DROW: Training-Free Load Speculative Execution Attacks on Apple Silicon

Yuchen Fan* Yu Jin* Chang Liu† Minghong Sun* Xuanzeng Song* Tingting Yin‡
Shuwen Deng*§¶

Abstract

Traditional speculative attacks rely on training hardware predictors, limiting practicality. We present DROW, exploiting **blind bypassing** on Apple silicon, where loads speculatively bypass stores without training. This primitive circumvents mistraining defenses to hijack data and control flow. DROW achieves 16.3× higher bandwidth than prior art, enabling practical exploits including cross-page browser leakage,ASLR/KASLR bypassing, and PAC circumvention.

1 Introduction

Speculative execution attacks have fundamentally altered security [3], yet most rely on a *training* phase to prime microarchitectural predictors. This dependency necessitates deep microarchitectural knowledge and complex gadget preparation. In contrast, *training-free* attacks, which trigger speculation unconditionally, significantly lower the barrier to exploitation but remain largely explored only in control-flow contexts on x86 [1]. This work investigates whether training-free mechanisms exist for data-flow speculation in other architectures.

We identify **blind bypassing** on Apple M-series CPUs—a mechanism where loads speculatively bypass older stores without training. Based on this, we propose DROW, a training-free Spectre-type attack. We address challenges of exploiting training-free primitives, including the lack of trainability and the difficulty of hijacking control flow. Our contributions are: (1) disclosing the blind bypassing mechanism on Apple Silicon; (2) characterizing the Spectre-BB primitive, achieving 16.3× higher bandwidth than state-of-the-art training-based attacks [2]; and (3) demonstrating real-world attacks including leaking cross-page data in Chrome, bypassing ASLR/KASLR (100% success), and compromising ARM Pointer Authentication (PA).

*Department of Electronic Engineering, Tsinghua University
†Department of Computer Science and Technology, Tsinghua University
‡Independent researcher
§Zhongguancun Laboratory
¶Shuwen Deng is the corresponding author.

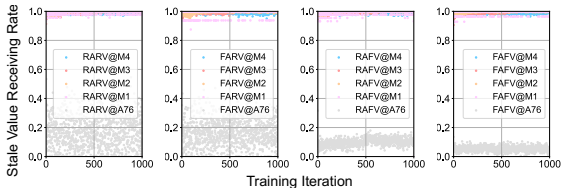

Figure 1: Bypass frequency vs. training iterations. Apple Silicon stays at 100% (Training-free), unlike Cortex.

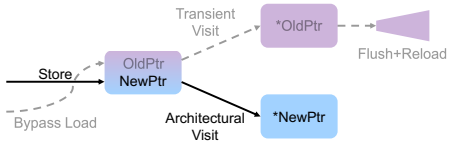

Figure 2: Overview of the Spectre-BB experiment. The speculative load retrieves a stale pointer to access secret data.

2 The Spectre-BB Primitive

Discovery of Blind Bypassing. We systematically analyzed load speculation on ARM and Apple Silicon. Figure 1 compares the bypass frequency under varying training iterations. On ARM Cortex-A76, the predictor learns to suppress the bypass as iterations increase (dropping to 0%). In contrast, Apple M-series CPUs maintain a ≈100% bypass rate regardless of iterations. This confirms the mechanism is *training-free* and stateless, making it immune to history-based defenses.

Primitive Construction. Figure 2 illustrates the Spectre-BB gadget. We deliberately stall a store instruction (e.g., via dependency chains). A subsequent load speculatively bypasses this store, retrieving a stale value (attacker-controlled pointer) from the memory hierarchy. The CPU speculatively dereferences this pointer, encoding the secret into the cache before the pipeline flushes.

Activation Conditions. Bypassing is suppressed only by *direct* (RAW) or *same-operand* dependencies. *Same-source* dependencies (e.g., array indexing) do not inhibit the bypass. The speculative window is limited primarily by ROB capacity.

Coherence-Based Eviction. We use a remote-core writing thread to induce coherence invalidations, creating a privilege-free timing channel to extract the signal from Figure 2.

3 Real-world Exploitation

We evaluate DROW's severity by identifying gadgets in major software and executing three end-to-end attacks on macOS 15.5 and Chrome v139.

Gadget Analysis & Capability. We developed a symbolic execution scanner based on *angr* to detect Spectre-BB gadgets (stalled store followed by dependent load). We identified 40,889 unique gadgets in the macOS XNU kernel, 60 in MbedTLS, and 13 in OpenSSL. In a native PoC on M3, Spectre-BB achieves a leakage bandwidth of 94.8 Kb/s with 99.07% accuracy, outperforming the training-based SLAP attack (5.8 Kb/s) by 16.3×.

Case 1: Breaking Browser Sandboxes. We bypass Chrome's site isolation using WebAssembly (WASM). Since WASM enforces strict runtime bounds checks, we exploit the *WASM Garbage Collection (GC)* proposal to induce a **Type Confusion**. We craft a gadget where the JIT compiler speculatively treats an attacker-controlled 64-bit integer as a pointer to a structured object (e.g., 'struct.ref'). Under Spectre-BB, the CPU bypasses the type check and dereferences this integer, allowing arbitrary memory reads. To exfiltrate data, we constructed a custom high-resolution timer in WASM, distinguishing L1 hits/misses. As shown in Figure 3, we achieve cross-site leakage at 21.3 bps with 94.7% accuracy.

Case 2: Breaking ASLR/KASLR. DROW serves as an oracle to distinguish mapped pages from unmapped holes. When Spectre-BB accesses an unmapped page, the speculative load fails to fill the cache; accessing a mapped page leaves a trace.

- **User ASLR:** We scan the virtual address space to locate the dyld shared cache, achieving 100% success in <140ms.
- **KASLR:** We leverage a userspace-triggerable gadget in the XNU kernel (via 'ioctl'). By passing guessed addresses to the kernel gadget, we probe the kernel map, breaking KASLR with 100% success in ≈24s.

Case 3: Breaking Pointer Authentication (PAC). ARM PAC signs pointers to ensure integrity. However, speculative execution suppresses authentication faults. On M1/M2/M3, we use Spectre-BB to speculatively execute the `autia` instruction. We brute-force the PAC signature (upper bits). Only a correct guess allows the subsequent load to succeed and fill the cache line. This side-channel allows forging valid pointers without the key (96.4% success on M3). *Insight on M4:* On the M4 chip, transient `autia` *always* speculates as successful regardless of correctness. While this breaks our oracle for key recovery, it implies the CPU aggressively ignores integrity checks during speculation.

4 Discussion & Conclusion

Related Work & Mitigation. Unlike training-based attacks like PACMAN [4] or SLAP [2], DROW exploits a *training-free* mechanism, offering stability and portability. Challenging mitigation: hardware fixes degrade performance, while software barriers in hot paths incur prohibitive overheads.

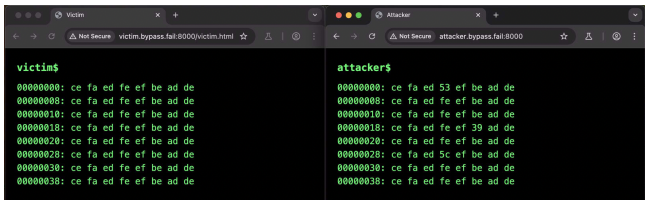

Figure 3: Cross-site data leakage in Chrome using DROW. We recover a secret string from a victim process with 94.7% accuracy.

Conclusion. We introduce DROW, a training-free primitive on Apple Silicon. Exploiting unconditional blind bypassing, we compromise browser sandboxes, ASLR/KASLR, and ARM PAC. This underscores the critical security risks of aggressive, stateless speculative optimizations.

Acknowledgment

This work was generously supported by the NSFC (No. U24A6009, 62572265), National Key Research and Development Program of China under Grant (2024YFB4405400), Youth Student Basic Research Project (Undergraduate Students) of the National Natural Science Foundation of China (Grant No. 625B1004), Beijing Municipal Science and Technology Project (Nos.Z241100004224028), Beijing Natural Science Foundation (L247013). We further would like to thank Shuo Li and Wenjian He from Linx Lab for offering feedback regarding the analysis of real-system vulnerabilities.

References

[1] Yun Chen, Ali Hajiabadi, and Trevor E Carlson. Gadgetspinner: A new transient execution primitive using the loop stream detector. In *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 15–30. IEEE, 2024.

[2] Jason Kim, Daniel Genkin, and Yuval Yarom. Slap: Data speculation attacks via load address prediction on apple silicon. In *S&P*, 2025.

[3] Paul Kocher, Jann Horn, Anders Fogh, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. Spectre attacks: Exploiting speculative execution. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1–19, 2019.

[4] Joseph Ravichandran, Weon Taek Na, Jay Lang, and Mengjia Yan. Pacman: attacking arm pointer authentication with speculative execution. In *Proceedings of the 49th Annual International Symposium on Computer Architecture*, pages 685–698, 2022.