TECHNICAL DOSSIER // SECURITY & VPNS

BlindSide Microarchitectural Exploit: Bypassing ASLR via Speculative Probing

Hardware Fault Suppression, Speculative Crash-Proof Probing, and Kernel Code Exfiltration

Published: October 29, 2026 | Distribution: Open Engineering Community | Document ID: LT-2026-B15

EXECUTIVE SUMMARY & MICROARCHITECTURAL SCOPE

This security whitepaper details the microarchitectural mechanics of the BlindSide exploit. While modern operating systems enforce fine-grained Address Space Layout Randomization (ASLR) and execute-only memory (XOM), BlindSide leverages speculative execution to perform crash-free memory probing. By suppressing MMU page faults within transient execution windows, attackers probe randomized kernel addresses and detect cache allocations via differential timing, completely stripping entropy without crashing the victim process.

Mitigation Technique	Traditional Memory Disclosure	Spectre-v1 / Speculative Read	BlindSide Speculative Probing
Fault Handling	Fatal Crash (SIGSEGV / Panic)	Requires architecturally valid pointer	Crash-Proof (Hardware Fault Suppression)
Kernel ASLR Bypass	Requires uninitialized leak	Requires known branch condition	De-randomizes 28-bit ASLR in < 15 sec
Execution Primitives	Reads valid data buffers	Leaks cache side-channel data	Synthesizes ROP chain blindly
Hardware Vulnerability	Software memory safety bug	Branch Predictor speculation	Out-of-Order Execution Fault Squashing

1. Modern Defense Foundations: Fine-Grained ASLR & XOM

Address Space Layout Randomization (ASLR) and Kernel ASLR (KASLR) randomize the base virtual addresses of executable code, stacks, heaps, and drivers, forcing exploits to guess locations. Execute-Only Memory (XOM) further strips read permissions from code segments, preventing attackers from reading memory directly to find Return-Oriented Programming (ROP) gadgets. Traditionally, an incorrect probe triggers an unhandled page fault, terminating the process and alerting defenders.

2. Speculative Execution & Fault Suppression Mechanics

Modern out-of-order CPUs speculatively execute instructions ahead of branch resolution. If a speculative load targets an unmapped or privileged memory address, the MMU registers an internal fault. Crucially, the CPU does not architecturally raise the exception until the instruction reaches the Reorder Buffer (ROB) commit stage. If the branch resolves as mispredicted, the CPU flushes the pipeline and squashes the fault, leaving microarchitectural cache state altered.

3. The BlindSide Speculative Probing Primitive

BlindSide constructs a speculative gadget containing a conditional branch that mispredicts, followed by a speculative dereference of a candidate target address. If the candidate address is mapped in physical memory, the CPU loads the data and executes a secondary dependent memory access that brings a known probe array element into L1/L2 cache. If the address is unmapped, speculative execution stalls and squashes without allocating the probe array into cache.

4. De-randomization and Gadget Synthesis

By iterating through address ranges and measuring probe access times via RDTSC/RDTSCP (Flush+Reload), the attacker detects which addresses contain valid pages. BlindSide differentiates between code and data pages by probing for execute permissions, speculatively executing instructions at target offsets, and observing side effects. Within seconds, full kernel code layouts are reconstructed, enabling arbitrary kernel ROP chains without a single process crash.

STRATEGIC ENGINEERING TAKEAWAYS & DEPLOYMENT CHECKLIST

• **Fault Squashing Threat:** Hardware speculative fault suppression completely eliminates the fatal crash barrier that historically protected ASLR and memory safety.
• **Hardware Barrier Mitigation:** Inserting architectural serializing instructions (e.g., LFENCE, CSDB) after vulnerable indirect branches neutralizes speculative probe windows.
• **Kernel Memory Isolation:** Complete Kernel Page Table Isolation (KPTI) and strict physical page unmapping reduce the surface of speculatively probeable kernel memory.