Prime+Reset: Introducing A Novel Cross-World Covert-Channel Through Comprehensive Security Analysis on ARM TrustZone

Yun Chen*  
National University of Singapore

Arash Pashrashid*  
National University of Singapore
Huawei Research Center

Yongzheng Wu  
Huawei Research Center

Trevor E. Carlson  
National University of Singapore

* Equal Contribution
Trusted Execution Environments

Classic Security Framework

- Hypervisor
- OS
- CPU

Applications:
- Biometric Authentication
- Electronic Payments
- Digital Rights Management

TEE Framework

- Hypervisor
- OS
- CPU

Trust Boundary

Code

Data
TrustZone TEE Architecture and Its Limitations

Limitations and Problems:
- TEEs are still vulnerable to various microarchitectural side channel attacks
- Lack of comprehensive microarchitectural security analysis of TrustZone

Prior Works:
- Limited scope: Only considering cache and PMUs as a source of leakage

Motivation:
- A comprehensive side/covert-channel vulnerability analysis on TrustZone
- Detecting a new leakage source
Leakage Analysis Criteria

**Question:** Are components of an OoO processor isolated between the Secure World and the Normal World?

Requirements for a successful side/covert channel:

- **RQ1**
  Shared resources among different execution domains that can create resource contention and execution footprints (e.g., cache)

- **RQ2**
  Microarchitectural events that can create distinguishable and leaking events during execution (e.g., out-of-order execution)

Analysis result:

<table>
<thead>
<tr>
<th>Component</th>
<th>Issue Ports</th>
<th>TLB</th>
<th>L1 Prefetcher</th>
<th>L1 Cache</th>
<th>L2 Prefetcher</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leakage?</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Reason</td>
<td>Pipeline flushing</td>
<td>Pipeline flushing</td>
<td>Entry invalidating</td>
<td>Entry invalidating</td>
<td>Resetting between worlds</td>
</tr>
</tbody>
</table>
Our Observation from L2 Prefetcher

Prefetcher is statically partitioned per core

Core 1 generates a segment fault

All blocks are reset after Core 1 segment fault
**PRIME+RESET: Covert-Channel using the L2 Prefetcher**

- **Sender** (Core 0):
  - Send b’0
  - Send b’1
  - Idle
  - Generate SIGSEGV

- **Receiver** (Core 1):
  - Train an entry in L2 prefetcher
  - Test the availability of trained entry

### Attack Table

<table>
<thead>
<tr>
<th>Attack</th>
<th>Error Rate</th>
<th>Max Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prime+Count&lt;sup&gt;1&lt;/sup&gt;</td>
<td>&lt;2%</td>
<td>~ 1 Kib/s</td>
</tr>
<tr>
<td>μarch-Count&lt;sup&gt;2&lt;/sup&gt;</td>
<td>&lt;2%</td>
<td>12 Kib/s</td>
</tr>
<tr>
<td>PRIME+RESET (ours)</td>
<td>&lt;2%</td>
<td>776 Kib/s</td>
</tr>
</tbody>
</table>