Restricting Control Flow During Speculative Execution
Zhuojia Shen, Jie Zhou, Divya Ojha, and John Criswell
University of Rochester

Meltdown and Spectre Attacks

- Exploit out-of-order and speculative execution
- Leak secret data via cache side channels
- Existing software defenses
- Retpoline
- Load fence
- Spectre-resistant SFI
- Speculative load hardening
- Still vulnerable to Branch Target Injection

Venkm: Our Software Solution

- Two Spectre defenses
  - Bundle Alignment + Branch Target Restrictions
  - Defeats Branch Target Injection [7]
  - Spectre-Resistant SFI on Stores
- Defeats Read-only Protection Bypass on code segment [7]

- Must instrument all code in the system
- Use a system that controls native code generation (e.g., SVA)
- Use an OS-level binary verifier (e.g., Google’s NaCl)

Bundle Alignment

- Instruction sequences
- Sized and aligned at a same fixed power of 2 (2^S)
- Control-flow transfers to the beginning of a bundle
  - Functions and basic blocks aligned at 2^S
  - Function calls at the end of a bundle

- Transform programs to bundles
  - Break larger basic blocks into smaller basic blocks
  - Add NOPs to the beginning of smaller basic blocks
  - Ensure instructions that must be co-located are not separated
  - Align all basic blocks at boundaries of 2^S

Branch Target Restrictions

- Restrict branch targets
  - All the branch targets point to the beginning of a bundle
  - All the branch targets within code segment
- Bit-masking instrumentations on indirect branches
  - If the bundle size is 2^S bytes:
    - Clear the lower S bits of the target address
  - If the code segment is placed in the first 2^T bytes:
    - Clear the upper (64 – T) bits of the target address
  - Dedicated returns transformed to indirect jumps
  - Co-locate instrumentations and the indirect branch in the same bundle

- Example of indirect function call (S = 4 and T = 32)
  1. mtcotr r27
  2. bl r27
  3. cllr d r27, r27, 4

- Example of return (S = 4 and T = 32)
  1. mtlr r0
  2. bl r0

Mitigate BTB/RSB poisoning with Bundle Alignment

- Only the starting address of a bundle can go into BTB/RSB
- Prevent existing Spectre defenses from being bypassed
- Form a complete defense against Branch Target Injection

Speculative Code Segment Integrity

- Spectre can break read-only memory protection [7]
  - Overwrite code segment speculatively
  - Spectre-resistant SFI [7] on stores
  - Create a data dependence between the SFI check and the store
  - Prevent stores from (speculatively) writing to code segment
  - Ensure the SFI check and the store are in the same bundle
  - Rearrange programs’ address space map
  - Make all code reside in one portion of virtual address space (below 2^S)
  - Strategically place code and data segments
    - Simplify SFI to use fewer instructions

Implementation and Preliminary Results

- Prototype implementation
  - Built on a 64-bit IBM POWER8 machine
  - Use LLVM compiler infrastructure
  - Implemented in two separate MachineFunctionPasses
- Spectre-resistant SFI on stores not implemented yet

- Evaluation on SPEC CPU 2017

Future Work

- Complete prototype implementation
  - Implement Spectre-resistant SFI on stores
  - Incorporate existing defenses
  - Apply Venkm on OS kernel
  - Measure memory overhead
  - Evaluation on more programs

Acknowledgements

This work is supported by The Office of Naval Research under Award Number 08813-16629.

References