RELOAD+REFRESH: Abusing Cache Replacement Policies to Perform Stealthy Cache Attacks

Samira Briongos\textsuperscript{1}, Pedro Malagón Marzo\textsuperscript{1}, José M. Moya Fernández\textsuperscript{1}, Thomas Eisenbarth\textsuperscript{2}

\textsuperscript{1}Universidad Politécnica de Madrid, \textsuperscript{2}University of Lübeck
Motivation

• Can we trust existing countermeasures against cache attacks?
  ▪ Hardware countermeasures
  ▪ Preemptive countermeasures
  ▪ Detection countermeasures
Motivation

• Can we trust existing countermeasures against cache attacks?
  ▪ Hardware countermeasures
  ▪ Preemptive countermeasures
  ▪ Detection countermeasures
Motivation

• Can we trust existing countermeasures against cache attacks?
  ▪ Hardware countermeasures
  ▪ Preemptive countermeasures
  ▪ Detection countermeasures

• How do they detect cache attacks?
  ▪ Collect information from performance counters (cache misses, cache accesses…)
  ▪ Compute behavioral models of the process
Motivation

• Can we trust existing countermeasures against cache attacks?
  ▪ Hardware countermeasures
  ▪ Preemptive countermeasures
  ▪ **Detection countermeasures**

• How do they detect cache attacks?
  ▪ Collect information from performance counters (cache misses, cache accesses…)
  ▪ Compute behavioral models of the process

• Could an attacker gain information without forcing misses?
  ▪ It is possible if the eviction order depends on the actual order utilization of the data
LLC Replacement policy

• The algorithm that decides which element has to be evicted from the cache in case of conflict.
• We focus on the L3 cache
  ▪ Shared
  ▪ Undocumented (Quad-Age LRU)
LLC Replacement policy

- The algorithm that decides which element has to be evicted from the cache in case of conflict.
- We focus on the L3 cache
  - Shared
  - Undocumented
- Idea:
  - We know the location of each block of data in the cache
  - We can enforce the order of the accesses to the data in the cache
  - We can measure access times, i.e. cache hits and misses
LLC Replacement policy: Test

- Manually define a replacement policy to test.
- Generate two eviction sets.
- Fill the cache set with the data in one of the eviction sets.
LLC Replacement policy: Test

- Manually define a replacement policy to test.
- Generate two eviction sets.
- Fill the cache set with the data in one of the eviction sets.
- Randomly access the data previously placed into the cache.
LLC Replacement policy: Test

- Manually define a replacement policy to test.
- Generate two eviction sets.
- Fill the cache set with the data in one of the eviction sets.
- Randomly access the data previously placed into the cache.
- Force a miss by accessing one element in the conflicting set.
- Check if the evicted block corresponds with the block predicted by the test policy.
LLC Replacement policy

- Data is linearly inserted in the cache set.
LLC Replacement policy

- Data is linearly inserted in the cache set.
- Only access to LLC update the ages of the blocks.
LLC Replacement policy

- Data is linearly inserted in the cache set.
- Only access to LLC update the ages of the blocks.
- Two different policies.
  - Adaptive Replacement Policy (4th and 5th generations).
  - Apparently fixed Replacement Policy (6th, 7th and 8th generations).
LLC Replacement policy

- Data is linearly inserted in the cache set.
- Only access to LLC update the ages of the blocks.
- Two different policies.
  - Adaptive Replacement Policy (4th and 5th generations).
  - Apparently fixed Replacement Policy (6th, 7th and 8th generations).
- The replacement policy is the same, evict blocks with age 3.
  - Differ in the insertion age (2 or 3).
LLC Replacement policy

D Request

Is D in the cache?
LLC Replacement policy

D Request

Is D in the cache?

Yes

Is it in L1 or L2?
LLC Replacement policy

D Request

Is D in the cache? Yes → Is it in L1 or L2? Yes → Return D
LLC Replacement policy

D Request

Is D in the cache? → Yes → Is it in L1 or L2? → Yes → Return D

No → Return D

Decrease its age
LLC Replacement policy

D Request

Is D in the cache?

Yes

Is it in L1 or L2?

Yes

Return D

No

Fetch D from main memory
Place D into the cache

Is there any empty block in the cache set?

No

Return D

Decrease its age
LLC Replacement policy

D Request

Is D in the cache? Yes

Is it in L1 or L2? Yes

Return D

Return D

Decrease its age

Is there any empty block in the cache set? Yes

Place D in the first one

Set the D age to its insertion age
LLC Replacement policy

D Request

Is D in the cache?
- Yes: Is it in L1 or L2?
  - Yes: Return D
  - No: Return D
  - Decrease its age

- No: Fetch D from main memory
  - Place D into the cache

Is there any empty block in the cache set?
- Yes: Place D in the first one
  - Set the D age to its insertion age

- No: Is there any block whose age is equal to 3?
LLC Replacement policy

D Request

- Is D in the cache?
  - Yes
    - Is it in L1 or L2?
      - Yes
        - Return D
      - No
        - Return D
        - Decrease its age
  - No
    - Fetch D from main memory
    - Place D into the cache

- Is there any empty block in the cache set?
  - Yes
    - Place D in the first one
    - Set the D age to its insertion age
  - No
    - Is there any block whose age is equal to 3?
      - Yes
      - Increase the ages of all the blocks
      - No
      - Increase the ages of all the blocks
LLC Replacement policy

D Request

Is D in the cache?

Yes

Is it in L1 or L2?

Yes

Return D

No

Fetch D from main memory
Place D into the cache

Is there any empty block in the cache set?

Yes

Place D in the first one

No

Set the D age to its insertion age

Is there any block whose age is equal to 3?

Yes

Replace the first block with age 3 with D

No

Increase the ages of all the blocks

No

Return D

Decrease its age
• Retrieve information without forcing misses on the victim
• Retrieve information without forcing misses on the victim
• Abuse the replacement policy:
  ▪ Place data in the L3 cache in the desired position and set its age
  ▪ Observe changes in the eviction candidate.
• Retrieve information without forcing misses on the victim
• **Abuse the replacement policy:**
  - Place data in the L3 cache in the desired position and set its age
  - Observe changes in the eviction candidate.
• **Assume:**
  - There is shared memory
  - The attacker can construct an eviction set
  - The attacker can ensure the order of the accesses
### RELOAD+REFRESH: stages

1. Prepare the cache

<table>
<thead>
<tr>
<th>Content</th>
<th>Age</th>
<th>Candidate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
RELOAD+REFRESH: stages

1. Prepare the cache

<table>
<thead>
<tr>
<th>Content</th>
<th>Target</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age</td>
<td></td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
2. Wait for the victim to access the data

<table>
<thead>
<tr>
<th>Content Age</th>
<th>Target</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
2. Wait for the victim to access the data

<table>
<thead>
<tr>
<th>Content</th>
<th>Target</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

RELOAD+REFRESH: Abusing Cache Replacement Policies to Perform Stealthy Cache Attacks
2. Wait for the victim to access the data

<table>
<thead>
<tr>
<th>Content</th>
<th>Target</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>ACCESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
RELOAD+REFRESH: stages

2. Wait for the victim to access the data

<table>
<thead>
<tr>
<th>Content</th>
<th>Target</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Candidate

<table>
<thead>
<tr>
<th>Content</th>
<th>Target</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

ACCESS

NO ACCESS
### 3. Retrieve the information (Force miss + Reload)

<table>
<thead>
<tr>
<th>Content</th>
<th>Target</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age</td>
<td></td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

**ACCESS**

<table>
<thead>
<tr>
<th>Content</th>
<th>Target</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age</td>
<td></td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

**NO ACCESS**
# RELOAD+REFRESH: stages

## 3. Retrieve the information (Force miss + Reload)

<table>
<thead>
<tr>
<th>Content Age</th>
<th>Target</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

**Candidate**

**ACCESS**

<table>
<thead>
<tr>
<th>Content Age</th>
<th>Target</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

**Candidate**

**NO ACCESS**
### RELOAD+REFRESH: stages

#### 3. Retrieve the information (Force miss + Reload)

<table>
<thead>
<tr>
<th>Content</th>
<th>Target</th>
<th>H</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

**ACCESS**

<table>
<thead>
<tr>
<th>Content</th>
<th>H</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

**NO ACCESS**
### 3. Retrieve the information (Force miss + Reload)

<table>
<thead>
<tr>
<th>Content</th>
<th>Target</th>
<th>H</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age</td>
<td></td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

**Candidate**

<table>
<thead>
<tr>
<th>Content</th>
<th>H</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

**ACCESS**

**NO ACCESS**
3. Retrieve the information (Force miss + Reload)

<table>
<thead>
<tr>
<th>Content</th>
<th>Target</th>
<th>H</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

Candidate ACCESS

<table>
<thead>
<tr>
<th>Content</th>
<th>H</th>
<th>Target</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

Candidate NO ACCESS
### RELOAD+REFRESH: stages

#### 3. Retrieve the information (Force miss + Reload)

<table>
<thead>
<tr>
<th>Content</th>
<th>Target</th>
<th>H</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

**ACCESS**

<table>
<thead>
<tr>
<th>Content</th>
<th>Target</th>
<th>H</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

**NO ACCESS**
## RELOAD+REFRESH: stages

### 4. Refresh the cache set state

<table>
<thead>
<tr>
<th>Content</th>
<th>Target</th>
<th>H</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

ACCESS

<table>
<thead>
<tr>
<th>Content</th>
<th>H</th>
<th>Target</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

NO ACCESS
4. Refresh the cache set state

<table>
<thead>
<tr>
<th>Content</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

**Candidate**

**ACCESS**

<table>
<thead>
<tr>
<th>Content</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

**Candidate**

**NO ACCESS**
## RELOAD+REFRESH: stages

### 4. Refresh the cache set state

<table>
<thead>
<tr>
<th>Age</th>
<th>Target</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

**Candidate**

<table>
<thead>
<tr>
<th>Age</th>
<th>Target</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

**ACCESS**

<table>
<thead>
<tr>
<th>Age</th>
<th>Target</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

**NO ACCESS**
### 4. Refresh the cache set state

<table>
<thead>
<tr>
<th>Content</th>
<th>Target</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age</td>
<td></td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACCESS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Content</th>
<th>Target</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age</td>
<td></td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NO ACCESS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
4. Refresh the cache set state

<table>
<thead>
<tr>
<th>Content Age</th>
<th>Target</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Candidate

<table>
<thead>
<tr>
<th>Content Age</th>
<th>Target</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

ACCESS

<table>
<thead>
<tr>
<th>Content Age</th>
<th>Target</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Candidate

<table>
<thead>
<tr>
<th>Content Age</th>
<th>Target</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

NO ACCESS
## 4. Refresh the cache set state

<table>
<thead>
<tr>
<th>Content</th>
<th>Target</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

- **Candidate**

<table>
<thead>
<tr>
<th>Content</th>
<th>Target</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

**ACCESS**

**NO ACCESS**
RELOAD+REFRESH: Evaluation

- Covert channel
  - Similar performance to existing approaches.
  - Slightly lower resolution

F-Score for the different approaches: Reload+Refresh (R+R), Flush+Reload (F+R) and Prime+Probe (P+P).

<table>
<thead>
<tr>
<th>Window length (ns)</th>
<th>50000</th>
<th>10000</th>
<th>1000</th>
<th>750</th>
<th>500</th>
<th>250</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>R+R</strong></td>
<td>0.988</td>
<td>0.975</td>
<td>0.925</td>
<td>0.684</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>F+R</strong></td>
<td>0.999</td>
<td>0.995</td>
<td>0.996</td>
<td>0.991</td>
<td>0.989</td>
<td>0.981</td>
</tr>
<tr>
<td><strong>P+P</strong></td>
<td>0.934</td>
<td>0.934</td>
<td>0873</td>
<td>0.716</td>
<td>0.548</td>
<td>-</td>
</tr>
</tbody>
</table>
RELOAD+REFRESH: Evaluation

• AES
  ▪ Retrieve the secret key

<table>
<thead>
<tr>
<th>Attack</th>
<th>R+R</th>
<th>F+R</th>
<th>P+P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Samples</td>
<td>3800</td>
<td>3500</td>
<td>3900</td>
</tr>
<tr>
<td>F-Score</td>
<td>0.98</td>
<td>0.99</td>
<td>0.97</td>
</tr>
</tbody>
</table>

• No cache misses
RELOAD+REFRESH: Evaluation

• AES
  ▪ Retrieve the secret key

<table>
<thead>
<tr>
<th>Attack</th>
<th>R+R</th>
<th>F+R</th>
<th>P+P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Samples</td>
<td>3800</td>
<td>3500</td>
<td>3900</td>
</tr>
<tr>
<td>F-Score</td>
<td>0.98</td>
<td>0.99</td>
<td>0.97</td>
</tr>
</tbody>
</table>

Reload+Refresh (R+R)
Flush+Reload (F+R)
Prime+Probe (P+P)

▪ No cache misses
RELOAD+REFRESH: Evaluation

- **RSA**
  - Multiply operations

<table>
<thead>
<tr>
<th>Attack</th>
<th>R+R</th>
<th>F+R</th>
<th>P+P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>96.1%</td>
<td>98.6%</td>
<td>95.4%</td>
</tr>
<tr>
<td>F-Score</td>
<td>0.952</td>
<td>0.99</td>
<td>0.945</td>
</tr>
</tbody>
</table>

- No cache misses
• Results of monitoring different performance counters

1-AES Normal  2-AES R+R  3-AES F+R  4-AES P+P  5-RSA Normal  6-RSA R+R  7-RSA F+R  8-RSA P+P
RELOAD+REFRESH: Detection

• Results of monitoring different performance counters

1 – AES Normal
2 – AES R+R
3 – AES F+R
4 – AES P+P
5 – RSA Normal
6 – RSA R+R
7 – RSA F+R
8 – RSA P+P

RELOAD+REFRESH: Abusing Cache Replacement Policies to Perform Stealthy Cache Attacks
Conclusions

• **Show an important limitation in detection algorithms**
  • The number of counters is limited
  • RELOAD+REFRESH hides misses from the monitor
  • Cat and mouse game
Conclusions

• **Show an important limitation in detection algorithms**
  • The number of counters is limited
  • RELOAD+REFRESH hides misses from the monitor
  • Cat and mouse game

• **Provide further insights into the cache architecture**
  ▪ Reverse-engineer the replacement policy
  ▪ Other cache attacks could benefit from this knowledge
Thank you for your attention

Q&A

Contact information: samirabriongos@die.upm.es
Sample code: https://github.com/greenlsi/reload_refresh