IOTLB-SC: An Accelerator-Independent Leakage Source in Modern Cloud Systems

Thore Tiemann [1], Zane Weissman [2], Thomas Eisenbarth [1], Berk Sunar [2]

[1] University of Lübeck, Lübeck, Germany

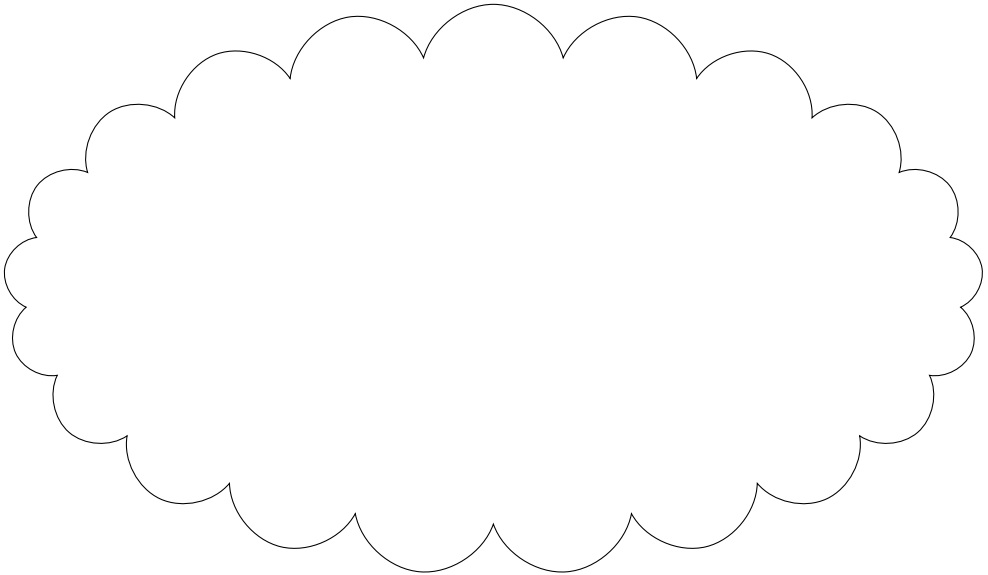
[2] Worcester Polytechnic Institute, MA, USA

ASIA CCS 2023, July 10–14 2023

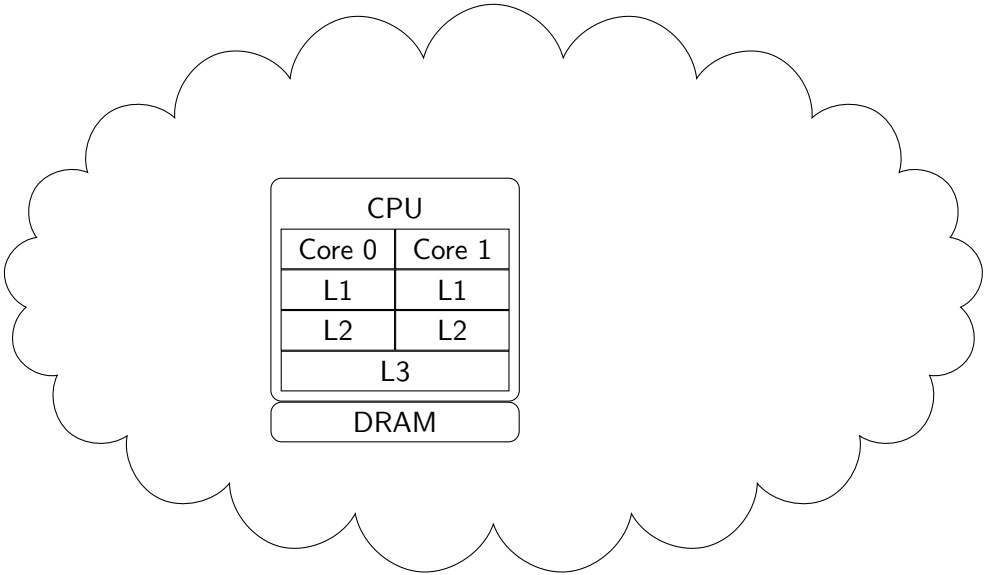
UNIVERSITÄT ZU LÜBECK
INSTITUTE FOR IT SECURITY

WPI

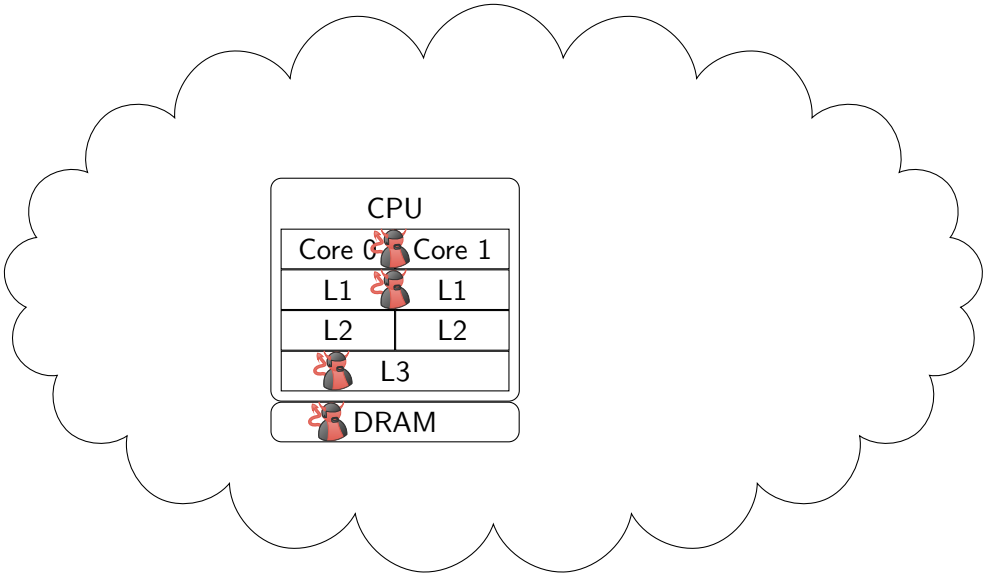
Motivation



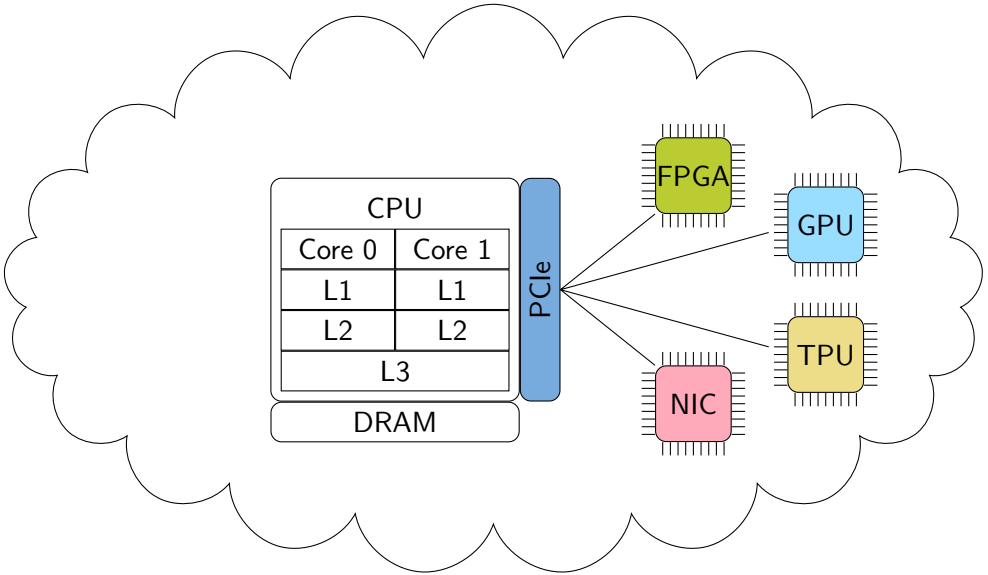
Motivation



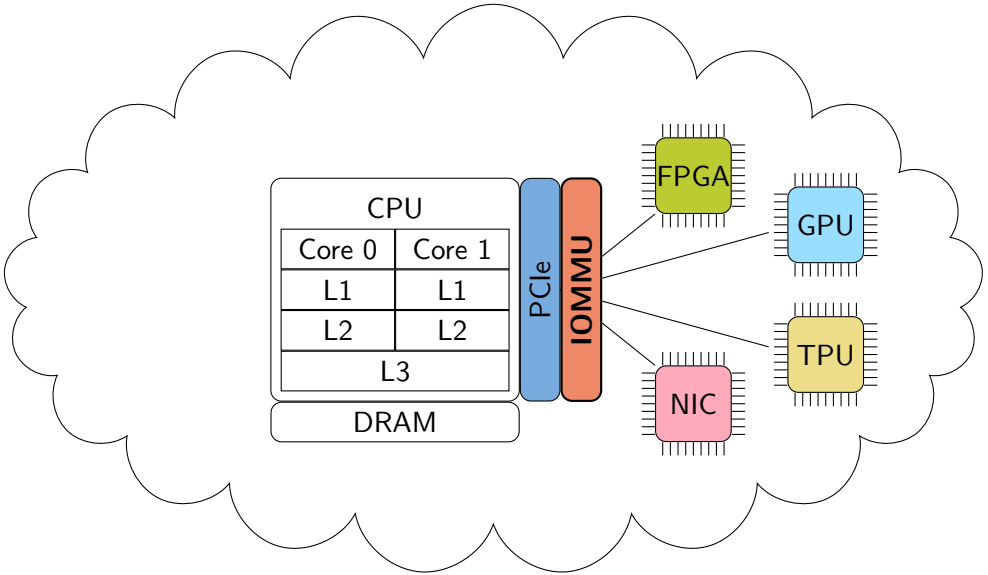
Motivation



Motivation



Motivation



Motivation

Research Question

Do IOTLBs introduce a side-channel?

IOTLB Side-channels

Problem

- IOMMU inaccessible from CPU

IOTLB Side-channels

Problem

- IOMMU inaccessible from CPU
- **Solution**: FPGA design to carry out our experiments

IOTLB Side-channels

Results

IOMMU disabled

- DMA read: 160–185 cycles

R. Neugebauer et al., "Understanding PCIe performance for end host networking", SIGCOMM, 2018

C. Peglow, "Security analysis of hybrid Intel CPU/FPGA platforms using IOMMUs against I/O attacks", Thesis, University of Lübeck, 2020

IOTLB Side-channels

Results

IOMMU disabled

- DMA read: 160–185 cycles

IOMMU enabled

R. Neugebauer et al., "Understanding PCIe performance for end host networking", SIGCOMM, 2018

C. Peglow, "Security analysis of hybrid Intel CPU/FPGA platforms using IOMMUs against I/O attacks", Thesis, University of Lübeck, 2020

IOTLB Side-channels

Results

IOMMU disabled

- DMA read: 160–185 cycles

IOMMU enabled

- First DMA read: 225–270 cycles

R. Neugebauer et al., "Understanding PCIe performance for end host networking", SIGCOMM, 2018

C. Peglow, "Security analysis of hybrid Intel CPU/FPGA platforms using IOMMUs against I/O attacks", Thesis, University of Lübeck, 2020

IOTLB Side-channels

Results

IOMMU disabled

- DMA read: 160–185 cycles

IOMMU enabled

- First DMA read: 225–270 cycles
- Next accesses: 160–185 cycles

R. Neugebauer et al., "Understanding PCIe performance for end host networking", SIGCOMM, 2018

C. Peglow, "Security analysis of hybrid Intel CPU/FPGA platforms using IOMMUs against I/O attacks", Thesis, University of Lübeck, 2020

IOTLB Side-channels

Results

IOMMU disabled

- DMA read: 160–185 cycles

IOMMU enabled

- First DMA read: 225–270 cycles
- Next accesses: 160–185 cycles

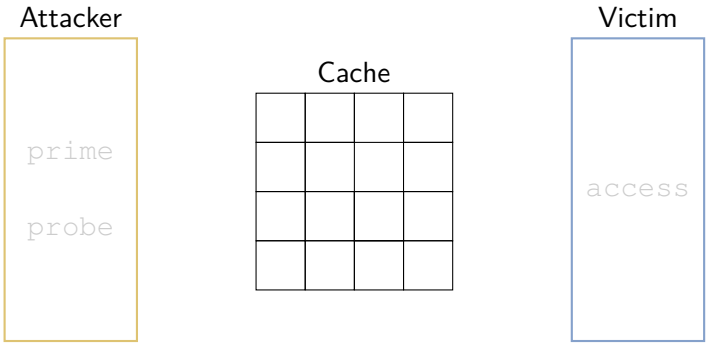
65–85 cycles difference between IOTLB hit and miss

R. Neugebauer et al., "Understanding PCIe performance for end host networking", SIGCOMM, 2018

C. Peglow, "Security analysis of hybrid Intel CPU/FPGA platforms using IOMMUs against I/O attacks", Thesis, University of Lübeck, 2020

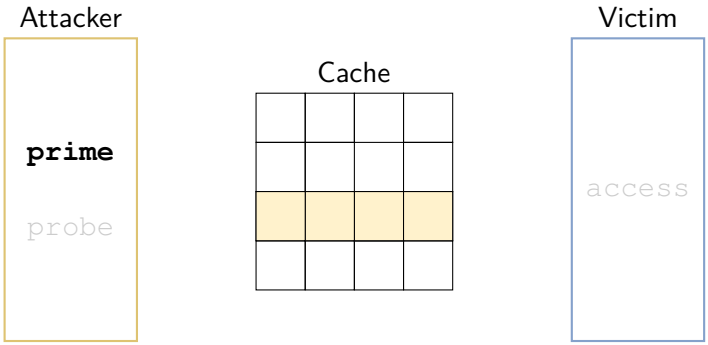
Eviction Sets

Prime+Probe



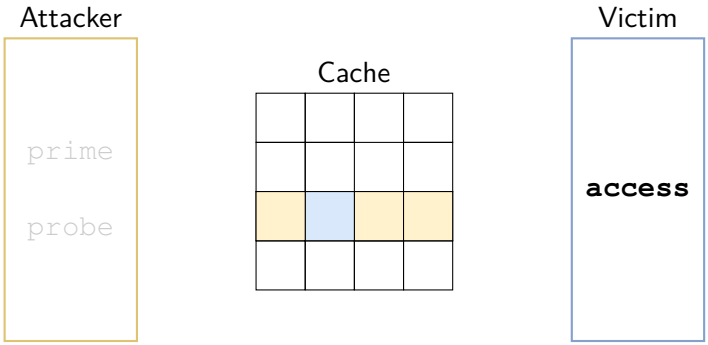
Eviction Sets

Prime+Probe



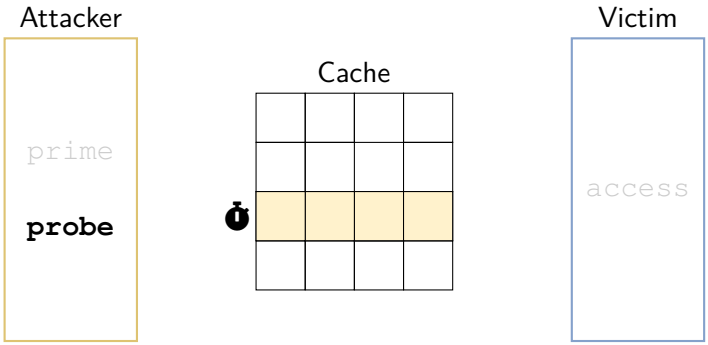
Eviction Sets

Prime+Probe



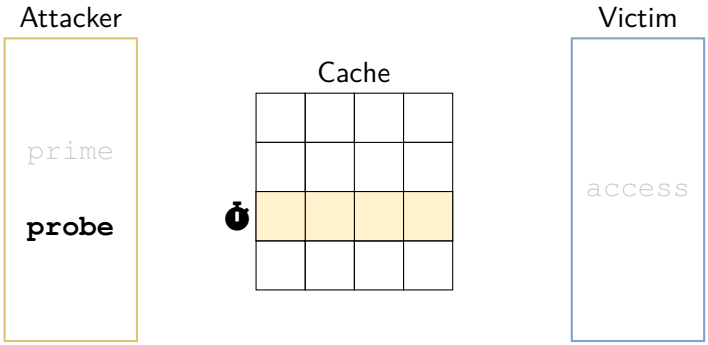
Eviction Sets

Prime+Probe



Eviction Sets

Prime+Probe



fast
Victim did not access

slow
Victim accessed

Eviction Sets

Algorithms

Grow-Split [Liu]

F. Liu et al., "Last-Level Cache Side-Channel Attacks are Practical", S&P, 2015

P. Vila et al., "Theory and Practice of Finding Eviction Sets", S&P, 2019

Eviction Sets

Algorithms

Grow-Split [Liu]

? Number of ways per cache set

F. Liu et al., "Last-Level Cache Side-Channel Attacks are Practical", S&P, 2015

P. Vila et al., "Theory and Practice of Finding Eviction Sets", S&P, 2019

Eviction Sets

Algorithms

Grow-Split [Liu]

- ? Number of ways per cache set
- ✔ Address to cache set mapping

F. Liu et al., "Last-Level Cache Side-Channel Attacks are Practical", S&P, 2015

P. Vila et al., "Theory and Practice of Finding Eviction Sets", S&P, 2019

Eviction Sets

Algorithms

Grow-Split [Liu]

- ? Number of ways per cache set
- ✔ Address to cache set mapping

Baseline Reduction [Vila]

F. Liu et al., "Last-Level Cache Side-Channel Attacks are Practical", S&P, 2015

P. Vila et al., "Theory and Practice of Finding Eviction Sets", S&P, 2019

Eviction Sets

Algorithms

Grow-Split [Liu]

- ? Number of ways per cache set
- ✔ Address to cache set mapping

Baseline Reduction [Vila]

- ✔ Number of ways per cache set

F. Liu et al., "Last-Level Cache Side-Channel Attacks are Practical", S&P, 2015

P. Vila et al., "Theory and Practice of Finding Eviction Sets", S&P, 2019

Eviction Sets

Algorithms

Grow-Split [Liu]

- ? Number of ways per cache set
- ✔ Address to cache set mapping

Baseline Reduction [Vila]

- ✔ Number of ways per cache set
- ? Address to cache set mapping

F. Liu et al., "Last-Level Cache Side-Channel Attacks are Practical", S&P, 2015

P. Vila et al., "Theory and Practice of Finding Eviction Sets", S&P, 2019

Eviction Sets

Algorithms

Grow-Split [Liu]

- ? Number of ways per cache set
- ✓ Address to cache set mapping

Baseline Reduction [Vila]

- ✓ Number of ways per cache set
- ? Address to cache set mapping

We combine both algorithms to not require prior knowledge.

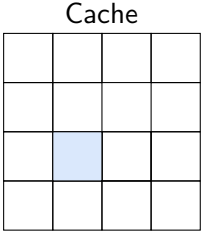
F. Liu et al., "Last-Level Cache Side-Channel Attacks are Practical", S&P, 2015

P. Vila et al., "Theory and Practice of Finding Eviction Sets", S&P, 2019

Eviction Sets

Grow-Reduce Algorithm

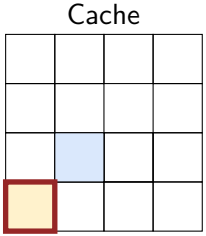
```
// Grow
while count < 50 do
    page ←∈ pool
    evset ←+ page
    pool ←− page
    if evicts(target, evset) then
        count ← count + 1
```



Eviction Sets

Grow-Reduce Algorithm

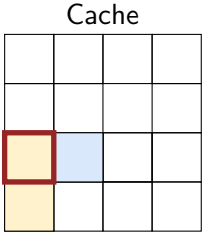
```
// Grow
while count < 50 do
    page ←∈ pool
    evset ←+ page
    pool ←− page
    if evicts(target, evset) then
        count ← count + 1
```



Eviction Sets

Grow-Reduce Algorithm

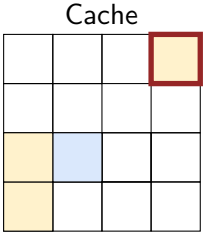
```
// Grow
while count < 50 do
    page ←∈ pool
    evset ←+ page
    pool ←− page
    if evicts(target, evset) then
        count ← count + 1
```



Eviction Sets

Grow-Reduce Algorithm

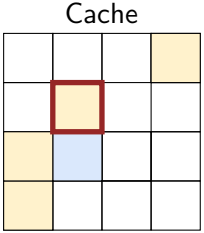
```
// Grow
while count < 50 do
    page ←∈ pool
    evset ←+ page
    pool ←− page
    if evicts(target, evset) then
        count ← count + 1
```



Eviction Sets

Grow-Reduce Algorithm

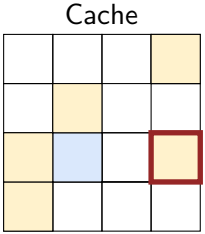
```
// Grow
while count < 50 do
    page ←∈ pool
    evset ←+ page
    pool ←− page
    if evicts(target, evset) then
        count ← count + 1
```



Eviction Sets

Grow-Reduce Algorithm

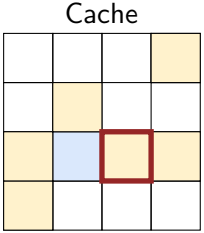
```
// Grow
while count < 50 do
    page ←∈ pool
    evset ←+ page
    pool ←− page
    if evicts(target, evset) then
        count ← count + 1
```



Eviction Sets

Grow-Reduce Algorithm

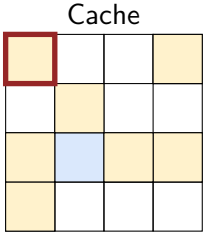
```
// Grow
while count < 50 do
    page ←∈ pool
    evset ←+ page
    pool ←− page
    if evicts(target, evset) then
        count ← count + 1
```



Eviction Sets

Grow-Reduce Algorithm

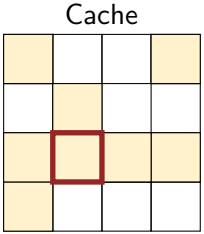
```
// Grow
while count < 50 do
    page ←∈ pool
    evset ←+ page
    pool ←− page
    if evicts(target, evset) then
        count ← count + 1
```



Eviction Sets

Grow-Reduce Algorithm

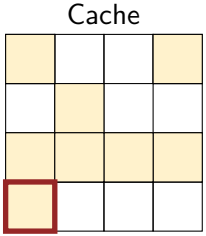
```
// Grow
while count < 50 do
    page ←∈ pool
    evset ←+ page
    pool ←− page
    if evicts(target, evset) then
        count ← count + 1
```



Eviction Sets

Grow-Reduce Algorithm

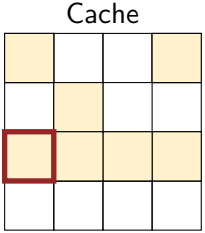
```
// Reduce
foreach page in evset do
    evset ←_ page
    if not evicts(target, evset) then
        evset ←+ page
return evset
```



Eviction Sets

Grow-Reduce Algorithm

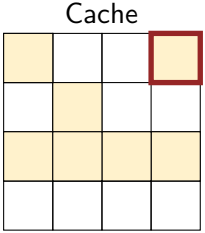
```
// Reduce
foreach page in evset do
    evset ←_ page
    if not evicts(target, evset) then
        evset ←+ page
return evset
```



Eviction Sets

Grow-Reduce Algorithm

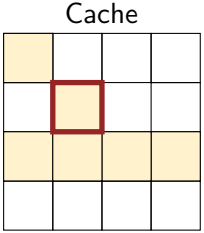
```
// Reduce
foreach page in evset do
    evset ←_ page
    if not evicts(target, evset) then
        evset ←_+ page
return evset
```



Eviction Sets

Grow-Reduce Algorithm

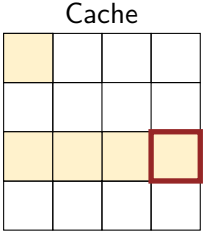
```
// Reduce
foreach page in evset do
    evset ←_ page
    if not evicts(target, evset) then
        evset ←+ page
return evset
```



Eviction Sets

Grow-Reduce Algorithm

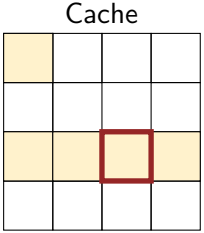
```
// Reduce
foreach page in evset do
    evset ←_ page
    if not evicts(target, evset) then
        evset ←+ page
return evset
```



Eviction Sets

Grow-Reduce Algorithm

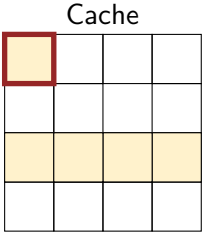
```
// Reduce
foreach page in evset do
    evset ←_ page
    if not evicts(target, evset) then
        evset ←+ page
return evset
```



Eviction Sets

Grow-Reduce Algorithm

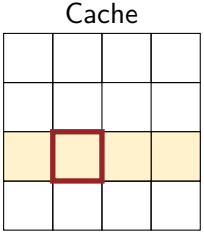
```
// Reduce
foreach page in evset do
    evset ←_ page
    if not evicts(target, evset) then
        evset ←+ page
return evset
```



Eviction Sets

Grow-Reduce Algorithm

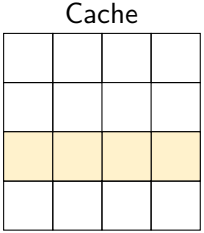
```
// Reduce
foreach page in evset do
    evset ←_ page
    if not evicts(target, evset) then
        evset ←+ page
return evset
```



Eviction Sets

Grow-Reduce Algorithm

```
// Reduce
foreach page in evset do
    evset ←_ page
    if not evicts(target, evset) then
        evset ←+ page
return evset
```



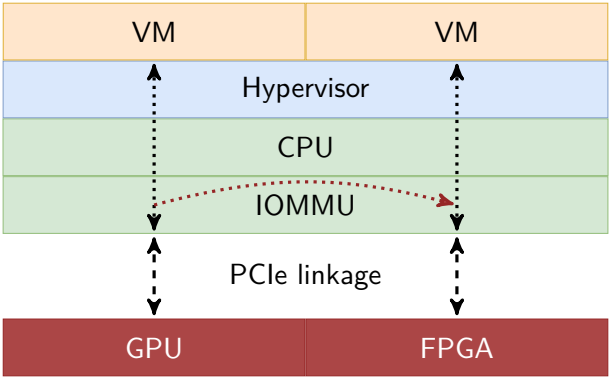
Eviction Sets

Evaluation

Flush	Algorithm	Sets	Set size	Useful sets per target	Average best eviction rate
✔	Grow-Split [Liu]	1.00	118.00	1.00	100.00 %
	Grow-Reduce	1.00	118.00	1.00	100.00 %
✘	Grow-Split [Liu]	10.70	50.69	0.98	28.00 %
	Grow-Reduce	32.08	110.05	0.98	82.23 %

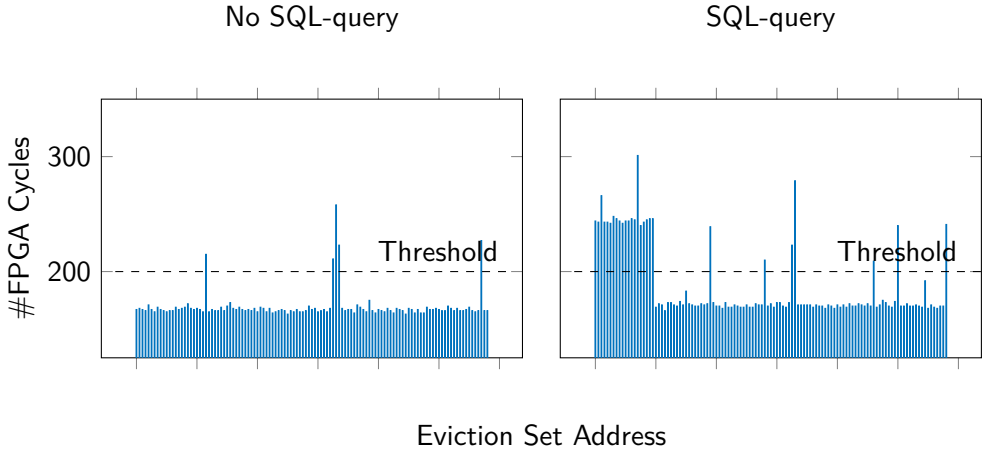
F. Liu et al., "Last-Level Cache Side-Channel Attacks are Practical", S&P, 2015

Threat Model



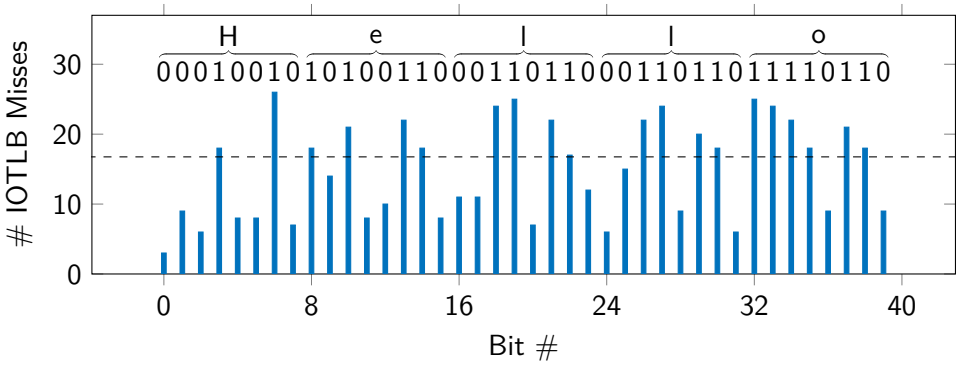
Case Study

GPU Database



Covert Channels

PCIe ↔ PCIe



Covert Channels

CPU → PCIe

- Requires ring 0 privileges

Covert Channels

CPU → PCIe

- Requires ring 0 privileges
- Flush+Reload

Covert Channels

CPU → PCIe

How about the reverse direction?

- Requires ring 0 privileges
- Flush+Reload

Covert Channels

CPU → PCIe

- Requires ring 0 privileges
- Flush+Reload

How about the reverse direction?

- Ansynchronous IOTLB flush

Covert Channels

CPU → PCIe

- Requires ring 0 privileges
- Flush+Reload

How about the reverse direction?

- Ansynchronous IOTLB flush
- Flush execution time is *not* data-dependent

Covert Channels

CPU → PCIe

- Requires ring 0 privileges
- Flush+Reload

How about the reverse direction?

- Ansynchronous IOTLB flush
- Flush execution time is *not* data-dependent
- No PCIe → CPU

Covert Channels

Channel	Method	Env	Throughput	Error	Message content
PCIe → PCIe	P+P	Host	3.4 bps	0%	All 1s
			6.65 bps	0%	50/50
			246.15 bps	0.1%	All 0s
			7.58 bps	0%	ASCII
CPU → PCIe	F+R	Host	15023 bps	30.09%	

Countermeasures

Application

Countermeasures

Application

- Constant-time code

Countermeasures

Application

- Constant-time code

Hypervisor

Countermeasures

Application

- Constant-time code

Hypervisor

- Address Translation Services

Countermeasures

Application

- Constant-time code

Hypervisor

- Address Translation Services
- Set-based IOTLB partitioning

Countermeasures

Application

- Constant-time code

Hypervisor

- Address Translation Services
- Set-based IOTLB partitioning

Physical

Countermeasures

Application

- Constant-time code

Hypervisor

- Address Translation Services
- Set-based IOTLB partitioning

Physical

- Plug devices into separate IOMMUs

Countermeasures

Application

- Constant-time code

Hypervisor

- Address Translation Services
- Set-based IOTLB partitioning

Physical

- Plug devices into separate IOMMUs

Hardware

Countermeasures

Application

- Constant-time code

Hypervisor

- Address Translation Services
- Set-based IOTLB partitioning

Physical

- Plug devices into separate IOMMUs

Hardware

- Way-based IOTLB partitioning

Countermeasures

Application

- Constant-time code

Hypervisor

- Address Translation Services
- Set-based IOTLB partitioning

Physical

- Plug devices into separate IOMMUs

Hardware

- Way-based IOTLB partitioning
- Un-cacheable translations

Conclusion

- Previously ignored side-channel for DMA-capable devices identified

Conclusion

- Previously ignored side-channel for DMA-capable devices identified
- Eviction set algorithm without prior knowledge

Conclusion

- Previously ignored side-channel for DMA-capable devices identified
- Eviction set algorithm without prior knowledge
- First IOTLB covert channel

Conclusion

- Previously ignored side-channel for DMA-capable devices identified
- Eviction set algorithm without prior knowledge
- First IOTLB covert channel

Thore Tiemann
@ t.tiemann@uni-luebeck.de
@ThoreTiemann
https://www.its.uni-luebeck.de/en/staff/thore-tiemann.html

Thank you for your attention!