SHELTER: Extending Arm CCA with Isolation in User Space

Yiming Zhang\textsuperscript{1,2}, Yuxin Hu\textsuperscript{1}, Zhenyu Ning\textsuperscript{3,1}, Fengwei Zhang\textsuperscript{1✉}, Xiapu Luo\textsuperscript{2}, Haoyang Huang\textsuperscript{1}, Shoumeng Yan\textsuperscript{4}, Zhengyu He\textsuperscript{4}

\textsuperscript{1}Southern University of Science and Technology, \textsuperscript{2}The Hong Kong Polytechnic University, \textsuperscript{3}Hunan University, \textsuperscript{4}Ant 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)

- Introduced as supplement to Armv9.2-A
- Hypervisor-based confidential VMs
- Confidential compute for 3rd parties

- Two added additional worlds
  - Secure
  - > Secure & EL3 Root
  - Normal
  - > Normal & Realm

The current version of CCA:
- Early Stage
- Construction of Realm VMs
- Hypervisor-based Virtualization

CCA is implemented in hardware and firmware

App
OS Kernel
VM1
Host
RME: Realm Management Extension
RMM: Realm Management Monitor
RMI: Realm Management Interface
RSI: Realm Services Interface
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
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 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

![Diagram](image-url)
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

- Syscall & lago attack checks
- Interrupt & Signal
- Multi-threaded synchronization primitive
## Security Analysis

<table>
<thead>
<tr>
<th>Adversary Subject</th>
<th>Main Attacks</th>
<th>Defense</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS/Hypervisor</td>
<td>Unauthorized memory access</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Invalid mapping or return value</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Illegal GPT modification</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>GPC circumvention</td>
<td>5</td>
</tr>
<tr>
<td>SApp</td>
<td>SApp abuse</td>
<td>12</td>
</tr>
<tr>
<td>TLB/Cache</td>
<td>Untended GPT sharing in TLB</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Unauthorized cache access</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>EL3 code cache injection</td>
<td>5</td>
</tr>
<tr>
<td>Peripherals</td>
<td>Malicious DMA</td>
<td>1</td>
</tr>
</tbody>
</table>

1 Multi-GPT isolation enforced by GPC; 2 Monitor checks (e.g., ensuring no memory overlap between SApps, checking syscall return value, verifying validity of the runtime); 3 Multi-core synchronization; 4 Microarchitectural Maintenance; 5 Monitor Maintaining in the highest privilege.
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**
- 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

Runtime Overhead on *real-world programs*

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
zhangfw@sustech.edu.cn