AutoLock: Why Cache Attacks on ARM Are Harder Than You Think

Marc Green\textsuperscript{W}, Leandro Rodrigues-Lima\textsuperscript{F}, Andreas Zankl\textsuperscript{F}, Gorka Irazoqui\textsuperscript{W}, Johann Heyszl\textsuperscript{F}, and Thomas Eisenbarth\textsuperscript{W}

August 18\textsuperscript{th} 2017 – USENIX Security Symposium
### The Big Picture

<table>
<thead>
<tr>
<th>Cache Attacks</th>
<th>AutoLock</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transition of attacks: desktop &amp; server → mobile</td>
<td>Dedicated countermeasures are still necessary for protection</td>
</tr>
<tr>
<td>Vulnerability and risk assessment are important, but challenging</td>
<td>Eviction-based attacks are harder than previously believed</td>
</tr>
<tr>
<td>Limited understanding of commercial microarchitectures</td>
<td>Undocumented performance feature of inclusive caches on ARM</td>
</tr>
</tbody>
</table>

---

**AutoLock: Why Cache Attacks on ARM Are Harder Than You Think | Green et al. | USENIX Security 2017 | 2**
Cache Basics
Cache Basics

Hierarchy

AutoLock: Why Cache Attacks on ARM Are Harder Than You Think | Green et al. | USENIX Security 2017 | 4

Fraunhofer AISEC
Cache Basics

Hierarchy

AutoLock: Why Cache Attacks on ARM Are Harder Than You Think | Green et al. | USENIX Security 2017 | 5
Cache Basics

Hierarchy

AutoLock: Why Cache Attacks on ARM Are Harder Than You Think | Green et al. | USENIX Security 2017 | 6
Cache Basics

Hierarchy

AutoLock: Why Cache Attacks on ARM Are Harder Than You Think | Green et al. | USENIX Security 2017 | 7
Cache Basics
Hierarchy

AutoLock: Why Cache Attacks on ARM Are Harder Than You Think | Green et al. | USENIX Security 2017 | 8
Cache Basics

Hierarchy

AutoLock: Why Cache Attacks on ARM Are Harder Than You Think | Green et al. | USENIX Security 2017 | 9
Cache Basics

Inclusiveness

![Cache Diagram]

**AutoLock: Why Cache Attacks on ARM Are Harder Than You Think** | Green et al. | USENIX Security 2017 | 10
AutoLock: Why Cache Attacks on ARM Are Harder Than You Think | Green et al. | USENIX Security 2017 | 11

Cache Basics
Inclusiveness

Diagram showing cache structures and memory layout.
Cache Basics

Inclusiveness

AutoLock: Why Cache Attacks on ARM Are Harder Than You Think | Green et al. | USENIX Security 2017 | 12
Inclusiveness

AutoLock: Why Cache Attacks on ARM Are Harder Than You Think | Green et al. | USENIX Security 2017 | 13
Cache Basics

Inclusiveness

AutoLock: Why Cache Attacks on ARM Are Harder Than You Think | Green et al. | USENIX Security 2017 | 14
Cache Basics

Inclusiveness

AutoLock: Why Cache Attacks on ARM Are Harder Than You Think | Green et al. | USENIX Security 2017 | 15
Cache Basics

Inclusiveness

AutoLock: Why Cache Attacks on ARM Are Harder Than You Think  |  Green et al. | USENIX Security 2017  |  16
Cache Basics

Inclusiveness

AutoLock: Why Cache Attacks on ARM Are Harder Than You Think | Green et al. | USENIX Security 2017 | 17
AutoLock: Why Cache Attacks on ARM Are Harder Than You Think | Green et al. | USENIX Security 2017 | 18
Cache Attacks
Cache Attacks
Cross-core Eviction

AutoLock: Why Cache Attacks on ARM Are Harder Than You Think | Green et al. |USENIX Security 2017 | 20
Cache Attacks

Cross-core Eviction

AutoLock: Why Cache Attacks on ARM Are Harder Than You Think | Green et al. | USENIX Security 2017 | 21
Cache Attacks
Cross-core Eviction

AutoLock: Why Cache Attacks on ARM Are Harder Than You Think | Green et al. | USENIX Security 2017 | 22
Cache Attacks
Cross-core Eviction

AutoLock: Why Cache Attacks on ARM Are Harder Than You Think | Green et al. | USENIX Security 2017 | 23
Cache Attacks

Cross-core Eviction

AutoLock: Why Cache Attacks on ARM Are Harder Than You Think | Green et al. | USENIX Security 2017 | 24
Cache Attacks

Cross-core Eviction

AutoLock: Why Cache Attacks on ARM Are Harder Than You Think | Green et al. | USENIX Security 2017 | 25
Cache Attacks
Cross-core Eviction

![Diagram showing cache eviction across cores.]

**AutoLock: Why Cache Attacks on ARM Are Harder Than You Think | Green et al. | USENIX Security 2017 | 26**
Cache Attacks
Cross-core Eviction

AutoLock: Why Cache Attacks on ARM Are Harder Than You Think | Green et al. | USENIX Security 2017 | 27
Cache Attacks
Cross-core Eviction

![Diagram showing cache and memory structures with core evictions and an auto lock mechanism.]
Cache Attacks
Cross-core Eviction

AutoLock: Why Cache Attacks on ARM Are Harder Than You Think | Green et al. | USENIX Security 2017 | 29
Cache Attacks
Cross-core Eviction

Qualcomm Krait 450

AutoLock: Why Cache Attacks on ARM Are Harder Than You Think | Green et al. | USENIX Security 2017 | 30
### Cache Attacks

#### Literature

<table>
<thead>
<tr>
<th>Method</th>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evict + Time</td>
<td>Eviction</td>
<td>[OST06]</td>
</tr>
<tr>
<td>Prime + Probe</td>
<td>Eviction</td>
<td>[OST06]</td>
</tr>
<tr>
<td>Evict + Reload</td>
<td>Eviction</td>
<td>[GSM15]</td>
</tr>
<tr>
<td>Evict + Prefetch</td>
<td>Eviction</td>
<td>[GMF+ 16]</td>
</tr>
<tr>
<td>Flush + Reload</td>
<td>Flush</td>
<td>[YF14]</td>
</tr>
<tr>
<td>Flush + Prefetch</td>
<td>Flush</td>
<td>[GMF+ 16]</td>
</tr>
<tr>
<td>Flush + Flush</td>
<td>Flush</td>
<td>[GMWM16]</td>
</tr>
<tr>
<td><strong>Flush on ARM</strong></td>
<td><strong>Eviction on ARM</strong></td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>---------------------</td>
<td></td>
</tr>
<tr>
<td>Easy, fast, and robust</td>
<td>Complex, slow, and error-prone</td>
<td></td>
</tr>
<tr>
<td>Only from ARMv8 onwards</td>
<td>All ARM architectures: v6, v7, v8</td>
<td></td>
</tr>
<tr>
<td>No guaranteed userspace access</td>
<td>Userspace privileges are sufficient</td>
<td></td>
</tr>
</tbody>
</table>

**Limited number of devices**

**Larger number of devices**

AutoLock: Why Cache Attacks on ARM Are Harder Than You Think | Green et al. | USENIX Security 2017 | 32
AutoLock
AutoLock

Cross-core Eviction

![Diagram](image)

AutoLock: Why Cache Attacks on ARM Are Harder Than You Think | Green et al. | USENIX Security 2017 | 34
AutoLock
Cross-core Eviction

Core 0
L1-I {...}
L1-D c
Core 1
I D
Core 2
I D
Core 3
I D {...} s t

L2 Cache (LLC) + AutoLock

prog_A{
...
}
RAM
prog_B{
...
}
AutoLock
Cross-core Eviction

AutoLock: Why Cache Attacks on ARM Are Harder Than You Think | Green et al. | USENIX Security 2017 | 37
AutoLock

Cross-core Eviction

Core 0
L1-I
{...}
L1-D
c

Core 1
I
D
t

Core 2
I
D

Core 3
I
D
u
{...}
t

L2 Cache (LLC) + AutoLock

prog_A{
...,
}

RAM

prog_B{
....,
}
AutoLock
Cross-core Eviction

Core 0
L1-I
{...}
L1-D
c

Core 1
I
D

Core 2
I
D

Core 3
I
D
{...}

t

L2 Cache
( LLC )
+ AutoLock

prog_A{
    ...
}

RAM

prog_B{
    ....
}
AutoLock

Cross-core Eviction

Core 0
L1-I
{...}
L1-D
c
Core 1
I
D
Core 2
I
D
Core 3
I
D
{...}
u
t

L2 Cache (LLC) + AutoLock

prog_A{
  ...
  
}
RAM

prog_B{
  ....
  
}
AutoLock

Cross-core Eviction

![Diagram showing L1-I, L1-D, I, D, and L2 Cache (LLC) + AutoLock]

prog_A{
    ...
}

RAM

prog_B{
    ....
}
AutoLock: Why Cache Attacks on ARM Are Harder Than You Think | Green et al. | USENIX Security 2017 | 42
AutoLock
Cross-core Eviction

![Diagram of AutoLock and cross-core eviction]

AutoLock: Why Cache Attacks on ARM Are Harder Than You Think | Green et al. | USENIX Security 2017 | 43
AutoLock
Cross-core Eviction

Core 0
L1-I
{...}
L1-D
c

Core 1
I
D
t

Core 2
I
D
c

Core 3
I
D
u
t

L2 Cache (LLC) + AutoLock

prog_A{
...
}

RAM

prog_B{
......
}
AutoLock

Cross-core Eviction

![Diagram showing the layout of cores and cache levels](image-url)
AutoLock
Cross-core Eviction

AutoLock: Why Cache Attacks on ARM Are Harder Than You Think | Green et al. | USENIX Security 2017 | 46
AutoLock
Cross-core Eviction

```
prog_A{
    ...
}
```

```
prog_B{
    .......
}
```

AutoLock: Why Cache Attacks on ARM Are Harder Than You Think | Green et al. | USENIX Security 2017 | 47
AutoLock

Cross-core Eviction

Core 0
L1-I
{...}
L1-D
c

Core 1
I
D

Core 2
I
D

Core 3
I
D
{...}
v
w

L2 Cache
(LLC)
+
AutoLock

prog_A{
  ...
}

RAM

prog_B{
  .......
}
AutoLock

Cross-core Eviction

Core 0
L1-I
{...}
L1-D
c
Core 1
I
D
Core 2
I
D
Core 3
I
D
{...}
v
w

L2 Cache (LLC) + AutoLock

prog_A{
    ...
}

RAM

prog_B{
    .......
}

AutoLock: Why Cache Attacks on ARM Are Harder Than You Think | Green et al. | USENIX Security 2017 | 49
AutoLock
Cross-core Eviction

Core 0
L1-I
\{...\}
L1-D
c

Core 1
I
D

Core 2
I
D

Core 3
I
D
\{...\}
x
w

L2 Cache (LLC) + AutoLock

prog_A{
  ...
}

RAM

prog_B{
  .......
}

Fraunhofer AISEC
AutoLock
Cross-core Eviction

AutoLock: Why Cache Attacks on ARM Are Harder Than You Think | Green et al. | USENIX Security 2017 | 51
AutoLock
Cross-core Eviction

```
Core 0
L1-I   L1-D
   {...}  c

Core 1
   I     D

Core 2
   I     D
   {...}  x

Core 3
   I     D

L2 Cache (LLC) + AutoLock

v 0
t 0
y 1
s 0
w 0
x 1
u 0
c 1

prog_A{
...
}

RAM

prog_B{
......
..}
```
AutoLock
Cross-core Eviction

Core 0
L1-I {...}
L1-D c

Core 1
I D

Core 2
I D

Core 3
I D
...
x
y

L2 Cache (LLC) + AutoLock

prog_A{
...
}

RAM

prog_B{
....... ...
}
AutoLock
Cross-core Eviction

![Diagram of AutoLock and Cross-core Eviction]

AutoLock: Why Cache Attacks on ARM Are Harder Than You Think | Green et al. | USENIX Security 2017 | 54
AutoLock
Cross-core Eviction

Core 0
L1-I
{...}
L1-D
c

Core 1
I
D
t
y

Core 2
I
D
s
w
x
u
c

Core 3
I
D
z
y

L2 Cache (LLC) + AutoLock

prog_A{
... 
}

RAM

prog_B{
....... 
}
AutoLock
Definition

A patented and undocumented performance feature of inclusive cache levels that transparently prevents the eviction of cache lines, if they are contained in higher cache levels.

Automatic + Lockdown = “AutoLock”
Implications of AutoLock
## Implications

### Impact in Theory

<table>
<thead>
<tr>
<th>Method</th>
<th>Task</th>
<th>Same-core</th>
<th>Cross-core</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evict + Time</td>
<td>Eviction</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Prime + Probe</td>
<td>Eviction</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Evict + Reload</td>
<td>Eviction</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Evict + Prefetch</td>
<td>Eviction</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Flush + *</td>
<td>Flush</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Implications
Impact in Practice

Qualcomm Krait 450

ARM Cortex-A57
## Implications
### SoC Evaluation

<table>
<thead>
<tr>
<th>Processor</th>
<th>System-on-Chip (SoC)</th>
<th>AutoLock</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARM Cortex-A7</td>
<td>Samsung Exynos 5422</td>
<td>✓</td>
</tr>
<tr>
<td>ARM Cortex-A15</td>
<td>Samsung Exynos 5422/5250</td>
<td>✓</td>
</tr>
<tr>
<td>ARM Cortex-A53</td>
<td>ARM Juno r0</td>
<td>✓</td>
</tr>
<tr>
<td>ARM Cortex-A57</td>
<td>ARM Juno r0</td>
<td>✓</td>
</tr>
<tr>
<td>Qualcomm Krait 450</td>
<td>Snapdragon 805</td>
<td>✗</td>
</tr>
</tbody>
</table>
## Implications

### Smartphone SoCs

<table>
<thead>
<tr>
<th>Manufacturer</th>
<th>SoC Family</th>
<th>% featuring A7,-15,-53,-57</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple</td>
<td>A10, A9, A8, A7</td>
<td>0</td>
</tr>
<tr>
<td>HiSilicon</td>
<td>Kirin 9xx, 6xx</td>
<td>86</td>
</tr>
<tr>
<td>Mediatek</td>
<td>MT67xx, MT659x/8x</td>
<td>100</td>
</tr>
<tr>
<td>Nvidia</td>
<td>Tegra X, K, 4</td>
<td>71</td>
</tr>
<tr>
<td>Qualcomm</td>
<td>Snapdragon 8xx, 6xx, 4xx</td>
<td>47</td>
</tr>
<tr>
<td>Samsung</td>
<td>Exynos 9, 8, 7, 5, 4</td>
<td>79</td>
</tr>
<tr>
<td>Xiaomi</td>
<td>Surge S</td>
<td>100</td>
</tr>
</tbody>
</table>
### Implications

**Previous Work on ARM**

<table>
<thead>
<tr>
<th>“ARMageddon”</th>
<th>“ROP Flush + Reload”</th>
<th>“TruSpy”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lipp et al. [LGS⁺16]</td>
<td>Zhang et al. [ZXZ16]</td>
<td>Zhang et al. [ZSS⁺16]</td>
</tr>
</tbody>
</table>

- ⇒ Device Selection
  - Qualcomm SoCs
  - Userspace flush
  - Cross-core eviction
- ⇒ Attack Selection
  - Flush + Reload
  - `cacheflush` syscall
  - No cross-core eviction
- ⇒ Device Properties
  - ARM Cortex-A8
  - Single-core setup
  - No cross-core eviction
AutoLock = Countermeasure?
Countermeasure?
Not Ultimately

**Vulnerable SoCs**
- ARM-compliant cores
- Userspace flush instr.

**Same-Core Attacks**
- ARM TrustZone
- Compromised OS

**Remote Evictions**
- Trigger self-evictions
- Increase load and wait time

**Redundant Targets**
- Attack multiple cache lines
- e.g. entire AES T-tables
Countermeasure?

Wait-a-minute Attack

- Attack by Irazoqui et al. [IIES14]
  - Observes usage of AES T-tables
  - One cache line per table
- Simple redundant variant
  - Observe all lines of all tables
  - Majority vote on derived keys
- Test environment
  - ARM Cortex-A15 with AutoLock
  - Full Linux operating system
Conclusion
AutoLock: undocumented feature of inclusive LLCs on ARM

Inhibits cross-core eviction and adversely affects attacks

Predominantly implemented in Cortex-A designs by ARM

Countermeasures are still necessary to protect against attacks
Questions?


