TECHNICAL DOSSIER // MICROARCHITECTURAL SECURITY & PROCESSOR DESIGN

Architectural Address Masking Side-Channel Vulnerabilities

Linear Address Masking (LAM), Speculative Deserialization, and Cache Probing Defenses

Published: October 30, 2026 | Distribution: Open Engineering Community | Document ID: LT-2026-B16-07

EXECUTIVE SUMMARY & MICROARCHITECTURAL SCOPE

Modern CPU architectures introduced pointer tagging features—Intel Linear Address Masking (LAM), AMD Upper Address Ignore (UAI), and ARM Top Byte Ignore (TBI)—to allow software to store metadata in unused canonical virtual address bits. SLAM (CVE-2023-45271) demonstrates that hardware implementations speculatively translate and execute using untranslated non-canonical address bits before hardware masking takes effect, creating new transient execution primitives that bypass KASLR and leak kernel secrets.

Attack Vector	Spectre-v1 (Bounds)	MDS (ZombieLoad)	Inception (Zen 4)	SLAM / SLAMDUNK
Target Mechanism	Branch Speculation	Line Fill Buffer	Branch History Buffer	Linear Address Masking (LAM)
CPU Architectures	Intel, AMD, ARM	Intel Skylake/Cascade	AMD Zen 3 / Zen 4	Intel (LAM), AMD (UAI), ARM (TBI)
Primitive Type	Out-of-Bounds Load	Transient LFB Sniff	Recursive Speculation	Uncanonical Address Dereference
KASLR Bypass	Requires Leak	Indirect Leak	Target Injection	Direct Speculative Dereference
Hardware Countermeasure	Speculative Barriers	Hardware Buffer Clear	Automated IBPB	Linear Address Separation (LASS)

1. Pointer Tagging and Address Masking Mechanics

64-bit virtual memory systems typically use 48 or 57 address bits. The remaining upper bits (e.g., bits 63:48 or 63:57) are reserved for canonical checking. Intel LAM, AMD UAI, and ARM TBI relax canonical enforcement, allowing user applications, garbage collectors, and memory sanitizers (ASan) to embed metadata tags in upper address bits. Hardware execution units automatically mask these bits prior to page table translation.

2. The Speculative Masking Disconnect

The SLAM microarchitectural flaw occurs during out-of-order memory disambiguation. When a tagged pointer is dereferenced, the memory execution engine initiates speculative address translation before verifying canonical validity or applying the mask. If an attacker crafts an uncanonical tagged pointer pointing into kernel memory space, the CPU speculatively loads kernel secrets into the L1 data cache before an architectural General Protection Fault (#GP) is raised.

3. Kernel Exploitation and KASLR De-randomization

Security researchers demonstrated that SLAM can be synthesized into reliable Spectre gadgets throughout unmodified Linux kernel codebases. By triggering speculative unmasked dereferences, attackers can determine whether a probed address is mapped or unmapped based on cache response times, completely defeating Kernel Address Space Layout Randomization (KASLR) in under five seconds and extracting root credentials.

4. Defenses: Linear Address Space Separation (LASS)

Remediating SLAM without crippling pointer-tagging performance requires hardware-enforced Linear Address Space Separation (LASS). LASS enforces architectural isolation between user space (lower half) and kernel space (upper half) at the earliest decode pipeline stage, preventing speculative translation across security rings regardless of tagged address bits.

STRATEGIC ENGINEERING TAKEAWAYS & DEPLOYMENT CHECKLIST

- **Hardware Masking Flaw:** CPUs speculatively dereference untranslated address bits before applying masking logic.
- **Bypasses KASLR and SMAP:** Enables high-speed speculative probing of protected kernel space without crashing the process.
- **Universal Tagging Risk:** Impacts Intel LAM, AMD UAI, and ARM TBI pointer-tagging implementations.
- **Hardware LASS Mitigation:** Requires architectural isolation (LASS) to block speculative user-to-kernel address transitions.