Assignment 1: 4K-Aliasing Attack

Multi-Core Architecture and Systems | 0368-4183 | Gal Aharoni | 203521984

Summary

I implemented Herbert H. Hacker's proposed attack that exploits 4K-aliasing in store/load forwarding. The hypothesis: processors might speculatively forward stores based on matching lower 12 bits of virtual addresses before completing TLB translation.

Result: The attack failed completely. Zero successful leaks.
Conclusion: Processors validate physical addresses before forwarding, even speculatively.

Background

When a load needs data, the processor checks if there's a pending store to the same physical address in the ROB. If yes, it forwards data directly from the store buffer (bypassing cache). This requires matching physical addresses, which means waiting for TLB translation.

Intel's manual (Section 15.8) mentions a performance penalty for "4K-aliasing" - when two addresses satisfy A ≡ B (mod 4096). Since page size is 4096 bytes, the lower 12 bits are the offset within a page and stay unchanged after translation.

Herbert's hypothesis: processors might speculatively forward based on just the lower 12 bits to avoid waiting. Attack idea:

1. Store secret to address A
2. Load from address B where B ≡ A (mod 4096) but different physical page
3. Get secret through speculative forwarding
4. Leak via cache side-channel before speculation is squashed

Implementation

I used `mmap()` to allocate two separate pages, then created addresses with the same offset. For example: addr_a = 0x1100, addr_b = 0x5100, both with offset 0x100, verified by `(addr_a & 0xFFF) == (addr_b & 0xFFF)`.

Core attack:

```
void attack(uint8_t *store_addr, uint8_t *load_addr, uint8_t secret) {
  *store_addr = secret;
  uint8_t loaded = *load_addr;
  uint8_t dummy = array2[loaded * 512];  // Cache side-channel
}
```

If speculation forwards the secret, accessing `array2[secret * 512]` leaves a cache footprint.

For the side-channel, I used Flush+Reload: flush all cache lines, run attack, time access to each line. Fast access (< 80 cycles) = cache hit. Repeat 1000 times and find which byte has the most hits.

Results

Baseline: First verified with Spectre attack - successfully leaked "The password is rootkea"

4K-Aliasing Attack:
Test 'A' (0x41): Leaked 0x00, confidence 1 - FAILED
Test 'X' (0x58): Leaked 0x00, confidence 0 - FAILED
Test 'B' (0x42): Leaked 0x00, confidence 1 - FAILED
Test 0x7F: Leaked 0x00, confidence 1 - FAILED
Summary: 0/4 successful

All tests leaked 0x00 (uninitialized memory in the second page), not the secrets.

Why It Didn't Work

The core flaw in Herbert's reasoning: He assumed processors speculatively forward based on the lower 12 bits before validating physical addresses. This assumption is incorrect. The store buffer requires physical address matching before forwarding, even speculatively.

Physical Address Validation: The store buffer doesn't make forwarding decisions based on virtual address bits alone. Forwarding wrong data would break program correctness (not just performance), so the hardware waits for TLB translation. This is a design choice - correctness over aggressive speculation.

Store Buffer Architecture: If the store buffer used only lower 12 virtual bits for matching, there would be constant false matches in normal code (any stores and loads with same page offsets would trigger forwarding attempts). This would hurt performance, defeating the optimization's purpose. The store buffer must be indexed by physical addresses or include physical bits in its matching logic.

What Herbert Observed: The 4K-aliasing penalty in Intel's manual comes from conservative stalling or predictor confusion when aliasing is detected - NOT from speculative forwarding that could be exploited. The hardware recognizes the ambiguity and stalls rather than speculating.

Contrast with Spectre: Branch prediction aggressively speculates on control flow (accepts being wrong often). Memory disambiguation is fundamentally different - it's conservative because forwarding wrong data corrupts program state. This explains why Spectre works but 4K-aliasing doesn't.

Limitations

- Couldn't use `perf` (missing kernel tools)
- Only tested Intel on Linux 6.14.0
- Maybe there's a better trigger I didn't find

Conclusion

The attack doesn't work because processors validate physical addresses before forwarding. This isn't a bug in my implementation - the cache side-channel works (Spectre proves it). It's fundamental processor design.
While branch prediction is aggressive and vulnerable, the memory subsystem is conservative. The 4K-aliasing penalty comes from stalling or flushes, not exploitable speculation.

Takeaway: Not every performance quirk is a security vulnerability. Herbert's hypothesis was reasonable but doesn't match reality.