Spectre and Meltdown vs. Real-Time: How Much do Mitigations Cost?

Ralf Ramsauer*, Jan Kiszka†, Wolfgang Mauerer*†

* Ostbayerische Technische Hochschule Regensburg
† Siemens AG, Corporate Research and Technology, München

Embedded Linux Conference Europe, Edinburgh

October 22, 2018

Motivation

Motivation

- CVE 2017-5753, CVE 2017-5715, CVE 2017-5754, CVE 2018-3640, CVE 2018-3639, CVE 2018-3665, CVE 2018-3693, CVE 2018-3620, CVE 2018-3646, CVE 2018-3615, CVE 2018-9056

Motivation

- CVE 2017-5753, CVE 2017-5715, CVE 2017-5754, CVE 2018-3640, CVE 2018-3639, CVE 2018-3665, CVE 2018-3693, CVE 2018-3620, CVE 2018-3646, CVE 2018-3615, CVE 2018-9056
- Spectre mitigations lower performance (throughput)
- Benchmarks for Nginx / Apache / *SQL / Compilation / Video {En,De}coding / Git(!) / FSmark / BlockBench / Dbench / IOZone . . .

Motivation

- CVE 2017-5753, CVE 2017-5715, CVE 2017-5754, CVE 2018-3640, CVE 2018-3639, CVE 2018-3665, CVE 2018-3693, CVE 2018-3620, CVE 2018-3646, CVE 2018-3615, CVE 2018-9056
- Spectre mitigations lower performance (throughput)
- Benchmarks for Nginx / Apache / *SQL / Compilation / Video {En,De}coding / Git(!) / FSmark / BlockBench / Dbench / IOZone . . .

. . . but what about determinism and real-time?

Spectre & Meltdown

Overview – Variants

- Variant 1 (Bounds Check Bypass)
- Variant 2 (Branch Target Injection)
- Variant 3 (Meltdown)
- Variant 4 (Speculative Store Bypass)
- L1TF (Foreshadow, Foreshadow-NG)
- ...

Logo Spectre, Meltdown & Foreshadow

Spectre & Meltdown

Attack Surface

- Exploitation of branch prediction and speculative execution
- Leak **sensitive information**
- Violation of memory protection guarantees
- **Local vector**: attacker needs to be able to execute arbitrary code
- High attack complexity

Logo Spectre, Meltdown & Foreshadow

Mitigations

(Focus on X86 / ARM64)

Spectre Mitigations

Variant 1 – Bounds Check Bypass

- __user pointer sanitization
- nospec accessors
- syscall / hypercall protection
- protect paths where user controls array access
- Cost per mitigation call: very low

Variant 2 – Branch Target Injection

- X86 / ARM64: `CONFIG_RETPOLINE`
- X86: speculation control through µcode
- ARM64: Requires SMCCC v1.1 (ATF v1.6)
- Fill return stack buffer on context switch
- Cost per mitigation call: very high

Spectre Mitigations

Variant 3 – Meltdown

- X86: Page Table Isolation (PTI)
- ARM64: Page Table Isolation (KPTI / KAISER)
- Different page directories for kernel/userspace
- Unmap kernel from userspace
- Cost per mitigation call: avg

Variant 4 – Speculative Store Bypass

- Intel: requires µcode update
- ARM64: Requires SMCCC v1.1 (ATF v1.6)
- notable performance impact: mitigations applied on per-process basis
- Cost per mitigation call: very high

Level 1 Terminal Fault – Foreshadow

Mitigation

- Disable SMT (Intel HTT, per FW or `nosmt`)
- PTE inversion (neglictible cost)
- (conditional) cache flushes on VMENTER
- Cost per mitigation call: native: very low – VM: very high

Affected CPUs

	v1	v2	v3	v4	L1TF
x86 AMD[a]			✓		✓
speculative x86 Intel[b]					
speculative ARM[c]			A75		✓

[a] www.amd.com/en/corporate/security-updates

[b] www.intel.com/content/www/us/en/architecture-and-technology/side-channel-variants-1-2-3.html

[c] developer.arm.com/support/arm-security-updates/speculative-processor-vulnerability

Mitigations vs. Real-Time: What do Mitigations Cost?

Real-Time vs. Mitigations

Real-Time

- Deterministic responses to stimuli
- Time-sensitive cyclic execution
- Bounded latencies
  (not too late, not too early)
- Repeatable results
- Optimise/Quantify **worst case**
- Determinism matters more than throughput!

Real-Time vs. Mitigations

Mitigations

- Cheap: PTE inversion, __user pointer sanitization
- Expensive: Cache flushes, TLB flushes, PTI, RSB fills, Retpoline, FW calls on ctx switches
- A **lot** of additional sources of overhead
- Mitigations may increase latency

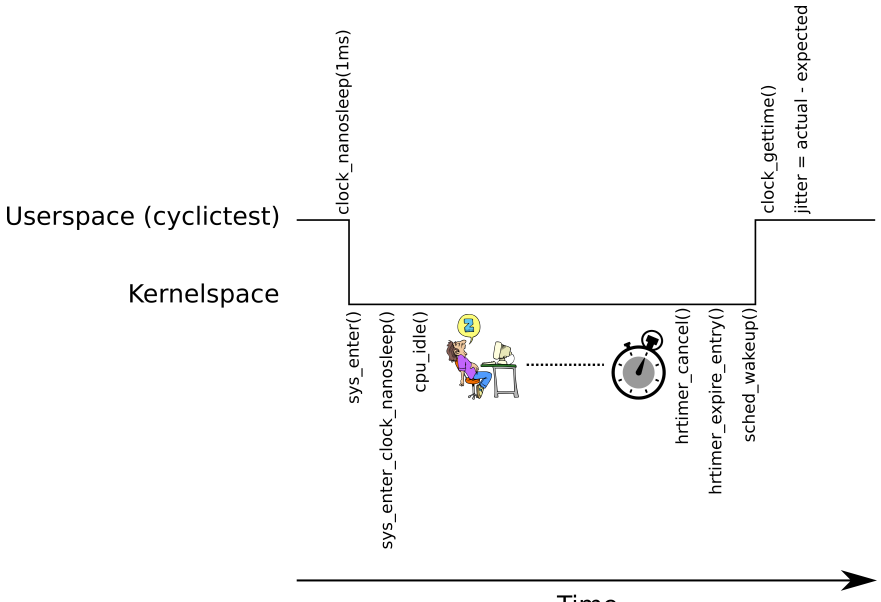
Measurement Setup

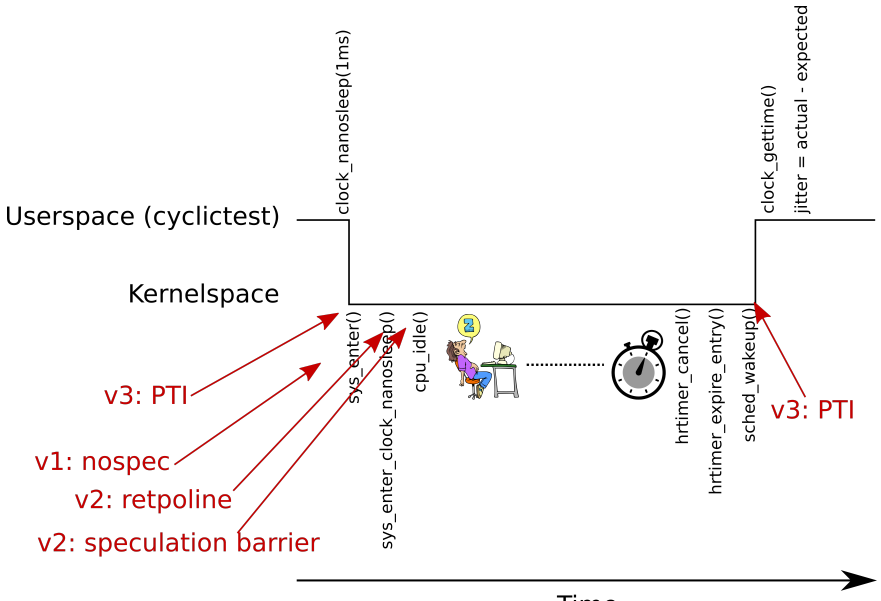
Basic idea

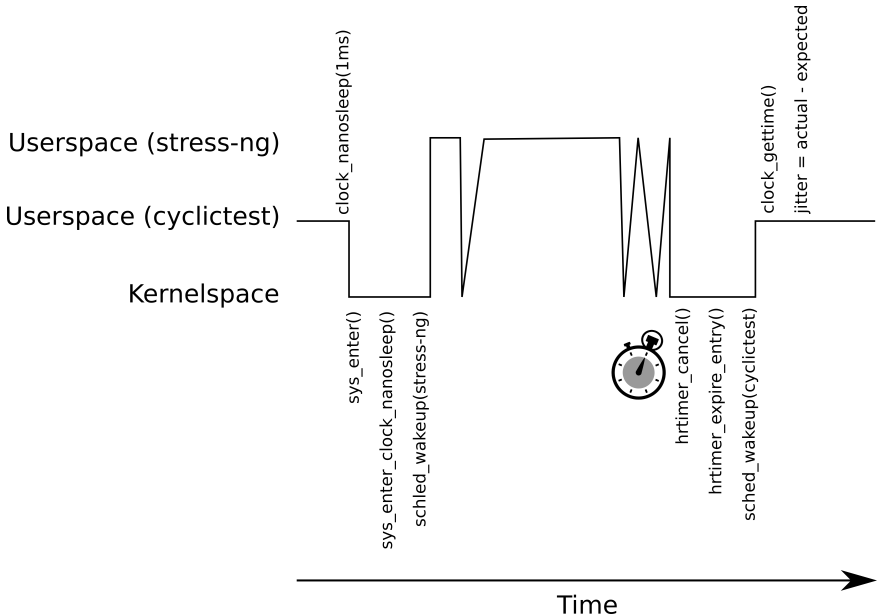
- Run **RT payload** on multiple cores
- Measure responsivity
- Repeat measurements
  - **w/ and w/o mitigations**
  - **w and w/o additional load**
- Reduce variation of mitigations
  - No protection
  - Default protection
  - PTI only
  - Variant 2 only
- Proper statistic analysis

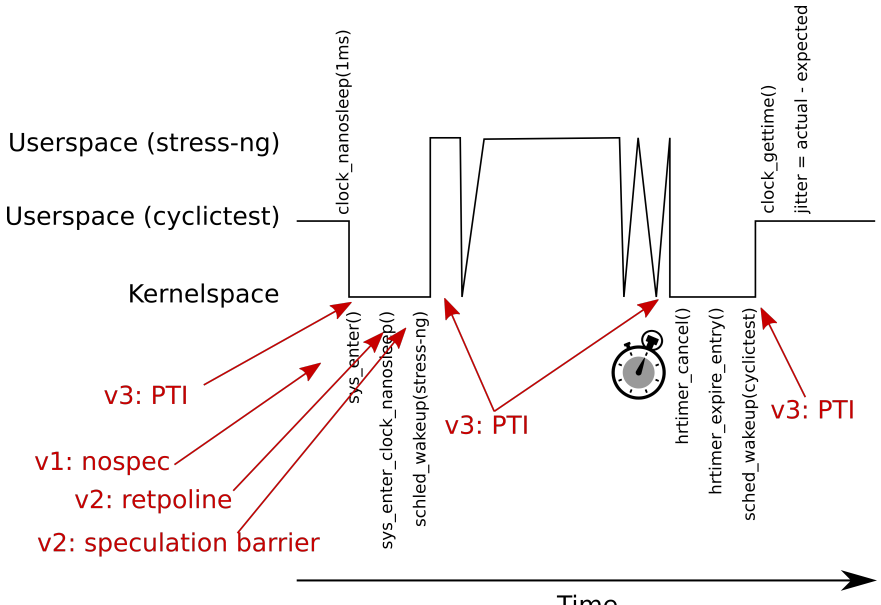
Tools

- RT payload: **`cyclictest`**
- Non-RT load: **`stress-ng`**
- Repeat in **virtualised environment (Jailhouse)**
- Complex RT applications may vary


Userspace (cyclictest)
Kernelspace
clock_nanosleep(1ms)
clock_gettime()
jitter = actual - expected
sys_enter()
sys_enter_clock_nanosleep()
cpu_idle()
hrtimer_cancel()
hrtimer_expire_entry()
sched_wakeup()
Time

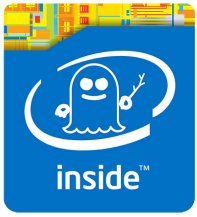

clock_nanosleep(1ms)
clock_gettime()
jitter = actual - expected
Userspace (cyclictest)
Kernelspace
sys_enter()
sys_enter_clock_nanosleep()
cpu_idle()
hrtimer_cancel()
hrtimer_expire_entry()
sched_wakeup()
v3: PTI
v1: nospec
v2: retpoline
v2: speculation barrier
v3: PTI
Time

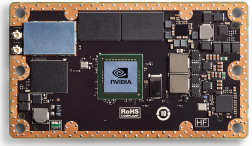

Userspace (stress-ng)
Userspace (cyclictest)
Kernelspace
clock_nanosleep(1ms)
sys_enter()
sys_enter_clock_nanosleep()
schled_wakeup(stress-ng)
hrtimer_cancel()
hrtimer_expire_entry()
sched_wakeup(cyclictest)
clock_gettime()
jitter = actual - expected
Time


Userspace (stress-ng)
Userspace (cyclictest)
Kernelspace
clock_nanosleep(1ms)
sys_enter()
sys_enter_clock_nanosleep()
schled_wakeup(stress-ng)
hrtimer_cancel()
hrtimer_expire_entry()
sched_wakeup(cyclictest)
clock_gettime()
jitter = actual - expected
v3: PTI
v1: nospec
v2: retpoline
v2: speculation barrier
v3: PTI
v3: PTI
Time

Target systems

- Xeon E5-2683 v4, 32 cores
  v1, v2, v3, v4, L1TF
- AMD Ryzen 2700X
  v1, v2, v4
- Nvidia Jetson TX1, 4x Cortex A57
  v1, v2, v4





Get targets under control

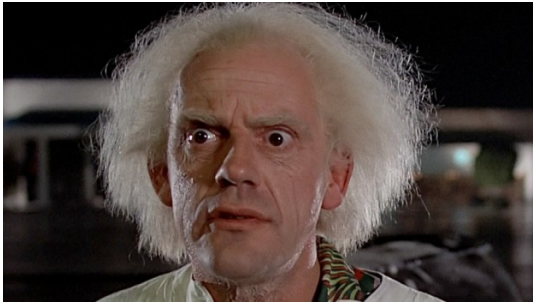
- Undesired high latencies
  - watchdogs, NMI watchdogs
  - Machine Check Polls
  - SMIs
  - `CONFIG_KSM`, `CONFIG_MIGRATION`, `CONFIG_COMPACTION`
  - High latencies when running across NUMA boundaries

`http://linuxrealtime.org/index.php/Improving_the_Real-Time_Properties`

Back to the Future

Get targets under control – cont'd

- reduce *system noise*
  - isolate RT CPUs with `isolcpus`
  - rebind SMP affinity of IRQs
  - rebind net device affinity
  - disable MCP
- Don't leave NUMA node
- cyclictest: broken taskset parsing
- stress-ng: SCHED_FIFO + alarm()s + RT = broken

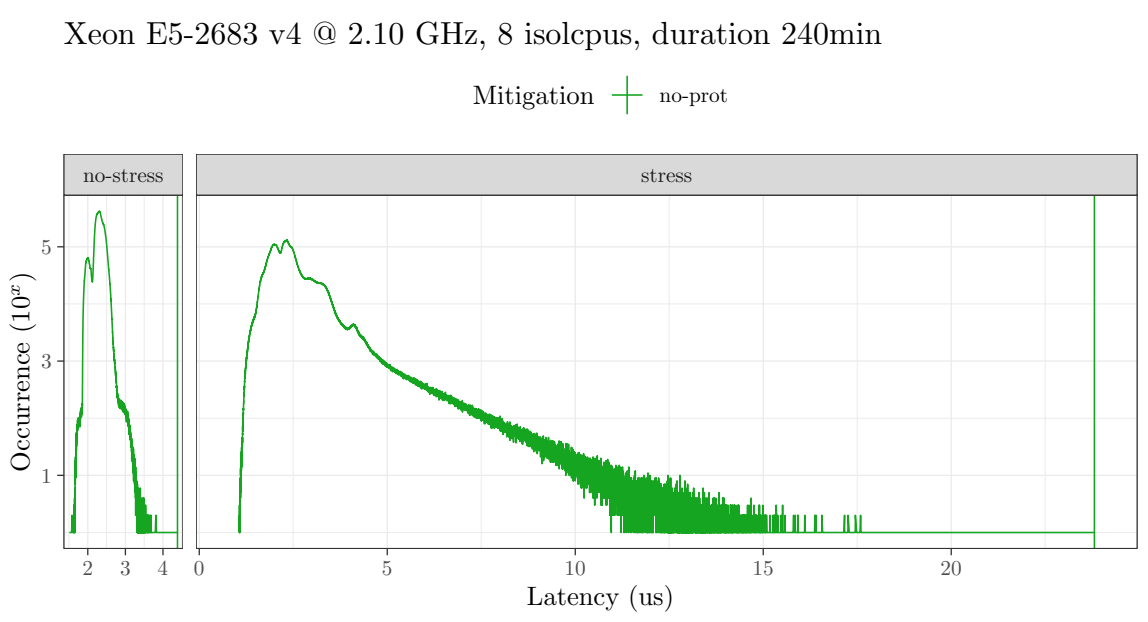


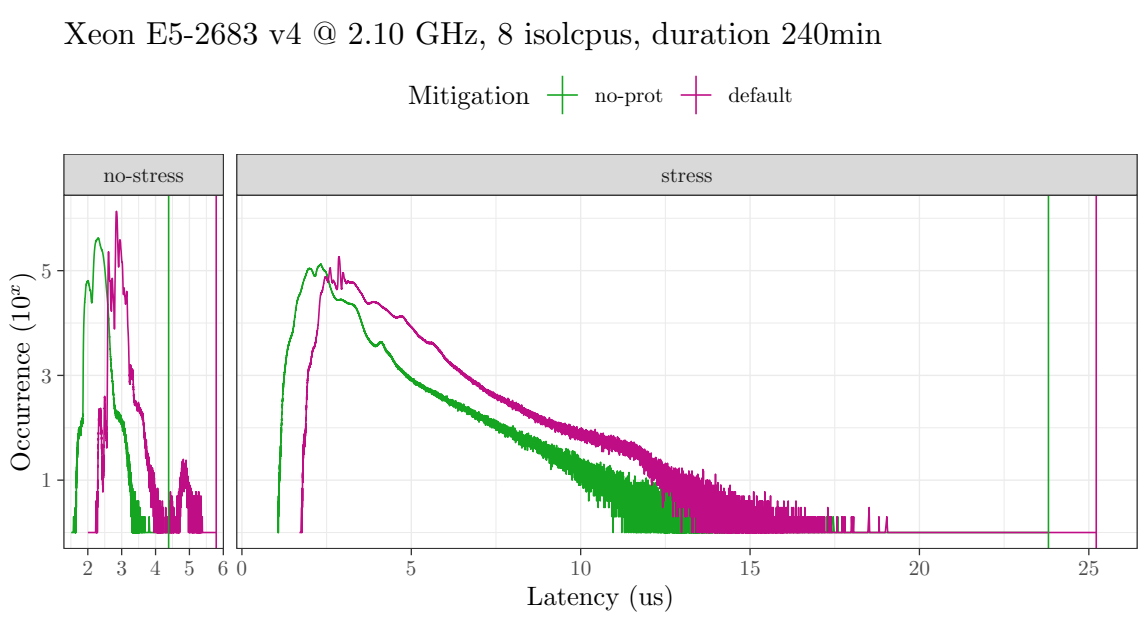
Back to the Future

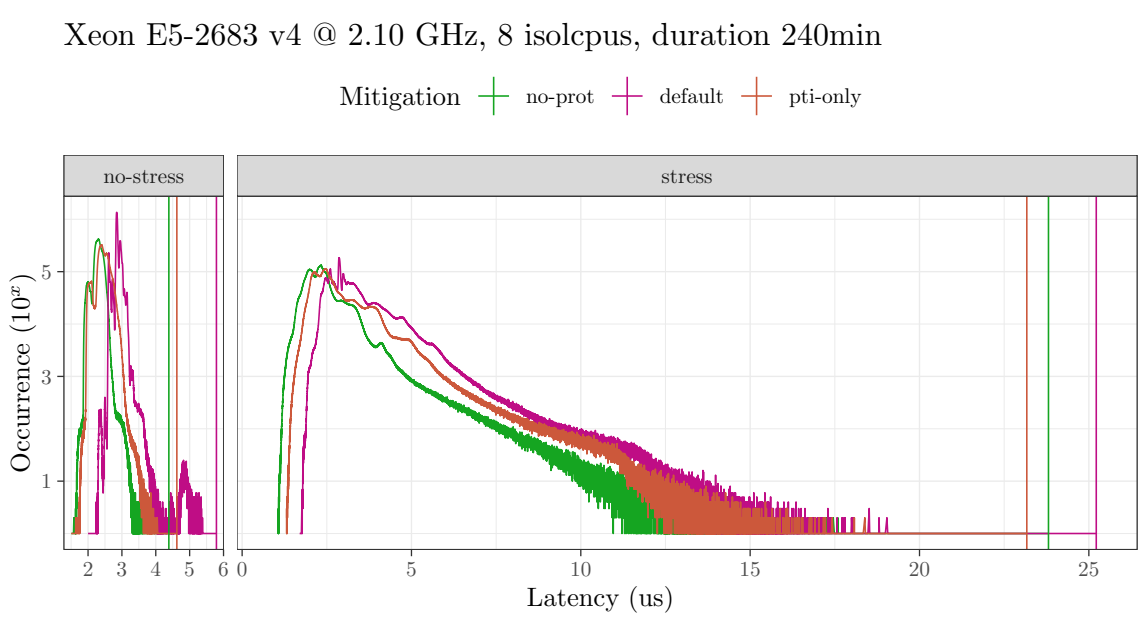
Get targets under control – cont'd II

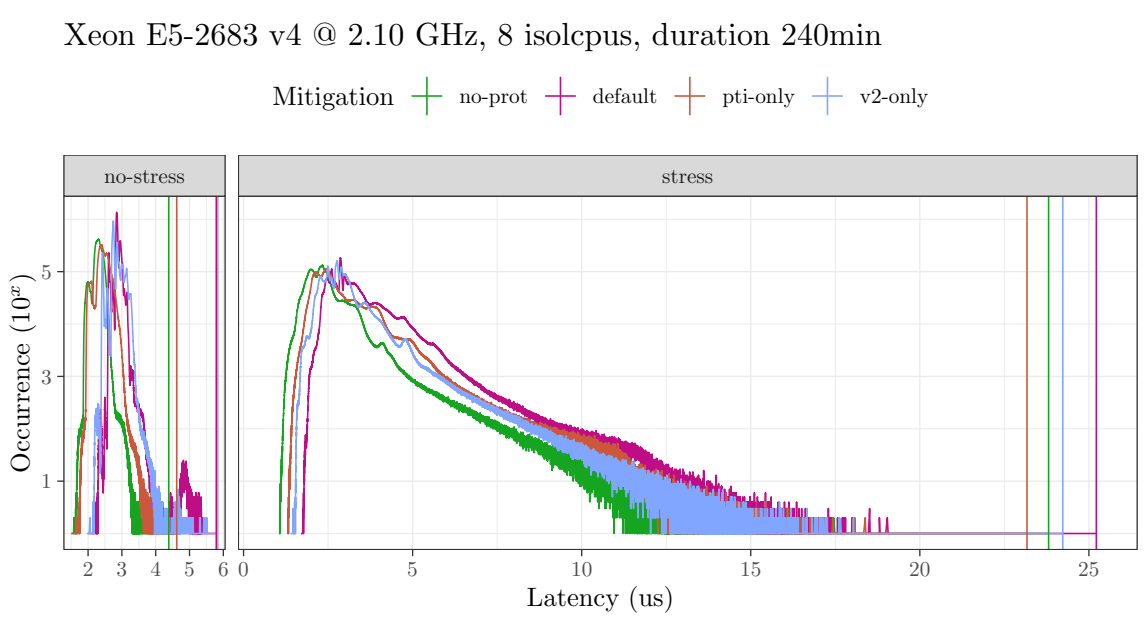
- X86: exchange microcode to test different mitigations
- Jetson TX1: Missing official ATF support for SMCCC v1.1. Sigh...
- Different kernel variants for different tests
- Issues with virtualised Jailhouse environment
  - X86: high number of vmexits set us on wrong track: configuration issue
  - ARM64: missing Spectre v2 mitigation
- Obstacles in the **whole** system stack

Back to the Future


Xeon E5-2683 v4 @ 2.10 GHz, 8 isolcpus, duration 240min
Mitigation
no-prot
no-stress
stress
Occurrence (10^x)
5
3
1
2
3
4
0
5
10
15
20
Latency (us)

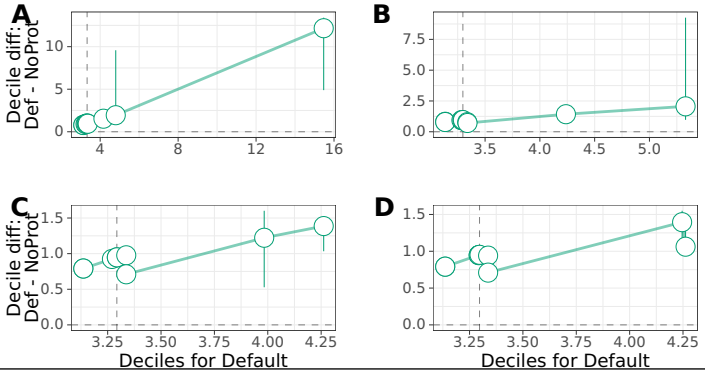

Xeon E5-2683 v4 @ 2.10 GHz, 8 isolcpus, duration 240min
Mitigation
no-prot
default
no-stress
stress
Occurrence (10^x)
1
3
5
2
3
4
5
6
0
5
10
15
20
25
Latency (us)


Xeon E5-2683 v4 @ 2.10 GHz, 8 isolcpus, duration 240min
Mitigation
no-prot
default
pti-only
no-stress
stress
Occurrence (10^x)
1
3
5
2
3
4
5
6
0
5
10
15
20
25
Latency (us)

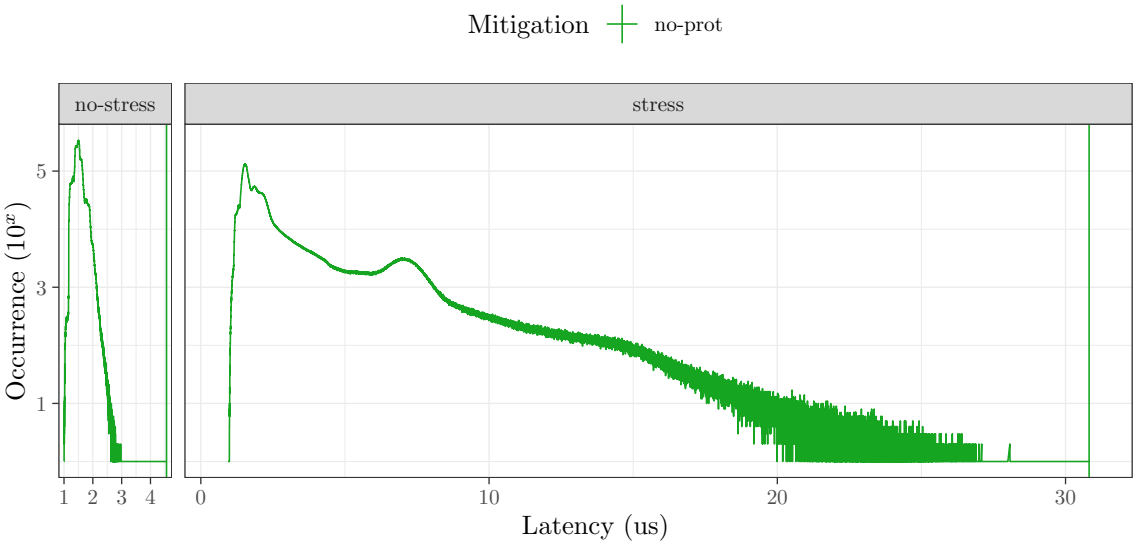

Xeon E5-2683 v4 @ 2.10 GHz, 8 isolcpus, duration 240min
Mitigation
no-prot
default
pti-only
v2-only
no-stress
stress
Occurrence (10^x)
1
3
5
2
3
4
5
6
0
5
10
15
20
25
Latency (us)

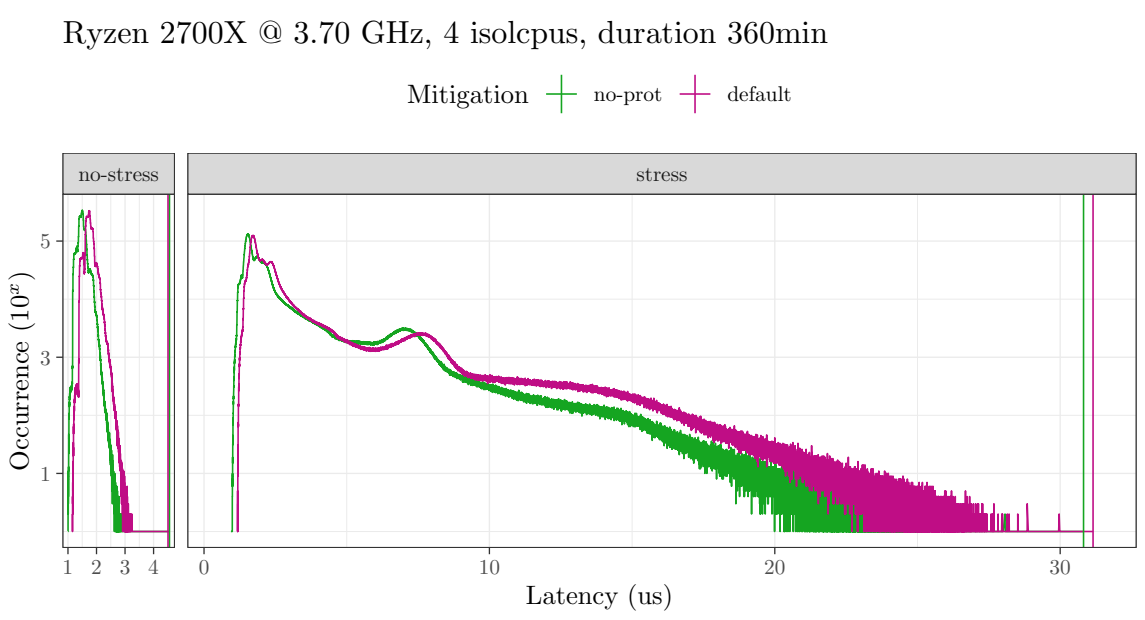
Can we be sure?

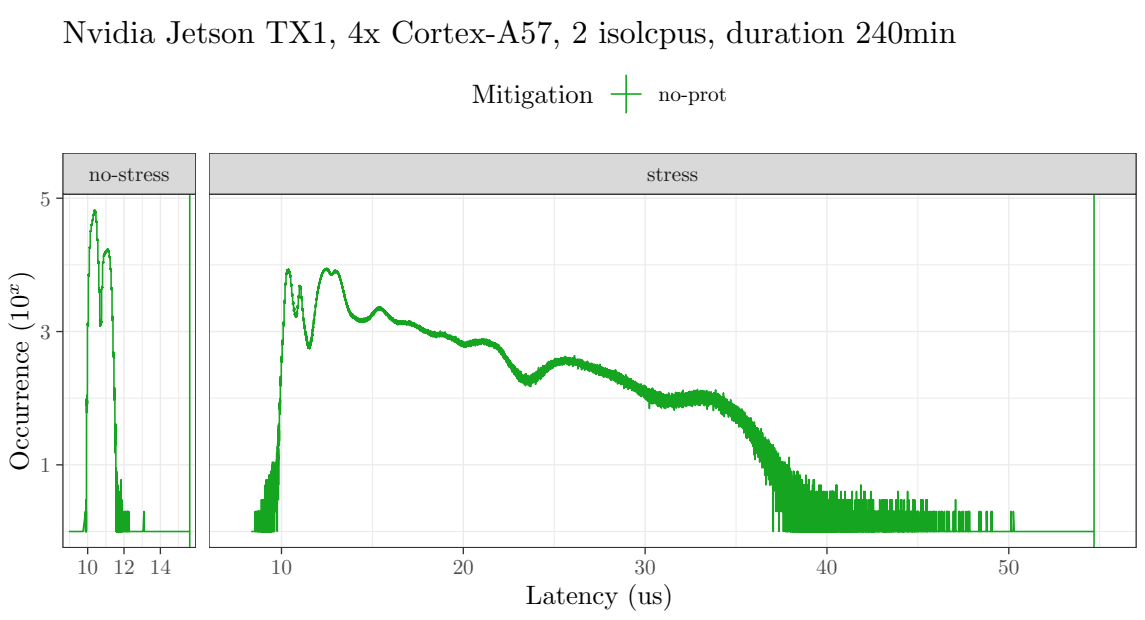
- Trust our measurements?
- Detailed story

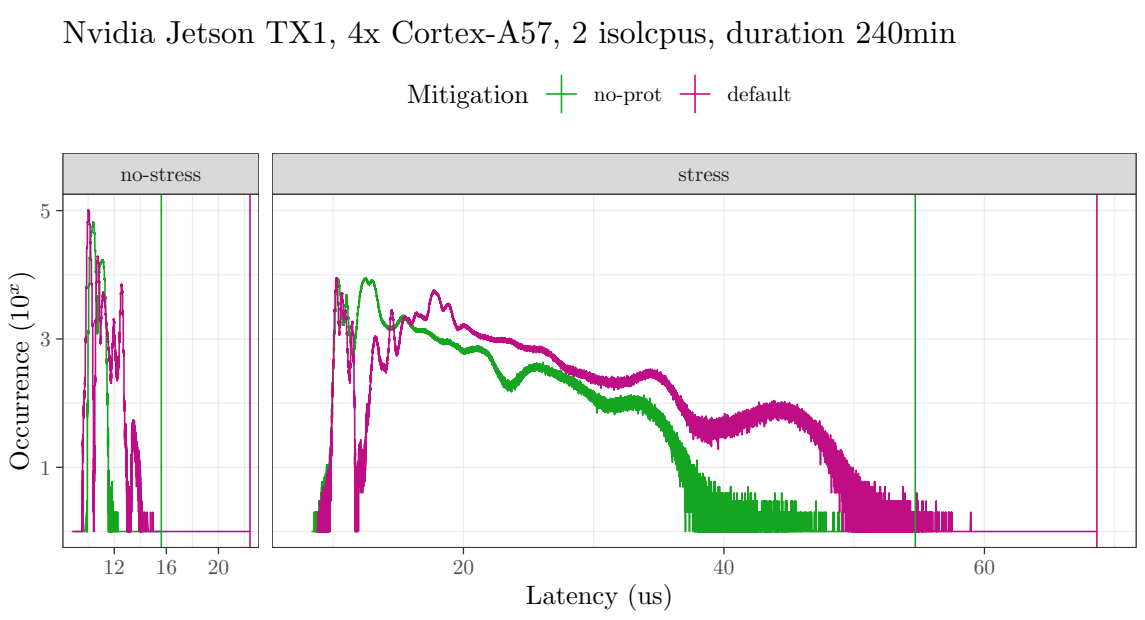


Ryzen 2700X @ 3.70 GHz, 4 isolcpus, duration 360min




Ryzen 2700X @ 3.70 GHz, 4 isolcpus, duration 360min
Mitigation
no-prot
default
no-stress
stress
Occurrence (10^x)
5
3
1
1
2
3
4
0
10
20
30
Latency (us)


Nvidia Jetson TX1, 4x Cortex-A57, 2 isolcpus, duration 240min
Mitigation
no-prot
no-stress
stress
Occurrence (10^x)
5
3
1
10
12
14
10
20
30
40
50
Latency (us)

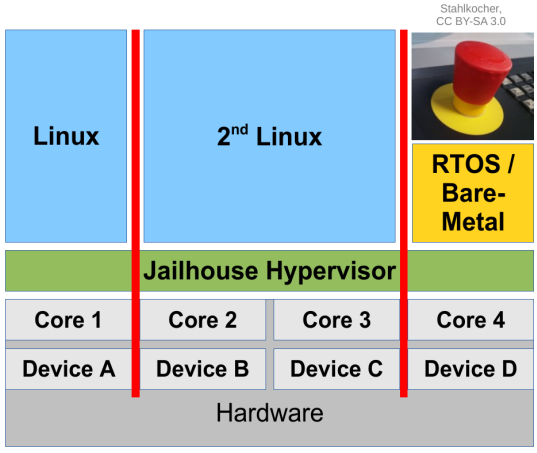

Nvidia Jetson TX1, 4x Cortex-A57, 2 isolcpus, duration 240min
Mitigation
no-prot
default
no-stress
stress
Occurrence (10^x)
5
3
1
12
16
20
20
40
60
Latency (us)

Mitigations & Virtualisation

Jailhouse – static partitioning for multicore systems

Design goals

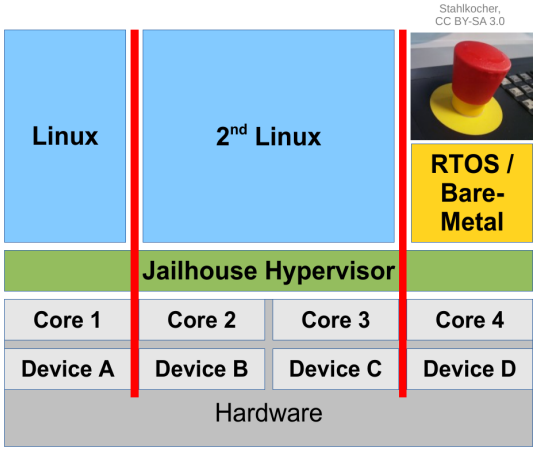
- Focus on maintaining static partitions
- No scheduling
- 1:1 resource assignment
- Hard RT properties with minimal overhead

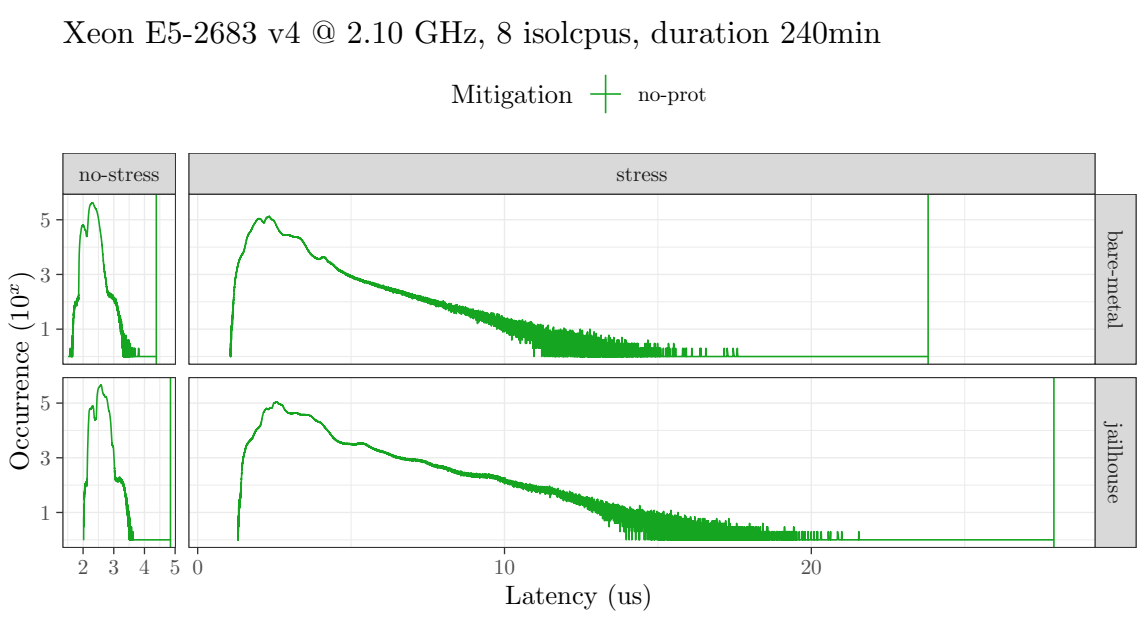


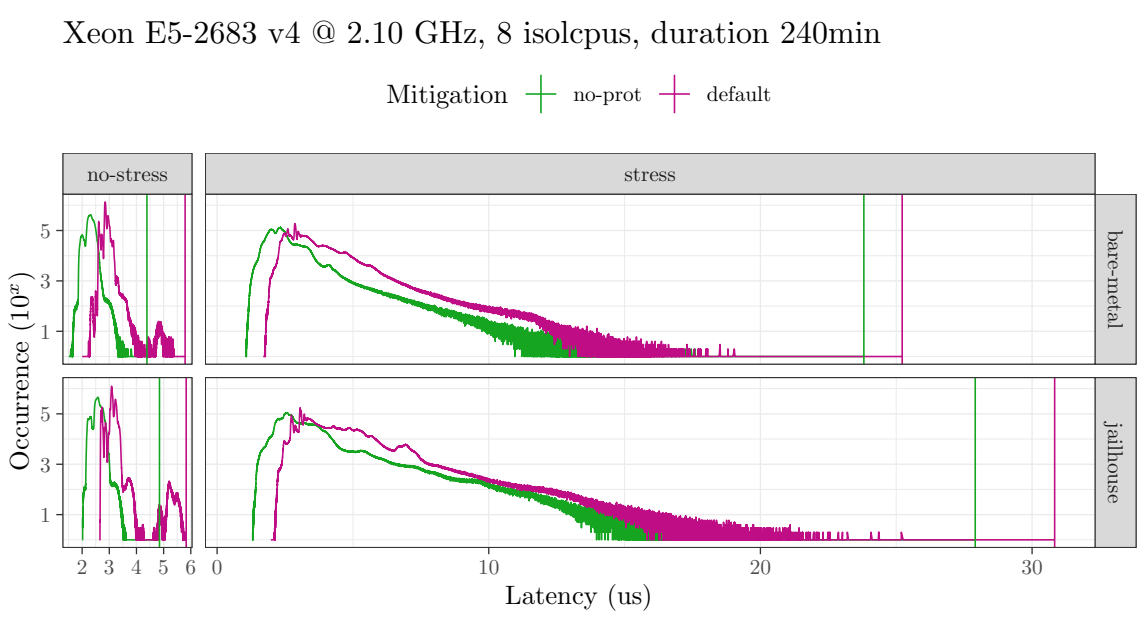
Jailhouse & Spectre

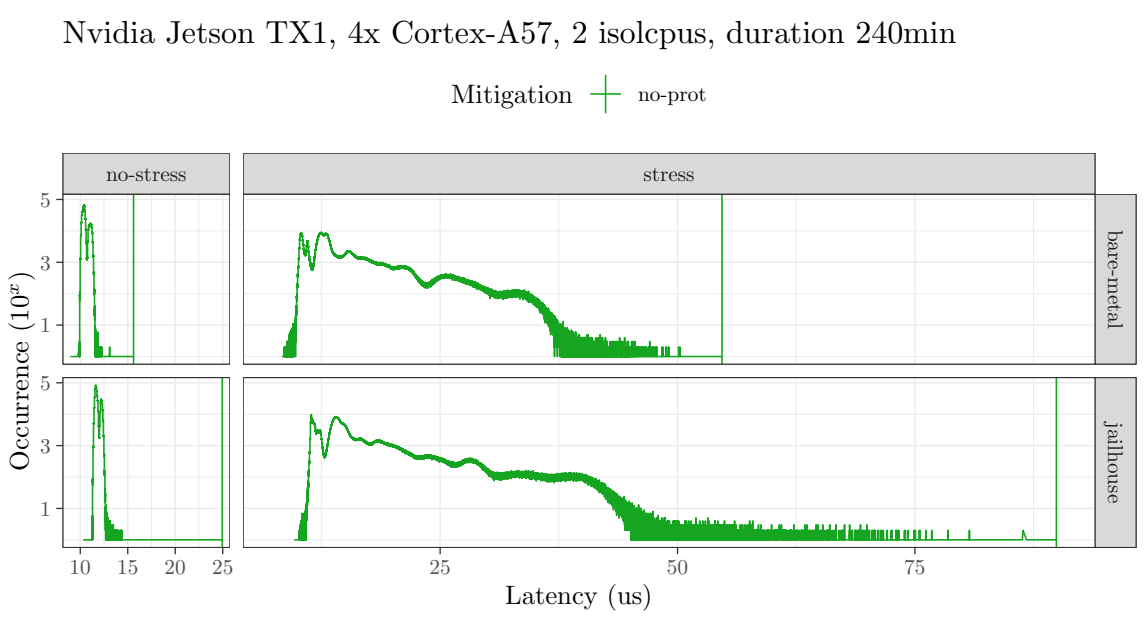
Mitigations & Alternatives

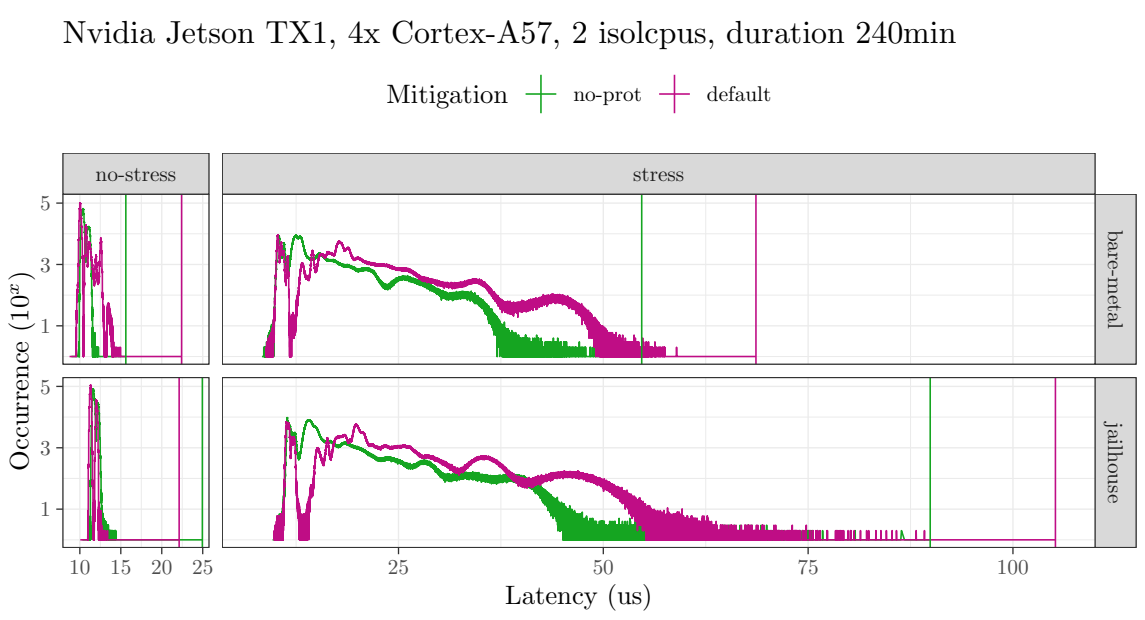
- Static system partitioning reduces exploitability
- Jailhouse eliminates *confused deputy* scenario by design:
  cannot leak what you cannot see
- Can static partition be an alternative to standard mitigations?




Xeon E5-2683 v4 @ 2.10 GHz, 8 isolcpus, duration 240min
Mitigation
no-prot
no-stress
stress
bare-metal
jailhouse
Occurrence (10^x)
5
3
1
2
3
4
5
0
10
20
Latency (us)


Xeon E5-2683 v4 @ 2.10 GHz, 8 isolcpus, duration 240min
Mitigation
no-prot
default
no-stress
stress
bare-metal
jailhouse
Occurrence (10^x)
5
3
1
2
3
4
5
6
0
10
20
30
Latency (us)


Nvidia Jetson TX1, 4x Cortex-A57, 2 isolcpus, duration 240min
Mitigation
no-prot
no-stress
stress
bare-metal
jailhouse
Occurrence (10^x)
5
3
1
10
15
20
25
50
75
Latency (us)


Nvidia Jetson TX1, 4x Cortex-A57, 2 isolcpus, duration 240min
Mitigation
no-prot
default
no-stress
stress
bare-metal
jailhouse
Occurrence (10^x)
5
3
1
10
15
20
25
50
75
100
Latency (us)

Jailhouse & Spectre

- Static partitioning: *more* expensive than native mitigations
- Stronger isolation, mitigate mitigate future exploits?
- ⇒ Keep RT system aspects **unpatched**
- Jailhouse: probably best case
  - but other hypervisors not examined
  - Complex hypervisors: different picture

Conclusion

- Histogram give a *tendency* of the influence of mitigations
- Measure your own workload
- **Local vector + use case:** Are mitigations actually required?

Open Issues

- stress-ng: use timers instead of alarm()s
- cyclictest: broken taskset parsing
- Jailhouse: optimizes spectre mitigation forwarding for ARM64 in the queue

Thanks for your attention!

{ralf.ramsauer,wolfgang.mauerer}@othr.de
jan.kiszka@siemens.com