SecureCells: A Secure Compartmentalized Architecture

Atri Bhattacharyya  Florian Hofhammer  Yuanlong Li
Siddharth Gupta  Andres Sanchez
Babak Falsafi  Mathias Payer
EPFL
firstname.lastname[at]epfl.ch
Motivation

Modern software is complex, untrusted
  • Buggy, malicious code

Compartmentalization
  • A crucial layer of defense

Numerous applications
  • E.g., browsers, server workloads, OSs

Mitigate high-impact vulnerabilities
  • E.g., Log4j, Heartbleed

Compartmentalization is a broadly applicable defense
Pitfalls for Existing Mechanisms

- **Compromise on security**
  - MPK-based mechanisms lack checks for code fetch

- **High performance overheads**
  - Process-based isolation with microsecond-scale system calls

- **Specialize for specific application scenarios**
  - CODOM prevents cross-compartment code sharing

Existing mechanisms inhibit widespread adoption
SecureCells: A Novel VM Architecture

- **Hardware-enforced security**
  - Strict checks on memory accesses, call gates

- **Common operations are fast**
  - VMA-granularity access control
  - Accelerated unprivileged instructions

- **Supports generic application scenarios**

- **Requires software/hardware changes**

SecureCells enables compartmentalization for a spectrum of applications
SecureCells: Abstractions

Compartment
- Security Division SecDiv_M
- Virtual Mem. Areas \{Cell_N\}
- Entry points
SecureCells: Abstractions

Compartment
- Security Division SecDivM
- Set of Cells \{Cell_n\}
- Entry points
SecureCells Design: Access Control

Cell-granularity access control

PTable stores permissions
- Replaces traditional page tables
- Per-SecDiv, per-Cell entries
- Independent read (r), write (w), execute (x)
- Optimized layout for fast lookups

Per-core MMU checks permissions
SecureCells Design: Instructions

Unprivileged insts. accelerate common operations

SDSwitch/SDEntry

SCGrant

SCRecv

SCProt

SCExcl

1-2 orders of magnitude faster than system calls

Strict security checks
SecureCells Implementation

Supports in-order and out-of-order cores

Access Control
  • Per-core User SecDiv Identifier (USID) register
  • Cell-based MMU

Userspace instructions

FPGA prototype
  • RISC-V based RocketChip
  • 8-cycle compartment switch
  • ~200-cycle permission transfers

SecureCells’ design is practical
Conclusion

SecureCells targets pervasive compartmentalization
Identifies and provides key requirements

Access control and userspace instructions

Fully open-sourced infrastructure, prototype
https://hexhive.epfl.ch/securecells