An Exploration of ARM System-Level Cache and GPU Side Channels

Patrick Cronin*, Xing Gao*, Haining Wang+, Chase Cotton*

University of Delaware*, Virginia Tech+
Computer Architecture – Then

• For many years laptop and desktops have been dominated by x86 while mobile devices are dominated by ARM
Sharing Too Much

- Apple has switched all of their new products to ARM based devices and Windows vendors are starting to follow suit
Sharing Too Much

• ARM processor architecture rapidly gaining popularity and acceptance in consumer systems
  • Provides new vectors and easier access to previously x86 only side channel attacks
  • Examine whether same mistakes from previous systems carry over to new ARM devices
Attacking CPUs – Cache Side Channels

• Computer systems operate on memory
• Memory accesses can be very slow
• Many operations are in a pattern or predictable
Attacking CPUs – Cache Side Channels

• Computer systems operate on memory
• Memory accesses can be very slow
• Many operations are in a pattern or predictable
Attacking CPUs – Cache Side Channels

• Computer systems operate on memory
• Memory accesses can be very slow
• Many operations are in a pattern or predictable
Atacking CPUs – Cache Side Channels

• Computer systems operate on memory
• Memory accesses can be very slow
• Many operations are in a pattern or predictable
Attacking CPUs – Cache Side Channels

• Computer systems operate on memory
• Memory accesses can be very slow
• Many operations are in a pattern or predictable
Attacking CPUs – Cache Side Channels

• Computer systems operate on memory
• Memory accesses can be very slow
• Many operations are in a pattern or predictable
Attacking CPUs – Cache Side Channels

• Computer systems operate on memory
• Memory accesses can be very slow
• Many operations are in a pattern or predictable
Attacking CPUs – Cache Side Channels

• Computer systems operate on memory
• Memory accesses can be very slow
• Many operations are in a pattern or predictable
Attacking CPUs – Cache Side Channels

• Caches exploit the patterns in memory access
• Increase speed of the system at reasonable cost

- L1: Smallest, fastest cache level - $$$
- L2: Medium cache level - $$
- L3: Largest, slowest cache level - $

12
Revisting x86 - Cache Occupancy Channel

- [7] suggests a cache occupancy channel can be utilized to fingerprint websites and study this in x86
- The spy claims the entire cache and times how long it takes to access. As the victim runs, the cache is impacted and a timing feature can be extracted

Website Fingerprinting Attack – Process
How is ARM Different?

• x86 processors utilize straightforward cache design
How is ARM Different?

• ARM employs DynamIQ architecture and vastly different cache strategies w/ Integrated Accelerators
Adjusting the Attack for ARM

• ARM has heterogeneous processors which run at different frequencies
• ARM caches are designed with different algorithms than their x86 counterparts
Adjusting x86 Attacks to ARM – Core Types

• ARM SoC can contain multiple different core types

<table>
<thead>
<tr>
<th>Buffer Size</th>
<th>Access Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1MB</td>
<td></td>
</tr>
<tr>
<td>1MB</td>
<td></td>
</tr>
<tr>
<td>1MB</td>
<td></td>
</tr>
<tr>
<td>1MB</td>
<td></td>
</tr>
<tr>
<td>1MB</td>
<td></td>
</tr>
<tr>
<td>1MB</td>
<td></td>
</tr>
<tr>
<td>1MB</td>
<td></td>
</tr>
</tbody>
</table>

Low Power

- core [0]
  - L1I
  - L1D

High Power

- core [0]
  - L1I
  - L1D
Adjusting x86 Attacks to ARM – Core Types

- ARM Schedulers take advantage of High and Low power cores

- 10x difference in access speed on iPhone SE2 with foreground vs background web tab
- Differently shaped cache activity
- Caused by energy aware scheduler moving background tab to low cores
Adjusting x86 Attacks for ARM – Browsers

• Each browser has its own JavaScript engine and memory management

Buffer size must be carefully chosen
Adjusting x86 Attacks for ARM – Timing

• Constant war between high frequency sampling and access time
• Careful balancing act
  • Too Slow – won’t sample often enough
  • Too Fast – long downtime between samples
Adjusting x86 Attacks for ARM - Timing

• Invert measurement pattern
• Measure the number of accesses in the time period
• High granularity measurement always!

![Diagram showing x86 and Ours with 1ms timing and 11 accesses / 1ms]
Adjusting x86 Attacks for ARM – Invert

- Major Drawback
  - Exclusive caching

<table>
<thead>
<tr>
<th></th>
<th>Inclusive</th>
<th>Exclusive</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Access 1
Access 2
Access 3
Access 4
Access 5
Access 6
Adjusting x86 Attacks for ARM – Invert

- Major Drawback
  - Exclusive caching

<table>
<thead>
<tr>
<th></th>
<th>Inclusive</th>
<th>Exclusive</th>
<th>Access 1</th>
<th>Access 2</th>
<th>Access 3</th>
<th>Access 4</th>
<th>Access 5</th>
<th>Access 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L3</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Adjusting x86 Attacks for ARM – Invert

- Major Drawback
  - Exclusive caching

```
<table>
<thead>
<tr>
<th></th>
<th>Inclusive</th>
<th>Exclusive</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td>1 2</td>
<td>1 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L2</td>
<td>1 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L3</td>
<td>1 2</td>
<td></td>
</tr>
</tbody>
</table>
```

Access 1
Access 2
Access 3
Access 4
Access 5
Access 6
Adjusting x86 Attacks for ARM – Invert

• Major Drawback
  • Exclusive caching

<table>
<thead>
<tr>
<th>Inclusive</th>
<th>Exclusive</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td>1 2 3</td>
</tr>
<tr>
<td>L2</td>
<td>1 2 3</td>
</tr>
<tr>
<td>L3</td>
<td>1 2 3</td>
</tr>
</tbody>
</table>

Access 1  Access 2  Access 3  Access 4  Access 5  Access 6
Adjusting x86 Attacks for ARM – Invert

• Major Drawback
  • Exclusive caching

<table>
<thead>
<tr>
<th>Inclusive</th>
<th>Exclusive</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td><img src="image1" alt="L1 Inclusive and Exclusive" /></td>
</tr>
<tr>
<td>L2</td>
<td><img src="image2" alt="L2 Inclusive and Exclusive" /></td>
</tr>
<tr>
<td>L3</td>
<td><img src="image3" alt="L3 Inclusive and Exclusive" /></td>
</tr>
</tbody>
</table>

Access 1
Access 2
Access 3
Access 4
Access 5
Access 6
Adjusting x86 Attacks for ARM – Invert

- Major Drawback
  - Exclusive caching

<table>
<thead>
<tr>
<th></th>
<th>Inclusive</th>
<th>Exclusive</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td>5 2 3 4</td>
<td>5 2 3 4</td>
</tr>
<tr>
<td></td>
<td>1 2 3 4</td>
<td>1</td>
</tr>
<tr>
<td>L2</td>
<td>1 2 3 4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>L3</td>
<td>1 2 3 4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

Access 1  Access 2  Access 3  Access 4  Access 5  Access 6
### Adjusting x86 Attacks for ARM – Invert

- **Major Drawback**
  - Exclusive caching

<table>
<thead>
<tr>
<th></th>
<th>Inclusive</th>
<th></th>
<th>Exclusive</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td>5 6 3 4</td>
<td></td>
<td>5 6 3 4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L2</td>
<td>1 2 3 4</td>
<td>5 6</td>
<td>1 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L3</td>
<td>1 2 3 4</td>
<td>5 6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Access: 1 2 3 4 5 6
Adjusting x86 Attacks for ARM – Invert

- **Major Drawback**
  - Exclusive caching
- **Exclusive caching**
  - mainly for design density
- If we size our buffer incorrectly, we won’t affect the cache!

<table>
<thead>
<tr>
<th>Inclusive</th>
<th>Exclusive</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td>L1</td>
</tr>
<tr>
<td>5 6 3 4</td>
<td>5 6 3 4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Access 1</th>
<th>Access 2</th>
<th>Access 3</th>
<th>Access 4</th>
<th>Access 5</th>
<th>Access 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>L2</td>
<td>L2</td>
<td>L2</td>
<td>L2</td>
<td>L2</td>
<td>L2</td>
</tr>
<tr>
<td>1 2 3 4</td>
<td>1 2 3 4</td>
<td>1 2 3 4</td>
<td>1 2 3 4</td>
<td>1 2 3 4</td>
<td>1 2 3 4</td>
</tr>
<tr>
<td>5 6</td>
<td>5 6</td>
<td>5 6</td>
<td>5 6</td>
<td>5 6</td>
<td>5 6</td>
</tr>
</tbody>
</table>
Website Fingerprinting Attack

• Closed World
  • Only test against sensitive websites

• Open World
  • Try to identify sensitive websites from many websites

<table>
<thead>
<tr>
<th>Closed World Experiments</th>
</tr>
</thead>
<tbody>
<tr>
<td>• 100 Accesses to top 100 Websites</td>
</tr>
<tr>
<td>• Randomize Access Order to Ensure Fairness</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Open World Experiments</th>
</tr>
</thead>
<tbody>
<tr>
<td>• 100 Accesses to top 100 Websites</td>
</tr>
<tr>
<td>• 1 Access to 5,000 other Websites</td>
</tr>
<tr>
<td>• Randomize Access Order to Ensure Fairness</td>
</tr>
</tbody>
</table>
## Results – Web-Based

<table>
<thead>
<tr>
<th>Device</th>
<th>CPU</th>
<th>Browser</th>
<th>Closed World Ridge Regression</th>
<th>CNN</th>
<th>Open World Ridge Regression</th>
<th>CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Macbook Air</td>
<td>Apple M1</td>
<td>Chrome 89</td>
<td>95.6</td>
<td>92.2</td>
<td>88.1</td>
<td>89.8</td>
</tr>
<tr>
<td>Macbook Air</td>
<td>Apple M1</td>
<td>Safari 14</td>
<td>94.3</td>
<td>89.4</td>
<td>78.4</td>
<td>85.1</td>
</tr>
<tr>
<td>Macbook Air</td>
<td>Apple M1</td>
<td>Firefox 88</td>
<td>88.1</td>
<td>83.9</td>
<td>68.2</td>
<td>77.8</td>
</tr>
<tr>
<td>iPhone SE2</td>
<td>Apple A13</td>
<td>Safari 14</td>
<td>80.2</td>
<td>75.7</td>
<td>65.8</td>
<td>72.7</td>
</tr>
<tr>
<td>iPhone SE2</td>
<td>Apple A13</td>
<td>Chrome 90</td>
<td>80.2</td>
<td>75.9</td>
<td>65.0</td>
<td>73.3</td>
</tr>
<tr>
<td>Google Pixel 3</td>
<td>Snapdragon 845</td>
<td>Chrome 90</td>
<td>88.0</td>
<td>81.8</td>
<td>66.0</td>
<td>75.9</td>
</tr>
</tbody>
</table>
Crafting Another Contention Channel

• The dynamic shared unit interacts with multiple peripherals on the device
• Web content is hardware accelerated by GPU
• Can the GPU act as another channel?
Accessing the GPU from JavaScript

- WebGL/WebGL2
  - Animations, video, 3D experiences
  - Focused on *visually* – 60Hz
- WebGPU
  - Updates WebGL for computing
  - Supported in beta
- GPU.js
  - Allows quick creation of compute kernels
GPU Contention Challenges

• How do we measure GPU Contention?
• How do we create GPU Contention?
Measuring GPU Contention

• Cannot interrupt GPU kernel to check time
  • Browser developers removed timing ability due to exploits
• Time completions of kernel instead of interrupting kernel
  • Better granularity if we have very short kernel
Creating GPU Contention

- Matrix Multiplication
  - Very computation heavy
- Dot product
  - Lower complexity, but still lots of multiplication
- Sum array row
  - Minimal complexity
  - Access each element only once
GPU Contention Channel Results

<table>
<thead>
<tr>
<th>Device</th>
<th>GPU</th>
<th>Browser</th>
<th>Closed World Ridge Regression</th>
<th>CNN</th>
<th>Open World Ridge Regression</th>
<th>CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Macbook Air</td>
<td>Apple 7 Core</td>
<td>Chrome 89</td>
<td>90.5</td>
<td>85.3</td>
<td>76.6</td>
<td>81.4</td>
</tr>
<tr>
<td>Android</td>
<td>Adreno 630</td>
<td>Chrome 89</td>
<td>88.2</td>
<td>82.6</td>
<td>67.6</td>
<td>77.3</td>
</tr>
</tbody>
</table>

Better performance on the Google Pixel 3!
Contestation – Summary

• Examined 2 contention channels in ARM based devices
• Investigate how the different scheduling of heterogeneous core operating systems effects contention channels
  • Shared cache contention channel demonstrated up to 89% accurate open world attack
  • Novel GPU contention channel performed up to 2% better than cache contention channel on Android open world
Questions?