SHELTER: Extending Arm CCA with Isolation in User Space

Yiming Zhang¹,²*, Yuxin Hu¹*, Zhenyu Ning³,¹, Fengwei Zhang¹✉, Xiapu Luo², Haoyang Huang¹, Shoumeng Yan⁴, Zhengyu He⁴

¹Southern University of Science and Technology, ²The Hong Kong Polytechnic University, ³Hunan University, ⁴Ant Group
Confidential Computing

- Hardware-assisted security design
- Cloud and Edge devices
- Intel TDX, AMD SEV, Arm CCA
Arm Confidential Compute Architecture (CCA)

- Introduced as supplement to Armv9.2-A
- Two added additional worlds
  - Secure -> Secure & EL3 Root
  - Normal -> Normal & Realm
- CCA is implemented in hardware and firmware

RME: Realm Management Extension  RMM: Realm Management Monitor  RMI: Realm Management Interface  RSI: Realm Services Interface
Arm Confidential Compute Architecture (CCA)

The current version of CCA:

- Early Stage
- Construction of Realm VMs
- Hypervisor-based Virtualization

CCA is implemented in hardware and firmware.
Motivation

- Cooperating with CCA hardware to provide user-level isolation
- Complement to CCA’s Realm VM architecture
Threat Model & Assumptions

• An attacker can compromise Host OS, hypervisor, or privileged software in Secure, and Realm world (e.g., SPM or RMM)

• The Monitor is trusted and the hardware is correctly implemented

• Physical/Side-channel/denial-of-service attacks are out of scope

• Assuming remote attestation support and secure boot
Shelter

- SHELTER App (SApp)
  - Running on Normal World EL0
- Host OS
  - Non-security responsibilities
- Shelter Monitor
  - In Root world
  - Security responsibilities
- CCA hardware feature
  - Realm Management Extension (RME)
Granule Protection Check (GPC)

- RME enforced isolation is managed through a new Granule Protection Table (GPT)
- GPT is controlled by the Monitor in EL3
- GPT specifies what physical address spaces (PAS) a memory page belongs to

<table>
<thead>
<tr>
<th>Security state</th>
<th>Normal PAS</th>
<th>Secure PAS</th>
<th>Realm PAS</th>
<th>Root PAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Secure</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Realm</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>Root</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Granule Protection Check (GPC)

- RME enforced isolation is managed through a new Granule Protection Table (GPT)
- GPT is controlled by the Monitor in EL3
- GPT specifies what physical address spaces (PAS) a memory page belongs to

It is not satisfied with the goal of isolating memory between SApps and other privileged software in Normal, Secure, and Realm world.
Multi-GPT Memory Isolation

- Maintain multiple GPTs in EL3 Monitor
- Divide the physical address space (PAS) for different programs

---

**Multi-GPT Permission Configuration**

<table>
<thead>
<tr>
<th>GPTs</th>
<th>MSApp</th>
<th>MSApps</th>
<th>MS</th>
<th>MREE</th>
<th>MM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host</td>
<td>--</td>
<td>--</td>
<td>Normal</td>
<td>Normal</td>
<td>Root</td>
</tr>
<tr>
<td>SApp</td>
<td>Normal</td>
<td>--</td>
<td>Normal</td>
<td>--</td>
<td>Root</td>
</tr>
</tbody>
</table>

---

**Memory Isolation Enforced by RME**

**REE memory** ($M_{REE}$)

**Shared Memory** ($M_S$)

**SApp** ($M_{SApp}$)

**Other SApps** ($M_{SApps}$)

**Monitor** ($M_M$)
Multi-GPT Memory Isolation

- Establishing address-space-per-core for each SApp and other code region
Multi-GPT Memory Isolation

- The Monitor dynamically controls the access permissions of different programs
Performance Optimization

• New GPT construction causes long startup latency for SApps
  • **Root cause:** Shelter needs to add granule information containing a layout of the entire main memory for the new GPT and measure each GPT entry
Performance Optimization

• New GPT construction causes long startup latency for SApps
  • Root cause: Shelter needs to add granule information containing a layout of the entire main memory for the new GPT and measure each GPT entry

*Using shadow GPT, a template with copy and update to speed up SApp creation*
TLB-based GPT attack

- GPT entries are permitted to be cached in TLB as part of TLB entry
- GPT information in a TLB is permitted to be shared across multiple CPU cores
Defend against TLB-based GPT attacks

- TLB invalidation during switches and GPT modifications
- Disable the shareable property of TLB
Some Execution Features

- Memory management
  - Contiguous physical memory pool
  - Ensure multiple SApps do not have memory overlap
  - SApp Page table is isolated
Some Execution Features

- Memory management
- Contiguous physical memory pool
- Ensure multiple SApps do not have memory overlap
- SApp Page table is isolated
- Syscall & Iago attack checks
- Interrupt & Signal
- Multi-threaded synchronization primitive
Shelter Implementation

- Functional prototype implementation
  - FVP Base RevC-2xAEMvA with RME-enabled features
  - TCB: ATF with 2k SLoCs additions

- Official CCA software stacks
  - TCB: ATF + TF - RMM (released date 2022/11/09)
  - TF - RMM (v0.2.0) is around 8.2k SLoCs

- TCB comparison with CCA
  - 2k vs 8k SLoCs additions
Shelter Implementation

• Functional prototype implementation
  • FVP Base RevC-2xAEMvA with RME-enabled features
  • TCB: ATF with 2k SLoCs additions
• Official CCA software stacks
  • TCB: ATF + TF-RMM (released date 2022/11/09)
  • TF-RMM(v0.2.0) is around 8.2k SLoCs
• TCB comparison with CCA
  • 2k vs 8k SLoCs
Performance Evaluation

- No commercial hardware supporting CCA is available on the market
- FVP Simulator is **not cycle accurate**
Performance Evaluation

• No commercial hardware supporting CCA is available on the market
  • FVP Simulator is not cycle accurate

• GPT-analogue in Armv8-A Juno Board
  • Mimic all GPT in-memory operations
  • Replace the GPT-related registers with idle EL3 registers
  • Invalidate all TLBs instead of TLB GPT invalidation instructions (e.g., TLBI PAALLOS)
  • The other functionality are the same as those on the FVP
Application Benchmarks

SHELTER incurs <15% runtime-overhead on real-world workloads compared with Linux
Performance Optimization

 ✓ With shadow GPT, reducing overhead on average of 77.5% in SApp Creation
Comparison with CCA’s VM-based approach

- A basic CCA VM-based performance prototype with same GPT-analogue methodology and a Realm-context simulation

![Runtime Overhead on three large-scale applications](image)

- Avg. SHELTER 11.7% vs CCA Realm VM 32.0%
Conclusion

• **Shelter** leverages CCA hardware for a new creation of **user-level** isolated environment
  • complementary to CCA’s primary Realm VM-style architecture
  • A smaller TCB
  • Lower performance overhead
  • No hardware modification for compatible platforms, including mobile and server
• Open Source
  • [https://github.com/Compass-All/Shelter](https://github.com/Compass-All/Shelter)
Thanks for listening!

Q & A

yming.zhang@connect.polyu.hk