

iLeakage Microarchitectural Security Audit

Speculative Type Confusion, Transient Cache Eviction Timers, and Cross-Origin Site Isolation Bypass

EXECUTIVE ABSTRACT & SUMMARY FINDINGS: *iLeakage (CVE-2023-41993) represents the first documented transient execution attack breaking browser site isolation across all modern Apple Silicon processors (A12 through M3). By reverse-engineering Safari WebKit's JavaScript engine and Apple CPU branch predictors, attackers bypass 32-bit compressed pointer sandboxing and timer poisoning. This audit examines speculative type confusion gadgets, eviction set generation, and cross-tab credential exfiltration telemetry.*

Vulnerability Layer	Architectural Vector	Leakage Rate	Mitigation Impact
Safari WebKit	32-bit Compressed Pointers	Full 64-bit Address	Sandboxing Defeated
Apple Silicon APM	Branch Predictor Injection	>99.2% Speculation	High Mispredict Gain
Cache Telemetry	Eviction Set Prime+Probe	24 bytes / sec	No Permissions Required
Lockdown Mode	Webkit Process Isolation	Suppressed (Patched)	15% - 25% Page Latency

1. The Collapse of Safari's Address Space Isolation

Modern web browsers rely on Site Isolation to place different web domains into mutually isolated OS processes. However, due to resource constraints in mobile environments, WebKit historically grouped multiple cross-origin iframes and tabs into shared render processes on iOS and macOS, relying on internal JavaScript heap sandboxing to prevent cross-origin memory access.

iLeakage invalidates this premise. When an attacker-controlled tab opens a victim target (such as Gmail or a bank login) in a background window or iframe, both pages frequently share the same underlying virtual address space. By inducing speculative execution in the shared WebKit process, an attacker reads arbitrary memory contents across domain boundaries.

2. High-Precision Cache Eviction Timers Without performance.now()

Browser vendors mitigated classic Spectre attacks by reducing the precision of performance.now() (poisoning timers to 1 millisecond intervals) and disabling SharedArrayBuffer. iLeakage bypasses timer restrictions by constructing a high-entropy eviction set timing primitive using WebAssembly shared memory racing.

By allocating large arrays and measuring cache line conflict contention against the CPU L1 and L2 caches, the attacker measures DRAM access latency differences (15ns vs 85ns) without calling high-precision timing APIs. This software-based stopwatch enables reliable Prime+Probe cache telemetry directly within an unprivileged browser sandbox.

3. Speculative Type Confusion and 32-Bit Pointer Escalation

Apple Silicon implements 32-bit compressed pointers in JavaScriptCore (JSC) to reduce memory bandwidth overhead. Pointers are stored as 32-bit offsets relative to a fixed 4GB Gigacage base register. To read memory outside the Gigacage, iLeakage exploits speculative type confusion.

During a mispredicted branch inside an object getter, the CPU speculatively treats a 64-bit integer supplied by the attacker as an object pointer. Because branch prediction validation requires dozens of clock cycles, the CPU executes speculative load instructions using the unmasked 64-bit address, transmitting the secret byte into the CPU data cache before the pipeline flushes the branch.

4. Architectural Countermeasures and Process Sandboxing

Mitigating iLeakage requires full out-of-process iframes in WebKit (enforced via Apple's Lockdown Mode) and speculative execution barriers. However, inserting speculation fences (CSDB / DSB on ARM) on all indirect branches incurs an intolerable 30% performance penalty on JavaScript execution. Complete mitigation requires hardware-enforced pointer authentication and memory tagging (MTE).

CRITICAL ENGINEERING TAKEAWAYS & DEPLOYMENT DIRECTIVES:

1. iLeakage bypasses Safari browser Site Isolation to exfiltrate passwords and emails at ~24 bytes/sec.
2. Software timer poisoning fails to prevent cache timing attacks due to WebAssembly eviction-set clock primitives.
3. Speculative type confusion in JavaScriptCore escapes 32-bit Gigacage sandboxes into arbitrary 64-bit address space.
4. Full remediation requires strict per-origin OS process isolation, increasing mobile memory footprints by 15-25%.