Complex Shadow-Stack Updates (Intel® Control-Flow Enforcement Technology)

Document No: 356628-001US
August 2023

Notices and Disclaimers

Notice: This document contains information on products in the design phase of development. The information here is subject to change without notice. Do not finalize a design with this information.

Performance varies by use, configuration, and other factors. Learn more on the Performance Index site.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See backup for configuration details. No product or component can be absolutely secure.

Your costs and results may vary.

Performance varies by use, configuration, and other factors. Learn more on the Performance Index site.

Intel technologies may require enabled hardware, software or service activation.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document. The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm.

Revision History

Revision	Description	Date
356628-001US	Initial Release.	August 2023

Table of Contents

Contents

1. DOCUMENT OVERVIEW

Intel's Control-Flow Enforcement Technology (CET) introduces the concept of a shadow stack. When configured to do so, the CPU uses shadow stacks to ensure the correctness of certain control-flow transfers.

Some control-flow transfers update a shadow stack with multiple accesses, including modification of a stack element called a supervisor shadow-stack token. Such shadow-stack updates are said to be complex. Typically, complex shadow-stack updates pertain only to the shadow stacks used by supervisor software (software operating when the Current Privilege Level — CPL — is 0)[1] .

Certain events encountered during a complex shadow-stack update (including those leading to a VM exit from a virtual machine) can cause the shadow stack to become prematurely busy, meaning that the update fails to complete but has marked the shadow stack "busy." A prematurely busy shadow stack may lead to unexpected behavior.

This paper presents a mitigation that virtual-machine monitors (VMMs) can use to prevent the unexpected behaviors resulting from prematurely busy shadow stacks from affecting their virtual machines. This mitigation is based on new CPU support planned by Intel. The paper also makes recommendations to developers of operating systems so that they can avoid enabling supervisor shadow stacks when supported by a VMM that has not yet implemented the mitigation.

Please contact your Intel representative with any questions.

[1] Complex shadow-stack updates can apply to user shadow stacks, but only if an operating system has configured the interrupt-descriptor table (IDT) to deliver some events with CPL = 3. Intel is not aware of any operating system that does this. For that reason, this paper focuses on supervisor shadow stacks.

2. HIGH LEVEL SUMMARY

Intel's Control-Flow Enforcement Technology (CET) introduces the concept of a shadow stack. When configured to do so, the CPU uses shadow stacks to ensure the correctness of certain control-flow transfers.

Some control-flow transfers update a shadow stack with multiple accesses, including modification of a stack element called a **supervisor shadow-stack token** (henceforth **token**). A shadow stack's token includes a bit that indicates whether the shadow stack is in use or **busy**. Such shadow-stack updates are said to be **complex**. Typically, complex shadow-stack updates pertain only to the shadow stacks used by supervisor software (software operating when the Current Privilege Level — CPL — is 0).

Certain events encountered during a complex shadow-stack update can cause the shadow stack to become **prematurely busy**, meaning that the update fails to complete but has marked the shadow stack "busy." Prematurely busy shadow stacks may lead to unexpected behavior.

This paper explains how a shadow stack may become prematurely busy. It makes recommendations to developers of operating systems and virtual-machine monitors regarding ways that they can avoid the unexpected behaviors that may result from shadow stacks becoming prematurely busy.

This paper is structured as follows:

- Section **3**: Describes complex shadow-stack updates and when they occur.
- Section 4: Explains how a complex shadow-stack update may cause a shadow stack to become prematurely busy.
- Section 5: Example of unexpected behavior that may result.
- Section 6: Describes mitigations, including software updates that customers should implement now and hardware enhancements that will be available in the future.
- Appendix A: Provides additional architectural background.

Note: When Flexible Return and Event Delivery (FRED) transitions are enabled, control-flow transfers do not use supervisor shadow-stack tokens. The issues described in this paper do not apply when FRED transitions are enabled.

3. BACKGROUND: COMPLEX SHADOW-STACK UPDATES

This section explains the nature of complex shadow-stack updates and the circumstances under which they occur.

A complex shadow-stack update occurs when a control-flow transfer switches to a new shadow stack and then pushes values onto that new shadow stack.

Any of the following control transfers switch to a new shadow stack:

- Delivery of an event using the Interrupt Descriptor Table (IDT) or an execution of far CALL that changes the CPL.
- IDT event delivery that uses the Interrupt-Stack Table (IST).
- A task switch.

These transitions load the Shadow-Stack Pointer (SSP) with a new value, which is the address of the bottom (highest address) of the new shadow stack.

At the bottom of each supervisor shadow stack is a special 8-byte value called a supervisor shadow-stack token (henceforth token). The lowest bit of a token is its busy bit.

When a control-flow transfer switches to a new shadow stack, the CPU loads the SSP with the address of the new shadow stack's token and then reads the token. If the token's busy bit is 1, a general-protection fault (#GP) occurs (this is an error condition). If instead the busy bit is 0, the CPU sets the busy bit to 1 and completes the control-flow transfer. The write to set the busy bit is atomic with the read of the token.

Any of the following control transfers pushes three 8-byte values on the shadow stack:

- IDT event delivery or an execution of far CALL that begins with CPL < 3.
- IDT event delivery or an execution of far CALL that ends with CPL = 3.
- Certain task switches.

If such a control-flow transfer does not switch shadow stacks, these pushes are to the current shadow stack. If such a control-flow transfer does switch stacks, the pushes are to the new shadow stack, and they are performed only after setting the new shadow stack's token's busy bit as described earlier.

The combination of setting a token's busy bit and pushing values on a new shadow stack is called a **complex shadow-stack update**. Any of the following control-flow transfers will include a complex shadow-stack update:

- IDT event delivery or an execution of far CALL that begins with CPL < 3 and which changes the CPL. (This implies that the CPL had been either 1 or 2.)
- IDT event delivery that uses the Interrupt-Stack Table (IST) and that begins with CPL < 3.
- IDT event delivery that uses the IST and that ends with CPL = 3. (This implies that the IDT had been configured to deliver some event to a handler operating with CPL = 3.)
- A task switch.

For most 64-bit operating systems, a complex shadow-stack update occurs only as part of IDT event delivery that commences with CPL = 0 and that uses the IST.

4. ROOT CAUSE SUMMARY: PREMATURELY BUSY SHADOW STACK

The previous section defined complex shadow-stack updates as first setting a token's busy bit and then pushing three 8-byte values to the shadow stack.

Certain events may prevent execution of a memory access performed by a complex shadow-stack update. These events include page faults, machine checks[2], and (when software is operating in a virtual machine) EPT violations, EPT misconfigurations[3], page-modification log-full events, events related to Sub-Page Permissions (SPP), and instruction-timeout events. These events may manifest as direct exceptions (#PF, #MC), double faults (#DF), virtualization exceptions (#VE)[4], or VM exits[5]. A complex shadow-stack update results in a **prematurely busy** shadow stack if it sets the busy bit in a token and one of its pushes fails due to one of the events identified earlier[6].

Most operations that fail to complete due to one of these events do not update memory in a way that would affect the operation if it were re-executed. Complex shadow-stack updates are different: once it is set, the token's busy bit remains set and may prevent re-execution of the update (see Section 5: IMPACT: UNEXPECTED BEHAVIOR RESULTING FROM PREMATURELY BUSY SHADOW STACKS).

[2] A machine check that causes a shadow stack to become prematurely busy is not restartable.

[3] EPT violations and EPT misconfigurations are events that may occur when Extended Page Tables (EPT) are in use.

[4] A #VE results from conversion of an EPT violation into an exception. Such conversion does not occur during IDT event delivery. Thus, a complex shadow-stack update that makes a shadow stack prematurely busy can lead to a #VE only during an execution of the CALL instruction that begins when the CPL is either 1 or 2.

[5] Because task switches cannot occur in VMX non-root operation, a complex shadow-stack update that is part of a task switch is subject only to page faults and machine checks.

[6] The shadow-stack pushes are the last memory accesses performed by event delivery. Event delivery necessarily completes if all three shadow-stack pushes complete.

5. IMPACT: UNEXPECTED BEHAVIOR RESULTING FROM PREMATURELY BUSY SHADOW STACKS

This section gives an example of unexpected behavior resulting from a prematurely busy shadow stack.

Suppose that some event E is being delivered in a virtual machine (VM) operating with CPL = 0 and suppose that guest OS has configured the IDT so that delivery of E uses the IST.

Since delivery of event E uses the IST, it switches shadow stacks. The CPU reads the token at the bottom of the new shadow stack, confirms that its busy bit is clear, and sets the busy bit. Since the CPL was 0 before event E occurred, the CPU then attempts to push data on the new shadow stack. Suppose that one of the shadow-stack pushes cause an EPT violation. The EPT violation prevents the complex shadow-stack update from completing and the new shadow stack is prematurely busy.

The EPT violation will cause a VM exit to the VMM. The VMM will resolve the EPT violation (for example, by mapping a guest-physical address that had been unmapped) and then resume the guest using VM entry. Because the VM exit occurred during delivery of event E, the VMM will inject E into the guest as part of VM entry. At the end of VM entry, the CPU will attempt to deliver E to the guest OS.

Once again, the CPU will find that event E is configured to use the IST and thus to switch shadow stacks. The CPU will read the token at the bottom of the new shadow stack to confirm that its busy bit is clear. Because of the earlier execution of the complex shadow-stack update, the busy bit is already set.

When IDT event delivery finds the busy bit in the shadow stack's token already set to 1, it will cease delivery of event E and cause a general-protection fault (#GP). The guest OS will not expect this fault and will likely panic and shut down.

The guest OS cannot reasonably avoid this situation because it has no control over which guest-physical addresses might cause EPT violations; that is under control of the VMM.

The VMM also cannot easily avoid this situation because it does not know which guest-physical addresses must be mapped to avoid it. Even if the VMM takes care to ensure that all guest shadow-stack pages are mapped using EPT, an EPT violation can still cause the new shadow stack to become prematurely busy if it occurs when attempting to access a guest paging structure to translate the linear address of a shadow-stack page.

6. PLANNED MITIGATIONS

This section provides details of mitigations of the technical issues described earlier. It includes the following:

1. Steps that an OS (or other system software) must take to avoid the unexpected behavior resulting from prematurely busy shadow stacks. These appear in Section 6.1 Immediate Actions Required: Operating Systems.
2. Steps that a VMM must take to support each guest OS in this avoidance. These appear in Section 6.2 Immediate Actions Required: Virtual-Machine Monitors.
3. New CPU support planned by Intel, including a new feature to report to a VMM when a VM exit causes a shadow stack to become prematurely busy. This is detailed in Section 6.4 Future Mitigation: VMM Support.
4. A workaround that a VMM can implement to free a prematurely busy shadow stack and so prevent unexpected behavior. This is explained in Section 6.4 Future Mitigation: VMM Support.

The general idea is that a VMM can use the CPU support in #3 and the software workaround in #4 to mitigate the effects of prematurely busy shadow stacks that might otherwise affect its VMs. The software steps in #1 and #2 are designed to ensure that an OS running in a VM will enable supervisor shadow stacks when the mitigation is in place but will not do so when it is not.

The mitigations are based on a new feature flag enumerated by the CPUID instruction: CPUID.(EAX=07H,ECX=1H):EDX[bit 18]. At the time of this publication, all CPUs enumerate this bit as 0, but Intel expects in the future to update affected CPUs to enumerate this bit as 1. VMMs should virtualize the CPUID instruction to enumerate this bit as 1 only if they implement the workaround in Section 6.4 Future Mitigation: VMM Support

6.1 Immediate Actions Required: Operating Systems

Note: Although this section uses the term “OS,” the required actions here apply to any system software that might choose to enable supervisor shadow stacks. This includes OSes, VMMs, and BIOSes.

Note: This section does not apply to OSes that use FRED transitions. An OS that enables FRED transitions is not subject to prematurely busy shadow stacks and need not take the steps specified in this section.

This section details the steps that an OS must take to ensure that it is not subject to unexpected behavior resulting from prematurely busy shadow stacks. Specifically, it explains what an OS must do before enabling supervisor shadow stacks.

Before enabling supervisor shadow stacks, an OS must ensure that complex shadow-stack updates cannot result in a prematurely busy shadow stack. Recall that this may occur due to an event resulting from actions taken by the OS itself or (if the OS is running in a VM) by a VMM.

To avoid causing a shadow stack to become prematurely busy due to a page fault, an OS that enables supervisor shadow stacks should configure paging to ensure that such a page fault cannot occur. It can do this by properly mapping the base address of any shadow stack to which IDT event delivery might switch. It is expected that operating systems already configure paging in this way because to do otherwise risks the loss of interrupts and other events.

In addition, an OS should check the value of CPUID.(EAX=07H,ECX=1H):EDX[bit 18]. If that bit is enumerated as 0, the OS should not enable supervisor shadow stacks. If the bit is enumerated as 1, the OS can enable supervisor shadow stacks if it chooses. The following items explain:

- If the OS is not operating in a VM, an execution of CPUID returns a value established by the CPU. As mentioned earlier, Intel CPUs will return the value 1 for CPUID.(EAX=07H,ECX=1H):EDX[bit 18], meaning that the OS may enable supervisor shadow stacks. It can do so without risk because for such an OS a shadow stack cannot become prematurely busy due to an event resulting from actions taken by a VMM.
- If the OS is operating in a VM, executions of CPUID are virtualized and appear to return values established by the VMM. The VMM should virtualize CPUID.(EAX=07H,ECX=1H):EDX[bit 18] to return 1 only if it has implemented the workaround described in Section 6.4: Future Mitigation: VMM Support. This means that the OS will enable supervisor shadow-stacks only if the VMM has implemented the workaround.

If an OS knows through other means that its supervisor shadow stacks cannot become prematurely busy, it could choose to enable supervisor shadow stacks even if CPUID.(EAX=07H,ECX=1H):EDX[bit 18] is enumerated as 0.

6.2 Immediate Actions Required: Virtual-Machine Monitors

Guest software operating in a virtual machine cannot execute the CPUID instruction directly. All guest executions of CPUID cause VM exits and must be emulated by the VMM, which has complete control over the values returned.

CPUID.(EAX=07H,ECX=1H):EDX[bit 18] is a new feature flag that indicates a shadow stack will not become prematurely busy due to an event under control of a VMM. A VMM should enumerate this bit as 0 unless it implements the workaround in Section 6.4: Future Mitigation: VMM Support.

6.3 Future Mitigation: CPU Support

This section begins by describing an enhancement to the Virtual Machine Extensions (VMX) planned by Intel that allows a VMM to determine when a VM exit may have caused a shadow stack to become prematurely busy and that gives the VMM the information necessary to mitigate the consequences. The feature is called here prematurely busy shadow-stack reporting. A CPU enumerates support for prematurely busy shadow-stack reporting by setting bit 3 in the IA32_VMX_EXIT_CTLS2 MSR.

Intel will release a functional update in IPU 23.3. Public disclosure is planned for August 08, 2023.

A VMM enables prematurely busy shadow-stack reporting by setting bit 3 of the secondary VM-exit controls field in the Virtual-Machine Control Structure (VMCS). (This control is called “prematurely busy shadow stack.”) Software can write this field using the VMWRITE instruction and the field encoding 00002044H.

If prematurely busy shadow-stack reporting is enabled, a VM exit due to an event that caused a shadow stack to become prematurely busy may report this fact by setting bit 25 of the exit-reason field in the VMCS. Software can read the value of this field using the VMREAD instruction and the field encoding 00004402H.

VM exits due to the following reasons set this bit if they cause a shadow stack to become prematurely busy:

- EPT violations.
- EPT misconfigurations.
- SPP-related events.
- Page-modification log-full events.
- Instruction-timeout events.

(A shadow stack may become prematurely busy due to an exception that then causes a VM exit. The possible exceptions are page fault, double fault, and machine check. Such a VM exit will not set bit 25 of the exit-reason field. Note that a page fault or double fault will cause a shadow stack to become prematurely busy only if the guest OS has configured paging to allow this to occur. A machine check that causes a shadow stack to become prematurely busy is not recoverable. In none of these cases is it necessary for the VMM to take remedial action.)

VM exits due to EPT violations and SPP-related events always provide the Guest-Linear Address (GLA) of the access that led to the VM exit. When prematurely busy shadow-stack reporting is enabled, other VM exits that set bit 25 of the exit reason also report this GLA:

- EPT misconfigurations.

If prematurely busy shadow-stack reporting is enabled and an EPT misconfiguration causes a shadow stack to become prematurely busy, the resulting VM exit saves the GLA of shadow-stack access that caused the EPT misconfiguration. The GLA is undefined in other cases.

- Page-modification log-full events.

If prematurely busy shadow-stack reporting is enabled and a page-modification log-full event causes a shadow stack to become prematurely busy, the resulting VM exit saves the GLA of the shadow-stack access that caused the page-modification log-full event. The GLA is undefined in other cases.

- Instruction-timeout events.

If prematurely busy shadow-stack reporting is enabled and an instruction-timeout event causes a shadow stack to become prematurely busy, the event will occur during one of the shadow-stack accesses; if the exit qualification of the resulting VM exit does not set bit 0, the VM exit saves the GLA of that shadow-stack access. The GLA is undefined in other cases.

For all these cases, software can read the saved GLA using the VMREAD instruction and the field encoding 0000640AH.

As discussed earlier, an EPT violation normally results in a VM exit to the VMM. However, a VMM's configuration may specify that certain EPT violations be delivered to the guest OS as a virtualization exception (#VE).

Complex shadow-stack updates occur only during IDT event delivery or during execution of the far CALL instruction. An EPT violation that occurs during IDT event delivery always causes a VM exit and is never converted to a #VE.

Intel CPUs that support prematurely busy shadow-stack reporting will also ensure that any EPT violation that causes a shadow stack to become prematurely busy will cause a VM exit (and not be converted to a #VE), even an EPT violation during execution of far CALL.

6.4 Future Mitigation: VMM Support

This section describes a workaround that a VMM can implement to mitigate the effects of prematurely busy shadow stacks and thus allow guest software to enable supervisor shadow stacks. The workaround uses prematurely busy shadow-stack reporting defined in Section 6.3: Future Mitigation: CPU Support.

The workaround prevents the unexpected behavior that may result from a shadow stack made prematurely busy by an event under the VMM's control and which led to a VM exit. It does so by undoing the update of a token's busy bit.

To enable the workaround, a VMM should set the "prematurely busy shadow stack" VM-exit control. The VMM can then apply the following steps when handling a VM exit due to each of the following reasons: EPT violations, EPT misconfigurations, page-modification log-full events, or SPP-related events. These steps can be performed before or after normal handling of the VM exit:

1. Consult the exit reason field and check bit 25. If the bit is 1 (indicating that the VM exit resulted from an event that caused a shadow stack to become prematurely busy), continue to the next step. (If the bit is 0, only the normal VM-exit handling is required.)
2. Compute the GLA of the token in which the busy bit was set. This can be done by taking the GLA provided by the VM exit and setting bits 4:0 of the address to the value 18H.
3. Determine the physical address of the token.
    a. First, translate the GLA identified in the previous step to a GPA using the guest paging structures.
    b. Then, use the GPA to identify a physical address. (This could be done using the EPT paging structures or other VMM-managed data structures).
    c. Note that translating the GLA to a GPA may require the VMM to access guest paging structures whose GPAs are not mapped with EPT. (In fact, it is possible that the EPT violation that caused a shadow stack to become prematurely busy was due to an unmapped guest paging structure.)
4. Identify a mapping that the VMM can use to write to the token. The VMM may already have such a mapping, or it could create a temporary one for this purpose.
5. Clear the busy bit in the token using a locked compare-and-exchange instruction (e.g., an 8-byte form of CMPXCHG) with the mapping identified in the previous step, performed so as to modify the value of the token from GLA|1 to GLA. (A valid token contains its own linear address.) The instruction will read memory at the specified linear address and confirm that it contains a valid token with its busy bit set. If it does, the instruction will write back a valid token with its busy bit clear.

Performing these steps undoes the busy-bit setting of the complex shadow-stack update that occurred prior to the VM exit. The VMM can then resume the guest as it would normally. (If the VM exit occurred during event delivery, the subsequent VM entry would inject the original event.) Unless another VM exit occurs, the complex shadow-stack update should then complete in its entirety.

The following items regard other VM exits resulting from events that caused a shadow stack to become prematurely busy:

A VM exit due to a page fault (#PF)

Such a VM exit will not set bit 25 of the exit-reason field. A #PF that causes a shadow stack to become prematurely busy was due to the paging configuration established by guest software. For this reason, the VMM can resume the VM without further action. In this case, the guest OS is not following the recommendations in Section 6.1: Immediate Actions Required: Operating Systems and should be prepared for prematurely busy shadow stacks.

A VM exit due to a machine check (#MC)

Such a VM exit will not set bit 25 of the exit-reason field. As noted earlier, a machine check that causes a shadow stack to become prematurely busy is not recoverable and the VMM should not resume the VM.

A VM exit due to a double fault (#DF)

Such a VM exit will not set bit 25 of the exit-reason field. This can occur if a shadow-stack push encounters a #PF that does not cause a VM exit and subsequent delivery of the #PF encounters another exception.

As noted earlier, a #PF that causes a shadow stack to become prematurely busy was due to the paging configuration established by guest software. This implies that the guest OS is not following the recommendations in Immediate Actions Required: Operating Systems and should be prepared for prematurely busy shadow stacks. The VMM can resume the guest (perhaps injecting the #DF) without further action.

A VM exit due to an instruction-timeout event.

Such a VM exit will set bit 25 of the exit-reason field. If bit 0 of the exit qualification is 0, the VM exit saves the GLA of one of the shadow-stack pushes. The VMM can follow steps 1–5 above using this GLA.

If bit 0 of the exit qualification is 1, the VM context in the VMCS is not reliable. The VMM should not resume the VM.

A VMM that implements the workaround described above should indicate that fact to guest software by virtualizing CPUID.(EAX=07H,ECX=1H):EDX[bit 18] to return 1.

A VMM that does not implement the workaround (perhaps because it is running on a CPU that does not support prematurely busy shadow-stack reporting) should enumerate CPUID.(EAX=07H,ECX=1H):EDX[bit 18] as 0.

A VMM may support nested virtualization by virtualizing the CPU's Virtual Machine Extensions (VMX) to software operating in a VM. In virtualizing VMX, such a VMM should enumerate and virtualize the features identified in Section 6.3. This would allow a "guest VMM" operating in the VM to also implement the workaround described in this section.

Appendix A. ARCHITECTURE BACKGROUND

The original architecture for shadow stacks was documented in Revision 1.0 of *Control-flow Enforcement Technology Preview* (Document Number 334525-001, June 2016) and was later integrated into the *Intel® 64 and IA-32 Architectures Software Developer's Manual (*starting with Revision 71 in October 2019).

For complex shadow-stack updates, the original architecture required that the new SSP value be 8-byte aligned, but it made no other alignment requirements. This value could be at the base of a page (for example, 80004000H), meaning that the token could be on a different page than the three values subsequently pushed on the new shadow stack (these would be at addresses 80003FF8H, 80003FF0H, and 80003FE8H). If the token is on a different page than the pushed data, the shadow stack could become prematurely busy if the page with the token was mapped, and the other page was not.

Revision 74 of the *Intel® 64 and IA-32 Architectures Software Developer's Manual* (June 2021) changed the specification of complex shadow-stack updates to indicate that the CPU ensures that the new SSP value is such that the token and the three 8-byte pushes are all contained in an aligned 32-byte region. This is done by checking that low 5 bits of the new SSP have value 18H. (If they do not, a #GP occurs.)

Intel supports this change in all implementations of the shadow-stack feature (possibly by a microcode update).