Mitigating Spectre and Meltdown (and L1TF)

David Woodhouse

Kernel Recipes 2018

2018-09-28

Content

- Understanding speculative attacks
- Spectre & Meltdown
- KPTI / KAISER
- Microcode features
- Retpoline
- L1 Terminal Fault (L1TF)

Components of a speculative attack (1/3)

1. **Entering speculative execution**
   - Conditional branch

```
        movl   $100,%edi
loop:   …
        subl   $1,%edi
        jne    loop
```

Components of a speculative attack (1/3)

1. **Entering speculative execution**
   - Conditional branch
   - Indirect branch

```
func:  …

       movl   func,%edx
       jmp    *edx
```

Components of a speculative attack (1/3)

1. **Entering speculative execution**
   - Conditional branch
   - Indirect branch
   - Exceptions

```
movl   (%edx),%rax
movl   (%rax),%rbx
…
```

Components of a speculative attack (1/3)

1. **Entering speculative execution**
    - Conditional branch
    - Indirect branch
    - Exceptions
    - TSX

Components of a speculative attack (2/3)

2. **Prolonging speculative execution**
    - Load with cache miss
    - Dependent loads
    - Dependent arithmetic operations

Components of a speculative attack (3/3)

3) **Leaking information**

- Data cache
- Instruction cache
- Prediction cache
- Translation cache

Meltdown (aka ‘variant 3’)

- Allows direct read from non-permitted (e.g. kernel) memory.
- Runs entirely in unprivileged code.
- Affects Intel (not AMD), POWER, ARM Cortex-A75.

```
movl   $0xc1234567,%edx
movl   (%edx),%ecx
movl   mydata(%ecx),%edx
```

Spectre

- **Variant 2: Indirect branches**
  - Branch predictions from unprivileged mode, affect privileged code.
  - Attacker can cause kernel to *(speculatively)* run arbitrary code.

- **Variant 1: Conditional branches**
  - Loops will always happen n+1 times.
  - Sanity checks don’t prevent speculative execution.

Meltdown: KPTI / KAISER

- Kernel Address Isolation to have Side-channels Efficiently Removed
- Dual set of page tables per process
- Change `%cr3` (root of page tables) on each kernel entry/exit

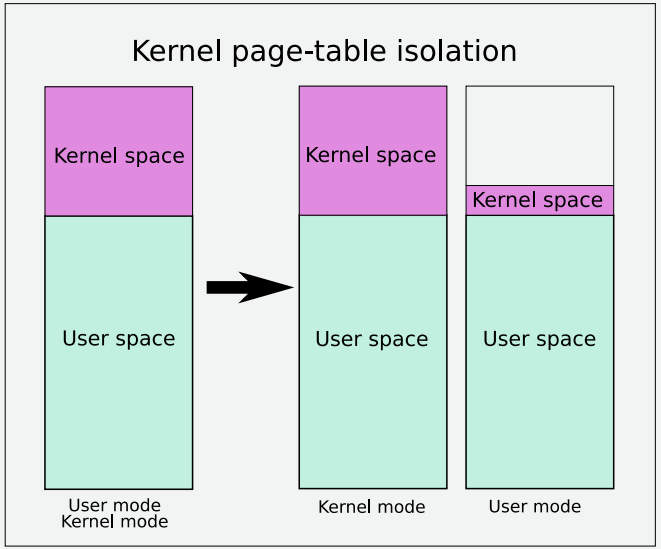

Image credit:
Wikipedia user Phoenix7777
CC BY-SA 4.0

Spectre v2: Microcode features

- New functions in MSRs:
  - Indirect Branch Restricted Speculation (IBRS)
  - Indirect Branch Prediction Barrier (IBPB)

Spectre v2: Retpoline

- Confuse the branch predictor!
- Original code:

```
        jmp *r11
```

- Replaced with:

```
        call set_up_target
capture_speculation:
        pause
        jmp capture_speculation
set_up_target:
        mov %r11, %(rsp)
        ret
```

Spectre v2: Retpoline for calls

```
call *r11
```

- Replaced with:

```
        jmp do_call
do_retpoline_jmp:
        call set_up_target
capture_speculation:
        pause
        jmp capture_speculation
set_up_target:
        mov %r11, %(rsp)
        ret
do_call:
        call do_retpoline_jmp
        …
```

Spectre v2: Reducing the retpoline impact

- Retpoline always causes a prediction miss

- If there's a common case, explicitly call it:

```
if (func == generic_func)
    generic_func();
else
    *func();
```

- Inline functions which take callback functions as arguments
  (e.g. `slot_handle_level_range()`)

∀ n: `callback(n);`

Spectre v2: Return Stack Buffer

- RSB cleared on context switch and VMEXIT.

```
    call 2f
1:  pause
    lfence
    jmp 1b
2:  call 4f
    …
```

} 32 times

- From Skylake onwards, Intel CPUs take branch predictions from the BTB when the RSB is depleted.

Spectre v1

- New `array_index_nospec()` adds data dependency in bounds checking.

```
if (index < size) {
    index = array_index_nospec(index, size);
    val = array[index];
}
```

- Similar masking in `get_user()` etc.

- Static analysis with Coverity and similar tools.

L1 Terminal Fault (L1TF)

- Non-present PTEs in a VM guest mishandled
- A hyperthread CPU can read any data its sibling is touching

L1 Terminal Fault (L1TF)

- Non-present PTEs in a VM guest mishandled
- A hyperthread CPU can read any data its sibling is touching
- Mitigations:
  - Flush L1$ on each VM entry

L1 Terminal Fault (L1TF)

- Non-present PTEs in a VM guest mishandled
- A hyperthread CPU can read any data its sibling is touching
- Mitigations:
  - Flush L1$ on each VM entry
  - Turn off hyperthreading

L1 Terminal Fault (L1TF)

- Non-present PTEs in a VM guest mishandled
- A hyperthread CPU can read any data its sibling is touching
- Mitigations:
  - Flush L1$ on each VM entry
  - Turn off hyperthreading
  - Co(re) scheduling: https://lwn.net/Articles/764482/

L1 Terminal Fault (L1TF)

- Non-present PTEs in a VM guest mishandled
- A hyperthread CPU can read any data its sibling is touching
- Mitigations:
  - Flush L1$ on each VM entry
  - Turn off hyperthreading
  - Co(re) scheduling: https://lwn.net/Articles/764482/
  - Secret Hiding

aws

State of Linux today

- Retpoline + IBRS on calls into firmware.
- IBPB on context switch between VMs or “sensitive” processes.
- Clear RSB on context switch and VMEXIT.
- Clear GPRs on kernel entry
- Flush dcache on kernel exit
- For Skylake+, pray to the deity of your choice.

aws

Xen

- IBRS supported for Xen entry if SKL+ or no retpoline.
- IBPB on context switch between VMs.
- Clear RSB on VMEXIT.
- Clear GPRs on Xen entry.
- No prayer required.

Application considerations

- IBRS / IBPB are kernel-only
- Use retpoline

Questions?

We’re hiring:
http://www.amazon.jobs/location/dresden-germany
http://www.amazon.jobs/location/bucharest-romania

aws