Speculative Data-Oblivious Execution: Mobilizing Safe Prediction For Safe and Efficient Speculative Execution

Jiyong Yu, Namrata Mantri, Josep Torrellas, Adam Morrison*, Christopher W. Fletcher
University of Illinois at Urbana-Champaign, *Tel Aviv University

INTRODUCTION

Speculative Execution Attacks
• Access instructions speculatively read sensitive data into architectural state (e.g., registers)
• Transmit instructions transmit sensitive data via shared hardware states
• Goal: leak secret (speculatively-accessed data)

if (addr < N) { // speculation
    // access instruction
    uint8_t val = A[addr];
    // transmit instruction
    uint8_t tmp = B[64 * val];
}

Existing Mitigations

Defense
Strategy
Invisible Loads
Delayed Execution

Examples
InvisiSpec [MICRO’18]
SpecShield [PACT’19]
SafeSpec [DAC’19]
Conditional Spec. [HPCA’19]
CleanupSpec [MICRO’19]
NDA [MICRO’19]
STT [MICRO’19]

Pros
High-performance
High-security
Never block execution
Guarantee security properties (e.g., non-interference)

Cons
Low-security
Low-performance
Do not deliver rigid and comprehensive security
Block execution of transmit instructions

FOUNDATION: SPECULATIVE TAINT TRACKING

Key feature: Blocking implicit channels
• For prediction: secret data cannot update predictors/be used for prediction
• For resolution: delay resolution (squashes) until condition is no longer secret

Observation: STT makes prediction SAFE
• Once applying the implicit channel protection, we can use prediction for performance optimization without worrying about any speculation leakage!

KEY IDEAS

Idea 1: Execute transmit instructions in a data-oblivious fashion → worst-case execution
Idea 2: Avoid worst-case execution by predicting how the execution should be performed
Idea 3: Protect the prediction with STT’s implicit channel protection

Key capability: execute unsafe transmitters early and safely

SPECULATIVE DATA OBLIVIOUS EXECUTION (SDO)

SDO Framework
• Define Data-Oblivious (DO) variants for a given transmit instruction
  o Each DO variant must be data-oblivious
  o Each DO variant may produce invalid result unless inputs satisfies certain condition
• Create dedicated DO predictor to predict DO variant at runtime
  o [Follow STT’s protection] At runtime:
    o Secret data cannot update DO predictor/be used for predicting DO variant
    o Delay resolution (squash) until condition is no longer secret

DO variant signatures
(dest, success) < DO-op, args
(dest, success) < DO-op, args

Resolving when safe
(destination is no longer secret)
Predictor update(...)
if (success,)
squash from “dest < op args”

DO variant must be data-oblivious

ATTACK VECTORS

Idea 3: Protect the prediction with STT’s implicit channel protection
• Customized DO predictor for loads (cache level predictor). General metrics:
  • Accurate and precise: predicted cache level equal to actual cache level
  • Accurate but imprecise: predicted cache level lower than actual cache level
  • Inaccurate: predicted cache level higher than actual cache level
• Resolve DO prediction when safe
  o Update predictor; squash pipeline if success = FALSE
  o For multi-processor:
    o DO-Ld_X must not modify cache state
    o Data fetched by DO-Ld_X may not be cached in L1
    o May miss cache invalidation
  o Solution: Apply invalidation infrastructure from InvisiSpec [MICRO’18]

PERFORMANCE EVALUATION

Evaluation Settings
• Gem5 simulator, w/ 3-layer cache with MESI protocol
  o Floating-point multiply/divide: always predict non-subnormal
  o Load: evaluating multiple DO predictors
    o Static L1: always predict DO-Ld_L1
    o Static L2: always predict DO-Ld_L2
    o Static L3: always predict DO-Ld_L3
    o Hybrid: our customized tournament cache-level predictor
    o Perfect: a theoretically-best DO predictor (oracle)

CONCLUSIONS

• SDO is a new speculative execution attack mitigation framework that enables strong security (equivalent to STT) and high performance
• Key ideas
  o STT provides principles for safe prediction and resolution
  o SDO uses safe prediction/resolution to execute transmit instruction early and safely by combining prediction with data-obliviousness

ACKNOWLEDGEMENTS

This work was funded in part by NSF under grant CNS-1816226, Blavatnik ICRC at TAU, ISF under grant 2005/17,and by an Intel Strategic Research Alliance (ISRA) grant.