Time Protection: The Missing OS Abstraction

Qian Ge, Yuval Yarom, Tom Chothia, Gernot Heiser
qian.ge@data61.csiro.au
SPECTRE

Top secret
Enforcing time protection at the OS level

Sending information through timing
Enforcing time protection at the OS level

Sending information through timing
Enforcing time protection at the OS level

Sending information through timing
Microarchitectural Timing Channels

Security Domain

Security Domain

Shared hardware caches
Microarchitectural Timing Channels

Welcome to Dresden

Security Domain

Security Domain

Shared hardware caches
Microarchitectural Timing Channels

Welcome to Dresden

Security Domain

Shared hardware caches
Welcome to Dresden
Welcome to Dresden

Shared hardware caches

Fast Fast Slow Fast Fast
..... S F S FFF SSSS F

Not English, but our protocol...
Contention, Contention, Contention,…

• Contention leaks information via timing
• Caches:
  • capacity-limited
  • stateful
• Resulting on temporal interference during:
  - time-shared access
  - concurrent access

Shared hardware caches
Contention, Contention, Contention...

- Contention leaks information via timing
- Caches:
  - capacity-limited
  - stateful
- Resulting on temporal interference during:
  - time-shared access
  - concurrent access

Any state-holding microarchitectural feature:
- Caches, branch predictor, TLB
Preventing Contention by Partitioning

Partitioned caches
Spatial Partitioning

Cache colouring:
- Distributing coloured cache sets to coloured memory frames
- Memory management policy
Spatial Partitioning

Cache colouring:
- Distributing coloured cache sets to coloured memory frames
- Memory management policy

Partition security domains on disjoint cache sets

Shared hardware caches
Spatial Partitioning

Cache colouring:
- Distributing coloured cache sets to coloured memory frames
- Memory management policy

Partition security domains on disjoint cache sets

• Resulting on temporal interference during:
  - time-shared access
  - concurrent access
Spatial Partitioning

Cache colouring:
- Distributing coloured cache sets to coloured memory frames
- Memory management policy

Partition security domains on disjoint cache sets

- Resulting on temporal interference during:
  - time-shared access
  - concurrent access

Shared hardware caches
Spatial Partitioning

Cache colouring:
- Distributing coloured cache sets to coloured memory frames
- Memory management policy

- Resulting on temporal interference during:
  - time-shared access
  - concurrent access

Partition security domains on disjoint cache sets

Shared hardware caches

Cannot be supported by on-core caches
Temporal Partitioning

Flushing on-core caches:
- Resetting states

Context switch

Shared on-caches
Temporal Partitioning

Context switch

Shared on-caches

Flushing on-core caches:
• Resetting states

• Resulting on temporal interference during:
  - time-shared access
  - concurrent access
Temporal Partitioning

Flushing on-core caches:
- Resetting states

- Resulting on temporal interference during:
  - time-shared access
  - concurrent access
Preventing Temporal Interference through Partitioning

Temporal partitioning

Spatial partitioning
Wait, Everyone Shares the Kernel....

Security Domain

Security Domain

Kernel Services

A shared partition

Shared hardware caches
Wait, Everyone Shares the Kernel....

Cache lines used by the kernel collide with Spy’s partition

A shared partition

Shared hardware caches
Wait, Everyone Shares the Kernel....

Poster Session @ 17:15

Cache lines used by the kernel collide with Spy’s partition

A shared partition

Shared hardware caches
Wait, Everyone Shares the Kernel....

- Cache lines used by the kernel collide with Spy’s partition
- Fast Fast Slow Fast Fast
- A shared partition

Poster Session @ 17:15

Kernel Services

Shared hardware caches
Cloning the Kernel Image

Analysing the kernel sections

```
.text
.rodata
.data
```
Cloning the Kernel Image

Analysing the kernel sections

.text
.rodata
.data

Duplicated sections

.text
.rodata
.data
Global (9KiB)

Maintaining coherency
Cloning the Kernel Image

Analysing the kernel sections

Kernel Clone: Generating a copy of the kernel image with user-level managed memory
Cloning the Kernel Image

Analysing the kernel sections

Global (9KiB)

Duplicated sections

Maintaining coherency

Kernel Clone: Generating a copy of the kernel image with user-level managed memory
Dedicated Kernel Images

Security Domain

Security Domain

Shared Global Data

Temporal partitioning

Deterministic usage

Spatial partitioning
Timing Channel through the Shared Kernel

Channel matrix: conditional probability of observing the output signal (time, spy) given the input signal (system-call number, Trojan)

Raw channel

Horizontal variation indicates a channel
Timing Channel through the Shared Kernel

Channel matrix: conditional probability of observing the output signal (time, spy) given the input signal (system-call number, Trojan)

Raw channel

Horizontal variation indicates a channel

Prevented by cloned kernel
How realistic is cloning?

<table>
<thead>
<tr>
<th>Arch</th>
<th>seL4 Clone</th>
<th>Linux fork + exec</th>
</tr>
</thead>
<tbody>
<tr>
<td>x86</td>
<td>79 μs</td>
<td>257 μs</td>
</tr>
<tr>
<td>Arm</td>
<td>608 μs</td>
<td>4,300 μs</td>
</tr>
</tbody>
</table>

Memory consumption

- x86: 224 KiB
- Arm: 120 KiB

Efficiency

.text
.rodata
.data
Global (9KiB)
Performance Impact

The cost of colouring a domain:

- 50% of the cache colour
- 50% of the cache colour + a coloured kernel

The slowdown of splash-2 against base line kernel
Performance Impact

The cost of colouring a domain:
- 50% of the cache colour
- 50% of the cache colour + a coloured kernel

The slowdown of splash-2 against base line kernel

The cost of time protection:
- spatial partition + temporal partition

<table>
<thead>
<tr>
<th></th>
<th>x86</th>
<th>Arm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>3.38%</td>
<td>1.09%</td>
</tr>
</tbody>
</table>

The geometric mean
We have also done:

- Partition IRQs
- Domain-switch actions
- A deterministic domain switching latency
- Preventing timing channels:
  - Shared kernel, Intra-core, cross-core, domain-switch latency, timer
- Performance evaluation:
  - IPC, domain switching latency, kernel cloning, cache colouring, time protection
Summary

• Define temporal isolation as a mandatory (black-box mechanisms) enforcement provided by the OS.
• Time protection for preventing microarchitectural timing channels
• Effective on closing studied timing channels
• Low overhead
THANK YOU

Trustworthy Systems
Qian Ge
PhD candidate

e  qian.ge@data61.csiro.au

www.data61.csiro.au