TECHNICAL GUIDANCE FOR MITIGATING TRANSIENT SCHEDULER ATTACKS

REVISION 2.0 | JULY 2025

1. INTRODUCTION

Transient Scheduler Attacks (TSA) are new speculative side channel attacks related to the execution timing of instructions under specific microarchitectural conditions. In some cases, an attacker may be able to use this timing information to infer data from other contexts, resulting in information leakage.

AMD is releasing updated CPU microcode which must be used in conjunction with operating system (OS) and hypervisor software updates to help mitigate TSA.

This paper provides technical background regarding TSA and how these attacks can be mitigated. AMD encourages developers to review this guidance and determine the appropriate mitigations for their environments.

As of the date of this whitepaper, only Family 19h products are known to be vulnerable to TSA.

2. TECHNICAL DESCRIPTION

Under specific microarchitectural conditions, instructions that read data from memory may experience an event known as "false completion". A false completion occurs when CPU hardware expects the load instruction to complete quickly (e.g., hit the L1 cache) but a condition arises which prevents the load from completing successfully. In this case, dependent operations may be scheduled for execution before the false completion is detected. As the load did not actually complete, data associated with that load is considered invalid. The load will be re-executed later in order to complete successfully, and any dependent operations will re-execute with the valid data when it is ready.

Unlike other speculative behavior such as Predictive Store Forwarding, loads that experience a false completion do not result in an eventual pipeline flush. While the invalid data associated with a false completion may be forwarded to dependent operations, load and store instructions which consume this data will not attempt to fetch data or update any cache or TLB state. As such, the value of this invalid data cannot be inferred using standard transient side channel methods.

In processors affected by TSA, the invalid data may however affect the timing of other instructions being executed by the CPU in a way that may be detectable by an attacker.

AMD has analyzed TSA and identified two sub-variants based on the source of the invalid data associated with a false completion. TSA-L1 refers to a TSA side channel through the L1 data cache, while TSA-SQ refers to a TSA side channel through the CPU store queue.

3. TSA VARIANTS

3.1. TSA-L1

Modern AMD processors employ a linear address based microtag[1] that assists with L1 data-cache lookups. Due to how the microtag is indexed, it is possible that a load may find a valid entry in the microtag even though the L1 does not actually contain the data for that load. If this occurs, it may result in a false completion where the invalid data associated with this false completion comes from the L1 entry associated with the matching microtag.

An attacker that is able to exploit TSA-L1 may therefore be able to infer any data that is present within the L1 data-cache, even if that data is from a different context. AMD's analysis has determined this may potentially result in information leakage from the OS kernel to a user application, from a hypervisor to a guest VM, from one user application to another, or from one guest VM to another.

TSA-L1 will not result in information leakage between SMT threads. The microtag structure is not shared between SMT threads in affected processors.

3.2. TSA-SQ

The store queue in an AMD processor contains store instructions in the process of being executed. These stores may sometimes forward data to matching loads through a process known as Store To Load Forwarding (STLF). In some cases, a load instruction may execute which matches the address of an older store, but the data associated with that older store is not yet available. This may result in a false completion of the load. If this occurs, the invalid data associated with this false completion may come from an older store which previously occupied that store queue entry.

An attacker that is able to exploit TSA-SQ may therefore be able to infer data used by older stores, even if those stores were executed in a different context. AMD's analysis has determined this may potentially result in information leakage from the OS kernel to a user application. AMD does not believe TSA-SQ is likely to result in information leakage between a hypervisor and guest VM, or between user applications or VMs because of the limited size of the store queue and the number of typical stores along these transition paths.

Due to the microarchitectural design of the store queue, TSA-SQ will not result in information leakage across SMT threads. If one of the SMT threads goes idle however, older stores from that thread may be visible to the sibling thread.

4. EXPLOITABILITY

Similar to other speculative side channels, AMD believes TSA is likely only exploitable by an attacker able to run arbitrary code on a machine, such as a malicious application or a malicious VM. AMD does not believe TSA is exploitable through malicious web sites.

The conditions required to exploit TSA are typically transitory as both the microtag and store queue will be updated after the CPU detects the false completion. Consequently, to reliably exfiltrate data, an attacker must typically be able to invoke the victim many times to repeatedly create the conditions for the false completion. This is most likely possible when the attacker and victim have an existing communication path, such as between an application and the OS kernel.

5. MITIGATION

Mitigation for TSA requires a combination of new microcode patches for affected systems as well as updates to OS and hypervisors. Please see the appendix for details on the minimum required microcode version. After the appropriate microcode patches are applied, CPL0 software may use the memory form of the VERW instruction to clear the microarchitectural data structures necessary to mitigate TSA-L1 and TSA-SQ. The existing architectural behavior of VERW is unmodified. Execution of VERW at non-CPL0 will not clear the relevant microarchitectural data structures.

If mitigation is required, AMD recommends OS and hypervisor software execute this VERW instruction whenever the system transitions from trusted to untrusted code, such as when the kernel exits to user-space, or when transitioning from the hypervisor to a VM. Additionally, AMD recommends executing the VERW instruction before the processor enters an idle state through HLT, MWAIT, or an IO C-State. If MWAIT is used, AMD recommends executing VERW before the MONITOR instruction to ensure the processor can transition into lower power states.

AMD believes the use of this mitigation may affect system performance. AMD recommends that system administrators evaluate the risk profile of their workloads to determine if mitigation is appropriate for their environment.

Note that VERW is the only mitigation recommended by AMD for TSA. Other cache flushing operations such as WBINVD or the FLUSH_CMD MSR (0x10B) may not be effective mitigations for TSA-L1.

6. CPUID UPDATES

To assist with the enumeration of capabilities related to TSA, AMD is defining three new CPUID bits related to TSA.

CPUID Fn8000_0021 ECX[2] (TSA_L1_NO). If this bit is 1, the CPU is not vulnerable to TSA-L1.

CPUID Fn8000_0021 ECX[1] (TSA_SQ_NO). If this bit is 1, the CPU is not vulnerable to TSA-SQ.

As of the date of this paper, AMD's analysis has determined that only Family 19h CPUs are vulnerable to TSA. AMD CPUs that are older than Family 19h are not vulnerable to TSA but do not set TSA_L1_NO or TSA_SQ_NO. Bare-metal software that detects such a CPU should assume the CPU is not vulnerable to TSA. Hypervisor software should synthesize the value of the TSA_L1_NO and TSA_SQ_NO CPUID bits on such platforms so guest software can rely on CPUID to detect if TSA mitigations are required.

AMD Family 1Ah models 00h-4Fh and 60h-7Fh are also not vulnerable to TSA but do not set TSA_L1_NO or TSA_SQ_NO. Future AMD CPUs will set these CPUID bits if appropriate.

CPUID Fn8000_0021 EAX[5] (VERW_CLEAR). If this bit is 1, the memory form of the VERW instruction may be used to help mitigate TSA.

Note that this bit will not be set even after the appropriate microcode patch is loaded on affected systems. Bare-metal software should check the loaded microcode patch level to determine if VERW may be used to help mitigate TSA. Hypervisor software should synthesize the value of VERW_CLEAR based on the loaded microcode patch level so guest software can rely on CPUID to detect if this functionality is present.

7. TSA AND AMD SEV-SNP

AMD believes TSA may result in the ability for a malicious hypervisor to infer data used by an AMD SEV-SNP guest. The microcode patches AMD is releasing for TSA will automatically mitigate this attack vector for AMD SEV-SNP guests. Guest owners should use the AMD SEV-SNP attestation report to verify that the appropriate microcode patch has been loaded by the host system.

To assist with the mitigation of TSA within AMD SEV-SNP VMs, AMD is releasing an SEV-SNP FW update for affected platforms that will allow the hypervisor to set the VERW_CLEAR CPUID bit in the CPUID page created during guest launch. Note that guest owners must still use attestation to verify that the appropriate microcode patch has been loaded to verify that the VERW instruction is effective for mitigating TSA.

On affected platforms, the SEV-SNP FW will not allow the hypervisor to set the TSA_L1_NO or TSA_SQ_NO CPUID bits.

8. CONCLUSION

Transient Scheduler Attacks are a new class of speculative side channels affecting certain AMD processors. AMD has developed mitigations for TSA which require OS and hypervisor changes in addition to updated CPU microcode. Additionally, AMD has defined new CPUID bits to assist software with the deployment of appropriate mitigations in both bare-metal and virtualized environments. This information is intended to assist operating system and hypervisor developers with the mitigation of these vulnerabilities.

AMD recommends that system administrators evaluate their deployments and risk profiles to determine if TSA mitigations are required for their specific environments.

1. See section 2.6.2.3 in "Software Optimization Guide for the AMD Zen4 Microarchitecture" (publication #57647)

DISCLAIMER

Any statements regarding security vulnerabilities or security features reflect AMD's understanding at the time of the statement. While AMD has taken care to prepare this document, it may contain technical inaccuracies, omissions, errors, and typographical errors, and AMD is not liable for or under any obligation to update or otherwise correct this information. Any computer system has risks of security vulnerabilities that cannot be completely prevented or mitigated. Any mitigations discussed herein are technical explanations of what the discussed mitigations are intended to accomplish. Actual mitigations in a resultant product may differ or may not meet the intended mitigations as described herein. AMD makes no representations or warranties with respect to the contents of this document, and assumes no liability of any kind, including the implied warranties of noninfringement, merchantability or fitness for particular purposes, with respect to the operation or use of AMD hardware, software or other products described herein. No license, including implied or arising by estoppel, to any intellectual property rights is granted by this document. Terms and limitations applicable to the purchase or use of AMD's products are as set forth in a signed agreement between the parties or in AMD's Standard Terms and Conditions of Sale.

AMD, the AMD Arrow logo, and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

© 2025 Advanced Micro Devices, Inc. All rights reserved

APPENDIX: MINIMUM MICROCODE REVISIONS

Family/Model/Stepping	Code Name	Microcode Version
Fam 19h, Model 01h, Stepping 1h	"Milan"	0x0A0011D7
Fam 19h, Model 01h, Stepping 2h	"Milan"	0x0A00123B
Fam 19h, Model 08h, Stepping 2h	"Chagall"	0x0A00820D
Fam 19h, Model 11h, Stepping 1h	"Genoa"	0x0A10114C
Fam 19h, Model 11h, Stepping 2h	"Genoa"	0x0A10124C
Fam 19h, Model 18h, Stepping 1h	"Stormpeak"	0x0A108109
Fam 19h, Model 21h, Stepping 0h	"Vermeer"	0x0A20102E
Fam 19h, Model 21h, Stepping 2h	"Vermeer"	0x0A201211
Fam 19h, Model 44h, Stepping 1h	"Rembrandt"	0x0A404108
Fam 19h, Model 50h, Stepping 0h	"Cezanne"	0x0A500012
Fam 19h, Model 61h, Stepping 2h	"Raphael"	0x0A60120A
Fam 19h, Model 74h, Stepping 1h	"Phoenix"	0x0A704108
Fam 19h, Model 75h, Stepping 2h	"Hawkpoint1"	0x0A705208
Fam 19h, Model 78h, Stepping 0h	"Phoenix2"	0x0A708008
Fam 19h, Model 7Ch, Stepping 0h	"Hawkpoint2"	0x0A70C008
Fam 19h, Model A0h, Stepping 2h	“Bergamo”	0x0AA00216