Branch History Injection (BHI) Spectre-v2 Vulnerability Audit

Cross-Privilege BHB Aliasing, eIBRS Microarchitectural Bypasses, and Defense-in-Depth

Document ID: LT-WP-2026-1702 | **Classification:** Public Technical Audit | **Release:** October 31, 2026 | **Author:** Lesty Tech Research Lab

EXECUTIVE ABSTRACT: *Branch History Injection (BHI, CVE-2022-40982) breaks Enhanced Indirect Branch Restricted Speculation (eIBRS) by demonstrating that global branch history buffers (BHB) are shared across privilege domains. An unprivileged user-space attacker or virtual machine guest can manipulate the global branch history to induce branch target collisions in the indirect branch predictor, speculatively dispatching arbitrary kernel-mode disclosure gadgets and leaking sensitive physical memory across container boundaries.*

Architecture Defense	eIBRS Only	BHI Software Sequence	Hardware BHI_DIS_S
Protection Domain	Tag Isolation Only	Branch History Clearing	Global BHB Partitioning
Syscall Latency Overhead	<1.5% Baseline	8.5% - 14.2% Degradation	2.1% - 3.8% Degradation
Spectre-v2 Gadget Protection	Bypassed by BHI	Mitigated via lfence	Hardware Isolated

1. Microarchitectural Root Cause: Shared Global History

Modern high-performance out-of-order x86 processors decouple branch history tracking from target prediction. While eIBRS tags branch target buffer (BTB) entries with privilege levels to prevent direct target spoofing, the Branch History Register (BHR) remains shared across Ring 3 and Ring 0. An attacker executes a crafted sequence of taken/not-taken branches to inject a specific 194-bit pattern, steering the kernel's indirect branch predictor toward malicious transient target gadgets.

2. Cloud Hypervisor & Cross-VM Attack Surfaces

In virtualized cloud environments (KVM, Xen, Hyper-V), guest virtual machines execute on shared physical CPU threads. A malicious guest OS can prime the host physical CPU's BHB before executing a VM-Exit instruction. When the hypervisor handles the trap, its indirect calls (e.g., vhost handler function pointers) speculatively execute gadget instructions in the hypervisor context, leaking host kernel addresses and breaking KASLR.

3. Software Workarounds: The Loop-Clearing Penalty

Initial software mitigations injected a sequence of dummy branches (e.g., 32 consecutive call/ret or jmp loops) on every kernel and hypervisor entry to saturate the BHB and flush malicious guest history. However, this approach severely penalizes IO-intensive workloads, introducing a 10% to 15% system call latency tax on NVMe storage and 100GbE networking stacks.

4. Silicon-Level Remediation: BHI_DIS_S & Hardware Fencing

Modern x86-64 microarchitectures incorporate dedicated hardware mitigation bits (IA32_MCU_OPT_CTRL MSR, bit 0: BHI_DIS_S). When activated, the CPU core automatically decouples user-space branch history from supervisor predictor lookups at the hardware pipeline level. Combined with fine-grained compiler instrumentation (fine-IBT and kCFI), speculative gadget reachability is completely constrained without runtime loop overhead.

Key Strategic Architecture Takeaways

- eIBRS hardware alone is insufficient to prevent Spectre-v2 because branch history registers remain unsegmented.
- BHI allows unprivileged attackers to train the branch predictor to execute arbitrary kernel disclosure gadgets speculatively.
- Software loop-clearing incurs a heavy 8-14% performance penalty on system call throughput and network IOPS.
- Enabling silicon-level BHI_DIS_S hardware controls in microcode restores security with negligible (<3%) latency penalty.