

PACMAN Apple Silicon Security Audit

Transient Execution Attack Dynamics, Pointer Authentication Bypass, and Hardware Exception Suppression

EXECUTIVE ARCHITECTURAL SUMMARY: Pointer Authentication Codes (PAC) enforce hardware-level control-flow integrity in Armv8.3-A and Apple Silicon (M1/M2/M3) by cryptographically signing pointer address bits with the QARMA algorithm. The PACMAN vulnerability (CVE-2022-26702) demonstrates that speculative execution can be turned into a side-channel oracle: incorrect PAC authentications executed speculatively suppress crash exceptions while modulating cache lines, enabling brute-force key recovery in under 3 minutes.

Defense / Attack Vector	Hardware Mechanism	Speculative Immunity	Brute-Force Feasibility	System Stability
Vanilla Buffer Overflow	Architectural Pointer Overwrite	None (No Speculation)	Trivial (Direct Hijack)	Deterministic Crash
Standard Arm MTE	4-Bit Physical Color Tags	Vulnerable (TikTag)	Possible (16 Combinations)	SIGSEGV on Mismatch
Architectural Arm PAC	16-Bit QARMA Cryptographic MAC	High (Architectural Path)	Impossible (Crash on Fail)	Immediate Kernel Panic
PACMAN Speculative Exploit	Speculative AUTIA/AUTDA Gadget	Zero (Pipeline Leak)	Guaranteed (Bypasses Crash)	Zero Crashes / Silent

1. Pointer Authentication Architecture (Armv8.3-A & Apple Silicon)

Modern 64-bit operating systems utilize only 48 or 52 bits of virtual address space. Armv8.3-A repurposes the unused upper 12 to 16 bits of a pointer to store a cryptographic Pointer Authentication Code (PAC).

- QARMA Tweakable Block Cipher: The CPU generates a PAC by encrypting the 64-bit pointer address, an architectural modifier (such as the stack pointer SP), and an internal secret 128-bit processor key (A-Key or B-Key).
- Authentication Traps: Before dereferencing a signed pointer, instructions like AUTIA or AUTDA recompute the MAC. If the signature matches, the pointer is stripped and executed; if it mismatches, the CPU inverts high address bits, triggering an uncatchable kernel panic upon dereference.

2. The PACMAN Gadget: Speculative Oracle Mechanics

The core discovery of PACMAN is that while architectural PAC execution crashes the processor on failure, transient speculative execution suppresses hardware exceptions.

- Speculative Branch Misprediction: An attacker trains a conditional branch predictor to expect a valid execution path. Behind the branch sits an AUTIA authentication instruction followed immediately by a memory load gadget.
- Suppression of Faults: The attacker passes an arbitrary brute-forced PAC guess. During the branch misprediction window, the CPU executes AUTIA speculatively. If the guess is correct, the load completes and primes a cache line. If incorrect, the pipeline flushes without crashing.

3. Cache Eviction & Microarchitectural Oracles

To extract the authentication result without triggering exceptions, PACMAN pairs speculative verification with a Prime+Probe or Flush+Reload microarchitectural side-channel:

- **Transmission Gadget:** The speculatively authenticated pointer is used to load an offset into a probe array: `array[pointer & 0x1]`.
- **Cache Hit Verification:** If the PAC guess was correct, the cache line is loaded into L1/L2. The attacker measures probe access latency: a sub-10ns access confirms a correct 16-bit cryptographic PAC signature.
- **Linear Search Convergence:** Because the system never panics, an attacker tests all 65,536 possible 16-bit PAC combinations in under 2.6 minutes, defeating kernel-level memory protections completely.

4. Mitigation Strategies & Silicon Hardening

Software mitigations require inserting heavy speculation barriers (DSB/ISB instructions) prior to every pointer authentication call, incurring severe 25% to 35% IPC performance degradation.

- **Hardware Redesign:** Future silicon architectures must enforce non-speculative PAC verification, preventing transient execution pipelines from dispatching dependent memory loads until cryptographic validity is architecturally confirmed.

STRATEGIC ENGINEERING TAKEAWAYS & DEPLOYMENT RULES:

- PACMAN transforms hardware pointer authentication into a speculative side-channel oracle with zero kernel crashes.
- Speculative execution completely bypasses Armv8.3 QARMA cryptographic integrity guarantees during branch stalls.
- Complete defense requires hardware speculation fencing before AUTIA/AUTDA instructions, trading IPC performance for integrity.