Prefetch Side-Channel Attacks: Bypassing SMAP and Kernel ASLR

D. Gruss, C. Maurice, A. Fogh†, M. Lipp, S. Mangard
Graz University of Technology, † G DATA Advanced Analytics

October 25, 2016 — CCS'16

Overview

- prefetch instructions don't check privileges
- prefetch instructions leak timing information

Overview

- prefetch instructions don't check privileges
- prefetch instructions leak timing information

exploit this to:

- locate a driver in kernel = defeat KASLR
- translate virtual to physical addresses

Intel being overspecific

NOTE

Intel being overspecific

NOTE

Using the PREFETCH instruction is recommended only if data does not fit in cache.

Intel being overspecific

NOTE

Using the PREFETCH instruction is recommended only if data does not fit in cache. Use of software prefetch should be limited to memory addresses that are managed or owned within the application context.

Intel being overspecific

NOTE

Using the PREFETCH instruction is recommended only if data does not fit in cache. Use of software prefetch should be limited to memory addresses that are managed or owned within the application context. Prefetching to addresses that are not mapped to physical pages can experience non-deterministic performance penalty.

Intel being overspecific

NOTE

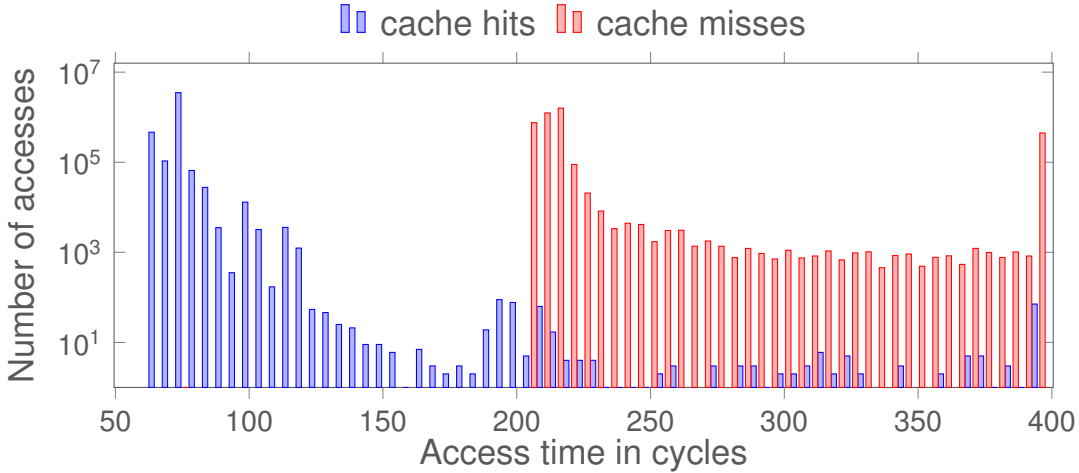
Using the PREFETCH instruction is recommended only if data does not fit in cache. Use of software prefetch should be limited to memory addresses that are managed or owned within the application context. Prefetching to addresses that are not mapped to physical pages can experience non-deterministic performance penalty. For example specifying a NULL pointer (0L) as address for a prefetch can cause long delays.

CPU Caches

Memory (DRAM) is slow compared to the CPU

- buffer frequently used memory
- every memory reference goes through the cache
- based on physical addresses

Memory Access Latency



Unprivileged cache maintainance

Optimize cache usage:

- `prefetch`: suggest CPU to load data into cache
- `clflush`: throw out data from all caches

... based on virtual addresses

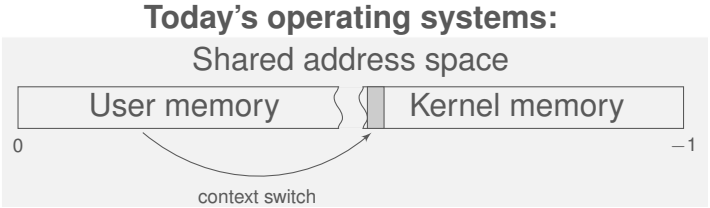
Software prefetching

`prefetch` instructions are somewhat unusual

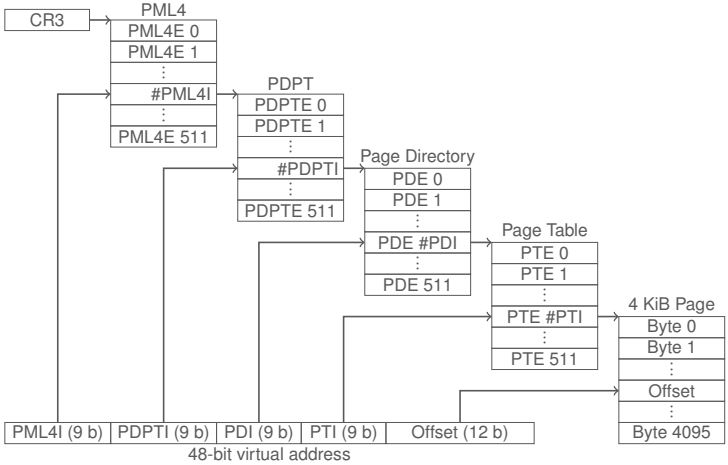
- hints – can be ignored by the CPU
- do not check privileges or cause exceptions

but they do need to translate virtual to physical

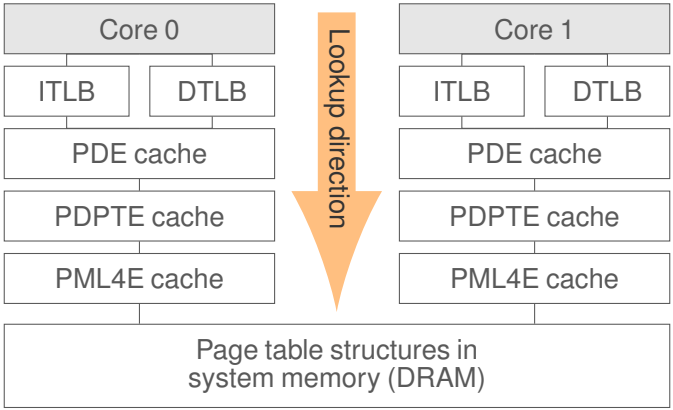
Kernel must be mapped in every address space



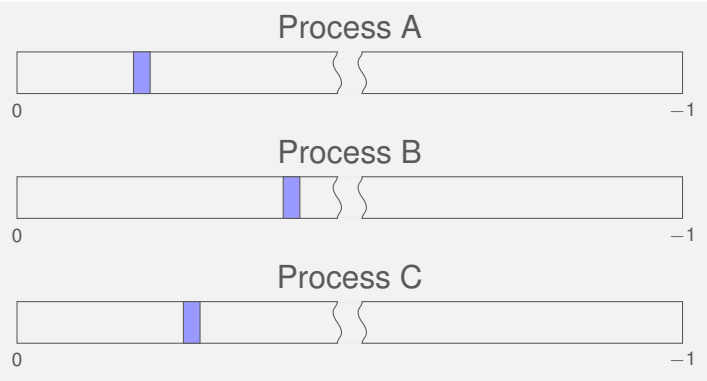
Address translation on x86-64



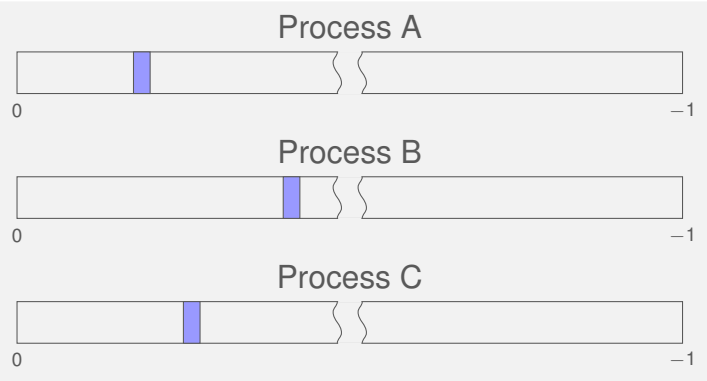
Address Translation Caches



Address Space Layout Randomization (ASLR)

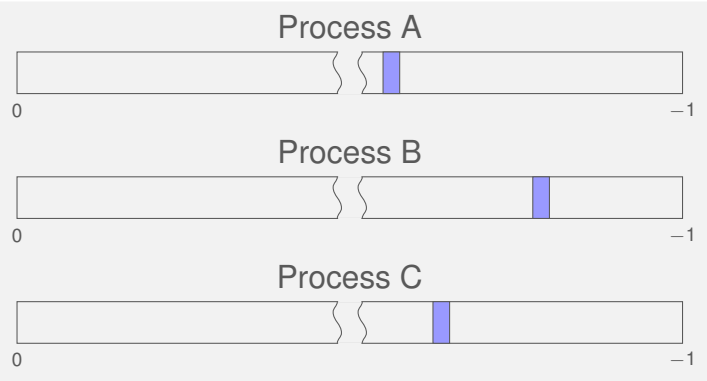


Address Space Layout Randomization (ASLR)

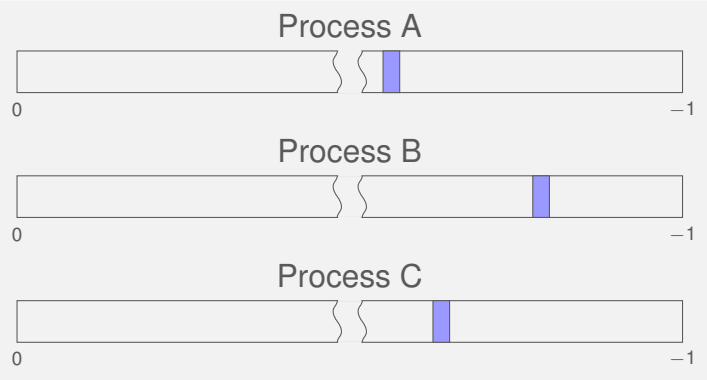


Same library – different offset!

Kernel Address Space Layout Randomization (KASLR)

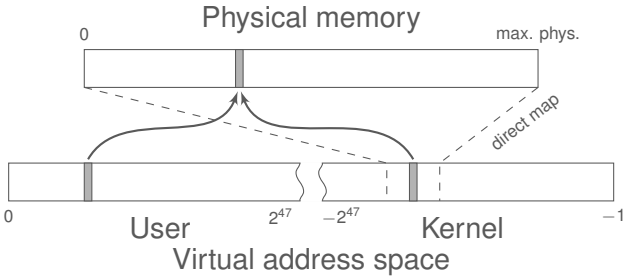


Kernel Address Space Layout Randomization (KASLR)

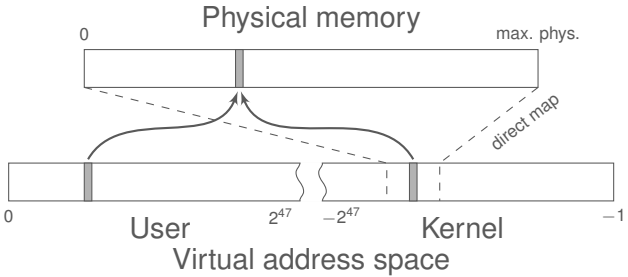


Same driver – different offset!

Kernel direct-physical map

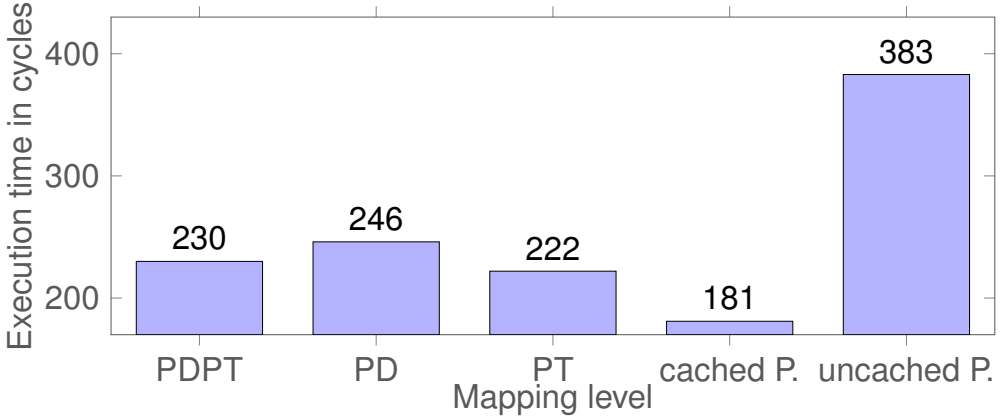


Kernel direct-physical map

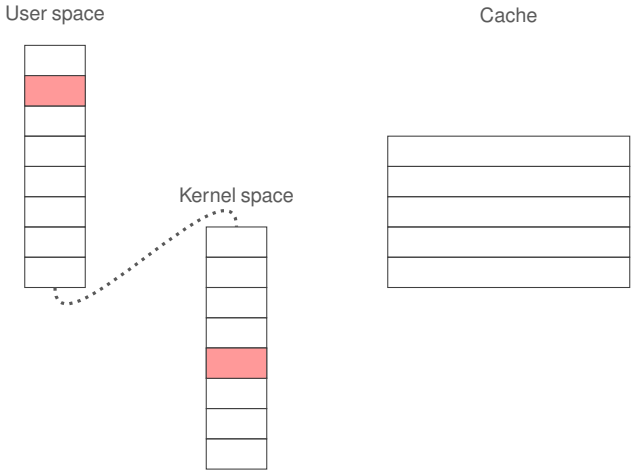


OS X, Linux, BSD, Xen PVM (Amazon EC2)

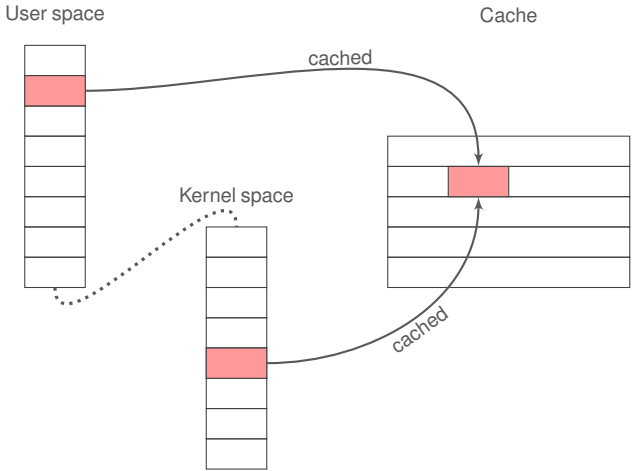
Translation-Level Oracle



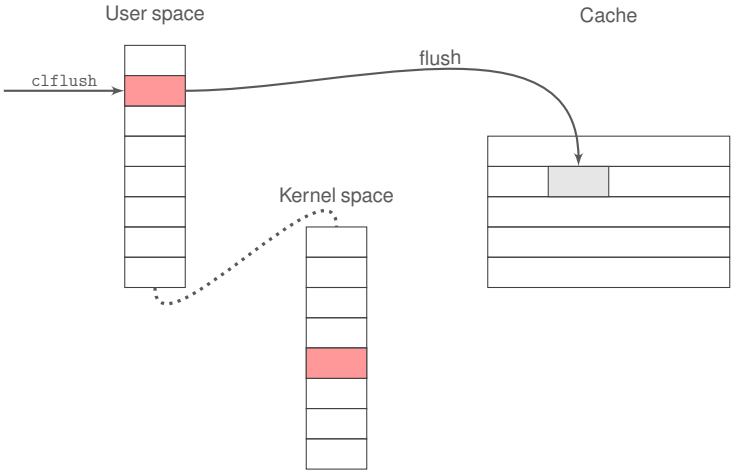
Address-Translation Oracle



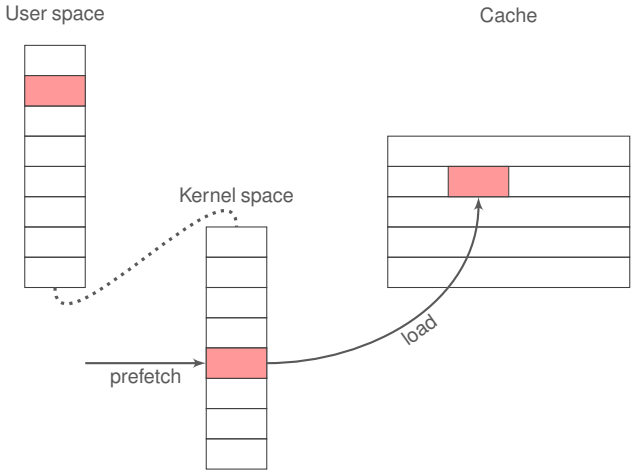
Address-Translation Oracle



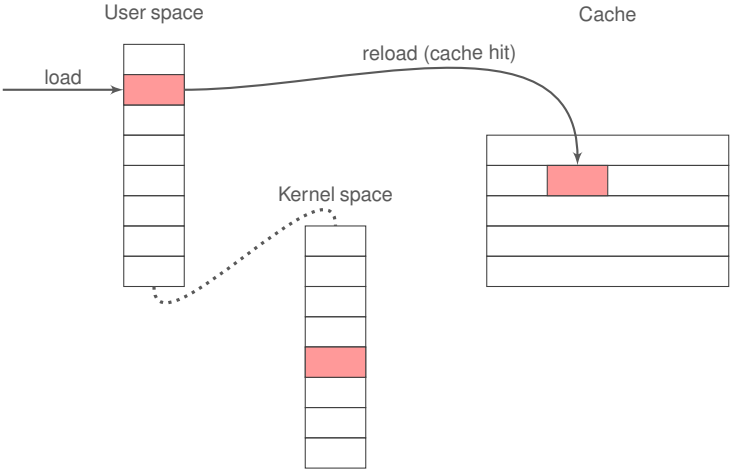
Address-Translation Oracle



Address-Translation Oracle



Address-Translation Oracle

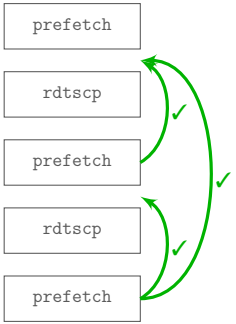


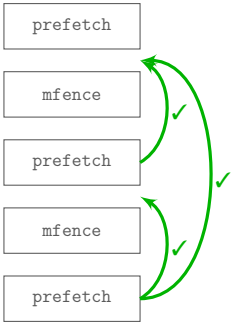
Timing the prefetch instruction

The CPU may reorder prefetch instruction – a look at `rdtscp`

Timing the prefetch instruction

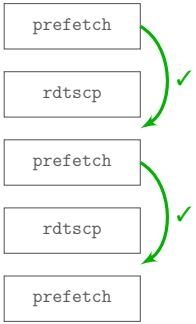
The CPU may reorder prefetch instruction – a look at `rdtscp`

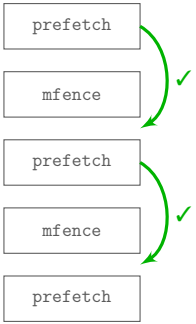




Timing the prefetch instruction

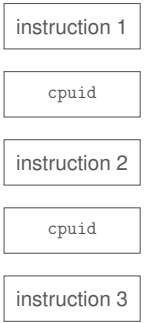
The CPU may reorder prefetch instruction – a look at `rdtscp`





Timing the prefetch instruction

The CPU may reorder instructions



but not over `cpuid`!

Windows 10 Memory layout

- HAL, kernel, kernel drivers located bewetween
  - start: `0xffff 8000 0000 0000`
  - end : `0xffff 9fff ffff ffff`

Breaking Windows KASLR

for all mapped pages (found via translation-level oracle):

Breaking Windows KASLR

for all mapped pages (found via translation-level oracle):

1. evict translation caches: `Sleep()` / access large memory buffer

Breaking Windows KASLR

for all mapped pages (found via translation-level oracle):

1. evict translation caches: `Sleep()` / access large memory buffer
2. perform syscall to driver

Breaking Windows KASLR

for all mapped pages (found via translation-level oracle):

1. evict translation caches: `Sleep()` / access large memory buffer
2. perform syscall to driver
3. time `prefetch(page address)`

Breaking Windows KASLR

for all mapped pages (found via translation-level oracle):

1. evict translation caches: `Sleep()` / access large memory buffer
2. perform syscall to driver
3. time `prefetch(page address)`

→ Fastest average access time is a driver page.

Breaking Windows KASLR

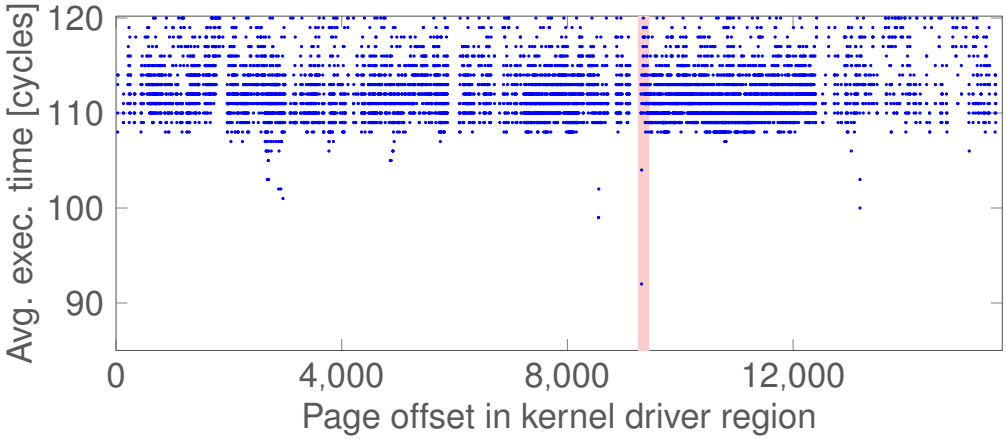
for all mapped pages (found via translation-level oracle):

1. evict translation caches: `Sleep()` / access large memory buffer
2. perform syscall to driver
3. time `prefetch(page address)`

$\rightarrow$ Fastest average access time is a driver page.

Full attack on Windows 10 in $<$ 12 seconds

Locate Kernel Driver (defeat KASLR)



Kernel exploits (10 years ago)

- overwrite return address

→ jump to userspace code

- overwrite stack pointer

→ switch to userspace stack

Mitigating kernel exploits

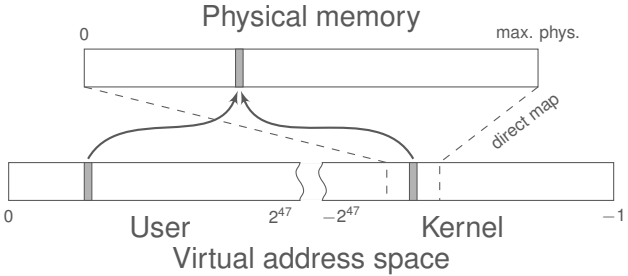
- Jump to userspace code? Nope! Hardware prevents that.

= supervisor-mode execution prevention (SMEP)

- Switch to userspace stack? Nope! Hardware prevents that.

= supervisor-mode access prevention (SMAP)

Kernel direct-physical map



Evading the mitigation

- get direct-physical-map address of userspace address

$\rightarrow$ jump/switch there

known as “ret2dir” attacks (Kemerlis et al. 2014)

Mitigating the evasion

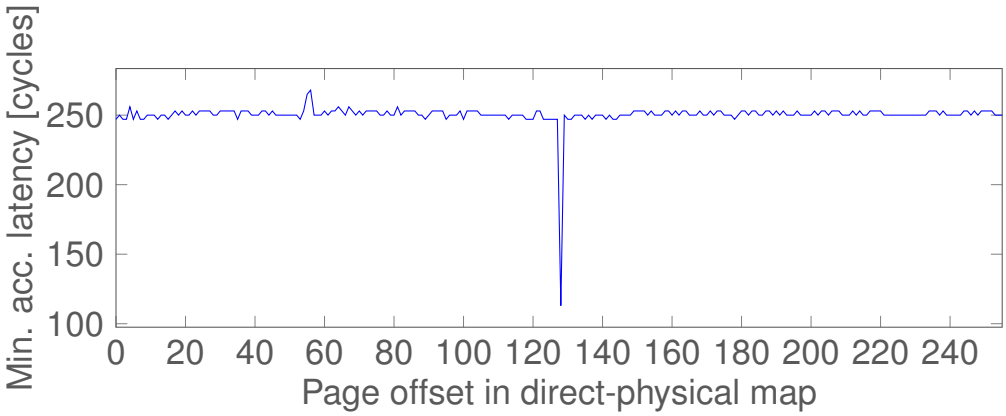
- getting rid of direct-physical map?

Mitigating the evasion

- getting rid of direct-physical map? Apparently not.

$\rightarrow$ do not leak physical addresses to user

Prefetching via direct-physical map



Prefetching via direct-physical map

- immediately leaks a direct-physical map address

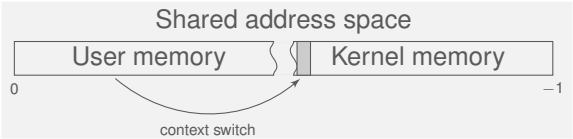
  → no physical address necessary (compared to ret2dir)

- if direct-physical map offset is known

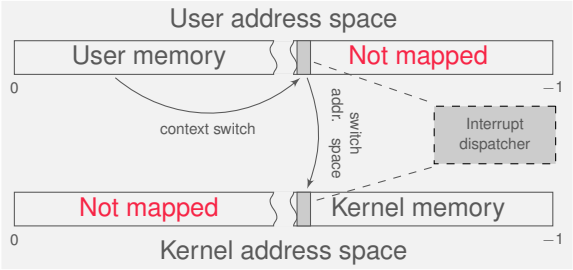
  → leaks physical address

Countermeasure

Today's operating systems:



Stronger kernel isolation:



Conclusion

- prefetch leaks significant information
- we can locate a driver in the kernel and thus break KASLR
- break SMAP/SMEP and get physical addresses
- countermeasure could be implemented in software

Prefetch Side-Channel Attacks: Bypassing SMAP and Kernel ASLR

D. Gruss, C. Maurice, A. Fogh†, M. Lipp, S. Mangard
Graz University of Technology, † G DATA Advanced Analytics

October 25, 2016 — CCS'16