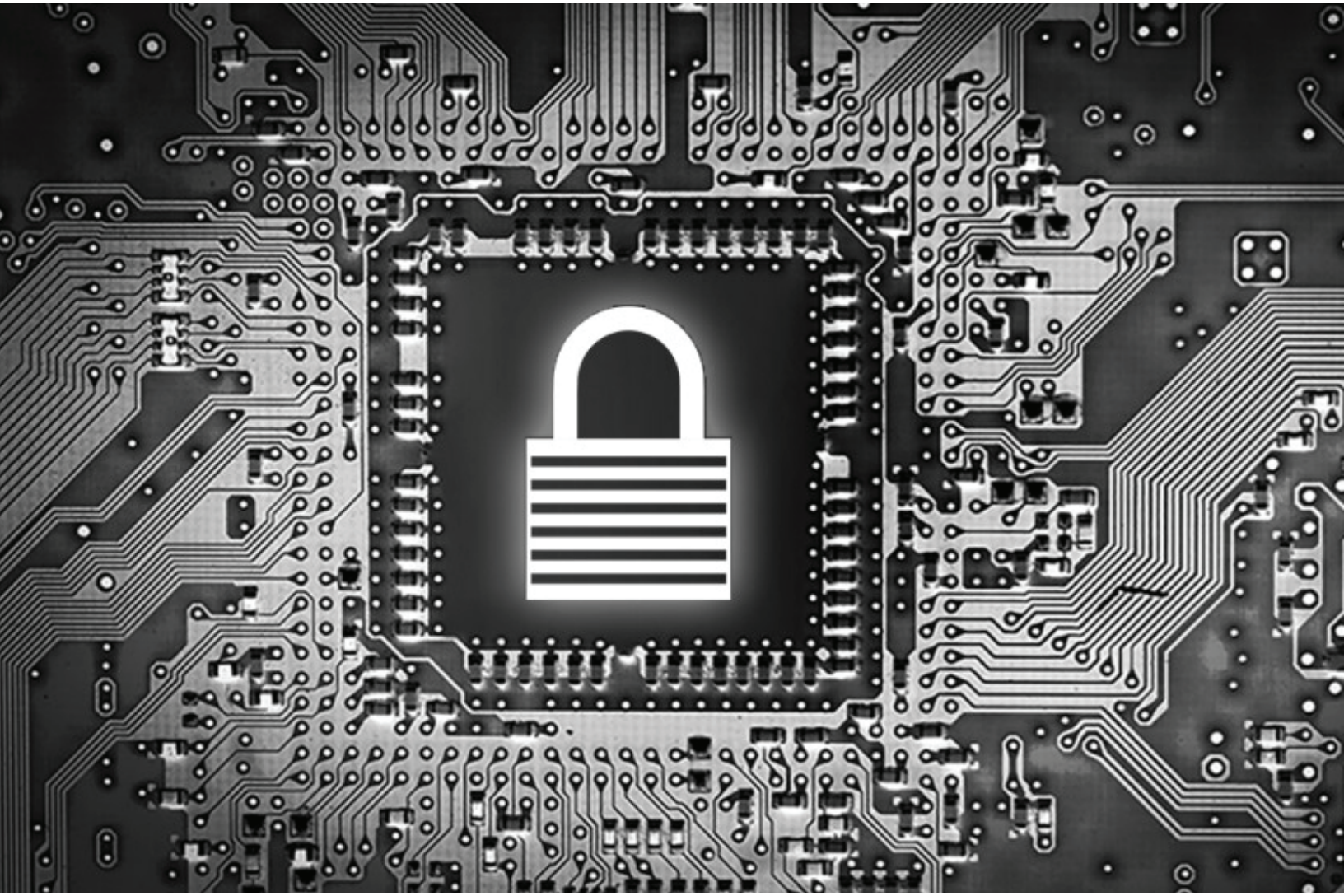
AMD



White Paper | AMD64 TECHNOLOGY

INDIRECT BRANCH CONTROL EXTENSION

REVISION 7.10.18

AMD64 TECHNOLOGY INDIRECT BRANCH CONTROL EXTENSION

This document describes an indirect branch control feature designed to mitigate indirect branch target injection on AMD products. There are three defined mechanisms: Indirect Branch Prediction Barrier (IBPB), Indirect Branch Restricted Speculation (IBRS), and Single Thread Indirect Branch Prediction mode (STIBP).

PRESENCE

The presence of the three features are indicated through three CPUID bits. AMD does enumerate these features differently than other x86 vendors. IBPB support is indicated by CPUID Function 8000_0008, EBX[12]=1. IBRS support is indicated by CPUID Function 8000_0008, EBX[14]=1. STIBP support is indicated by CPUID Function 8000_0008, EBX[15]=1. Support for IBPB implies that MSR 0x49 exists in the architecture. Support for IBRS or STIBP implies MSR 0x48 exists. Here is a simple representation in table form:

FEATURE	AMD VERSION	MSR EXIST
IBPB	8000_0008 EBX[12]=1	PRED_CMD (MSR 49)
IBRS	8000_0008 EBX[14]=1	SPEC_CTRL (MSR 48)
STIBP	8000_0008 EBX[15]=1	SPEC_CTRL (MSR 48)

USAGE

The indirect branch control extension is essentially three features that allow for increased control of indirect branches.

IBPB

Indirect branch prediction barrier (IBPB) exists at MSR 0x49 (PRED_CMD) bit 0. This is a write only MSR that both GP faults when software reads it or if software tries to write any of the bits in 63:1. When bit zero is written, the processor guarantees that older indirect branches cannot influence predictions of indirect branches in the future. This applies to jmp indirects, call indirects and returns. As this restricts the processor from using all previous indirect branch information, it is intended to only be used by software when switching from one user context to another user context that requires protection, or from one guest to another guest.

- **IBPB is the AMD recommended setting for Windows mitigation of Google Project Zero Variant 2 (Spectre).**
- **IBPB combined with Reptoline software support is the AMD recommended setting for Linux mitigation of Google Project Zero Variant 2 (Spectre).**

IBRS

Indirect Branch Restricted Speculation (IBRS) exists at MSR 0x48 (SPEC_CTRL) bit 0. When this bit is set, it keeps indirect branches that occurred in a lesser prediction mode from before it was set from influencing the future indirect branches that are going to execute now while IBRS is 1. A lesser prediction mode is CPL 3 vs CPL[2-0] and Guest vs Host mode. If software clears IBRS, it is now allowed for the older indirect branches that occurred when IBRS was 0 to be used to influence the indirect branches. It is also possible that while IBRS is 1, another write of 1 to IBRS bit 0 occurs. This starts a new window where older indirect branches should not influence future indirect branches. Therefore, if IBRS were set in a lesser privilege mode, on a transition to a more privileged mode the more privileged mode would have to set IBRS to 1 to indicate to hardware that it wants branches in the more privileged

mode separated from those in the lesser privileged mode with IBRS set. On processors with a shared indirect branch predictor, IBRS being set provides protection from being influenced by a sibling thread's indirect branch predictions. For the ret type of indirect branch, software is responsible for clearing out the return stack buffer with 32 calls that have a non-zero target. Processors that support more than 32 RSB entries will be responsible for clearing the extra RSB entries. Clearing out the return stack buffer maybe required on the transition from CPL3 to CPL0, even if the OS has SMEP enabled.

AMD is not recommending IBRS currently as a performant mitigation in Windows for Google Project Zero Variant 2 (Spectre).

STIBP

Single Thread Indirect Branch Predictor (STIBP) exists at MSR 0x48 (SPEC_CTRL) bit 1. When this bit is set in processors that share branch prediction information, indirect branch predictions from sibling threads cannot influence the predictions of other sibling threads. Return instructions are always immune to influence by the other thread and do not require this bit to be set for protection.

AMD is not recommending STIBP currently as a performant mitigation in Windows and Linux for Google Project Zero Variant 2 (Spectre).

Any attempt to write SPEC_CTRL bits 63:2 results in GP fault. If a processor only supports STIBP (bit 1) for ease of software implementation the processor does not GP fault attempts to write bit 0. In a similar manner, if a processor only supports IBRS, attempts to set STIBP do not GP fault.

Both SPEC_CTRL and PRED_CMD are not architecturally serializing WRMSRs. They are still execution serializing and prevent any execution of future instructions until they have completed.

EXTENDED USAGE MODELS

Different AMD processors have different implementations of the 3 features. Extra CPUID enumeration has been added to help provide more information on how the processor operates to help optimize performance. Software should check that the base feature exists before interpreting the additional feature bits.

CPUID Function 8000_0008, EBX[16]=1 indicates an IBRS always on mode. This indicates that the processor prefers that IBRS is only set once during boot and not changed. If IBRS is set on a processor supporting IBRS always on mode, indirect branches executed in a less privileged prediction mode will not influence branch predictions for indirect branches in a more privileged prediction mode. This also reduces the performance impact of the WRMSR on less privileged to more privileged entry point and the WRMSR on more privileged to less privileged exit points.

CPUID Function 8000_0008, EBX[17]=1 indicates an STIBP always on mode. This indicates that the processor prefers that STIBP is only set once during boot and not changed. This reduces the performance impact of the WRMSR at the necessary toggle points.

CPUID Function 8000_0008, EBX[18]=1 indicates that the processor prefers using the IBRS feature instead of other software mitigations such as retpoline. This allows software to remove the software mitigation and utilize the more performant IBRS mechanism.

These extended performance CPUID bits give software some insight into the performance impact of the basic functions of IBC on the processor. These bits are just recommendations and overall performance will vary based on the privileged software's capability to separate trusted software from untrusted software.

HYPERVISOR USAGE MODELS

Since the frequency of updates to SPEC_CTRL may be high in guest OSes, intercepting them could be a performance problem. Since a hypervisor may need to control IBRS and STIBP based on how it is scheduling the guests, both a host and guest version of SPEC_CTRL exists in the hardware. When in host mode, the host SPEC_CTRL value is controlling the hardware and all writes only update the host version of SPEC_CTRL. On a VMRUN the processor loads the guest version of SPEC_CTRL from the VMCB and processor behavior is controlled by the logical OR of the two registers. Therefore, if the hypervisor needs to enforce protection with IBRS and/or STIBP on, it can set them in the host version and keep the guest safe. When the guest writes SPEC_CTRL to turn on protection, the guest version updates and if the host version is zero, turns the protection on. On a VMEXIT, the Guest version is saved into the VMCB and the processor returns to only using the host SPEC_CTRL for protection. The Guest SPEC_CTRL is located at offset 0x2E0 in the VMCB. CPUID indication of this feature is in Fn 8000_000A, EDX[20]=1.

SMM

If the SMM handler is using the appropriate secure environment of ASEG, TSEG, and SMM lock then the SMM handler is already restricted and cannot speculate into other areas of the memory system. Therefore, it is not required the SPEC_CTL be modified on entry or exit into SMM and it will retain its current value on entry.

REVISION 7.10.18

AMD.com/en/corporate/speculative-execution

AMD