DEPARTMENT OF NETWORKED SYSTEMS AND SERVICES

COMPUTER ARCHITECTURES

Attacks against speculative execution

Gábor Horváth

BUTE Department of Networked Systems and Services
ghorvath@hit.bme.hu

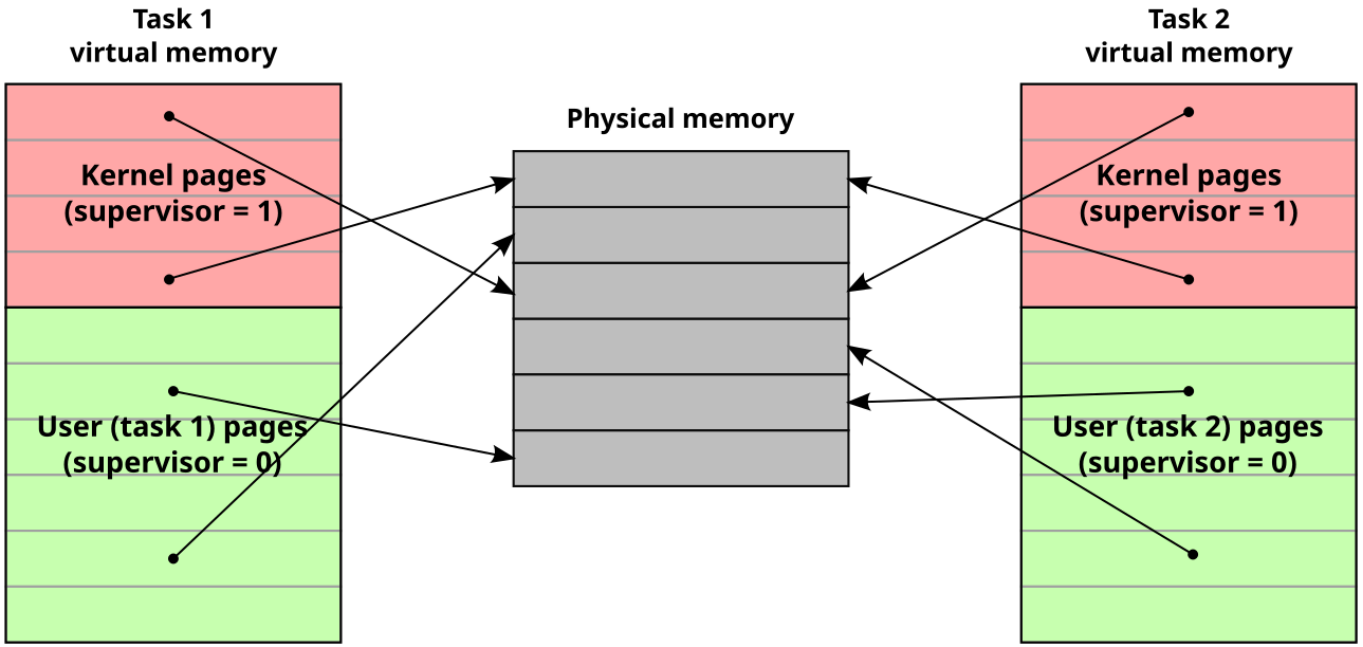
Budapest,
05/21/2025

MŰEGYETEM 1782

Ingredients

USER AND KERNEL PAGES

- Recall: each has its own page table
  - Task switching → Page table switching
- Also recall:
  - The page table of tasks contain part of the kernel address space, too
  - To enable fast execution of kernel calls
  - … These pages have „supervisor” bit=1



USER AND KERNEL PAGES

- User task:
  - Can access only its own pages
- Kernel:
  - Can access both the task's and the kernel's pager
- Kernel Call:
  - User task triggers a special interrupt
    - Puts the number of the kernel function to call into the EAX register
    - Linux: `INT 0x80`, Windows: `INT 0x2e`
  - On an interrupt, CPU always switches user → kernel mode
  - Kernel pages are now accessible
  - At the end of the kernel function it returns to user mode

SPECULATIVE EXECUTION

- Speculative execution = **executing instructions that might not be needed**
- Two main manifestations:
  - Branch prediction
    - Wrong prediction: instructions fetched by mistake are “invalidated”
  - Exception handling
    - At an exception, instructions whose execution started by mistake must be “invalidated”
- Meaning of “invalidation”:
  - Instructions that were executed by mistake are dropped
    - They do not affect the archtiectural state
  - … This is not always enough !!! They should disappear completely!
    - Microarchitectural state can not always be restored
    - Instruction might affect cache content, TLB and BTB content, etc.
    - Fortunately these are not visible from outside (sure?)

TIMING ATTACKS

- = side channel attack
- = Trying and measuring

- Craching length N passwords in just 26·N steps (instead of 26^N):
  - Let's try: Ax, Bx, Cx, …, Zx
  - Reasure response time
  - Response takes longer if we got the first letter correct
    - Strcmp has to compare the second letter, too!
  - We move on to the second letter, etc.

Meltdown

CONCEPT

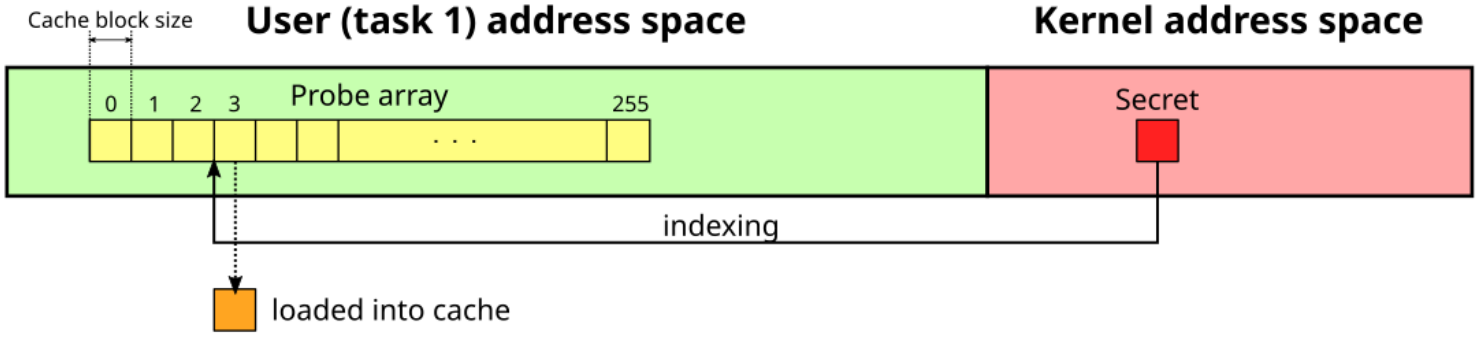
- The root of the problem:
  - Attacker exploits the speculative execution
  - Speculative instructions can leave microarchitectural traces
  - By leveraging this, kernel memory can be accessed

- Idea:
  - Let's try to access kernel data (a secret)
  - It shouldn't work (protection fault)
  - But the data access may have executed before the protection fault is detected
  - Of course, it will be invalidated, but the data remains in the cache
  - … and that's what we exploit!

IMPLEMENTATION

- Kernel secret we want to read: „secret”

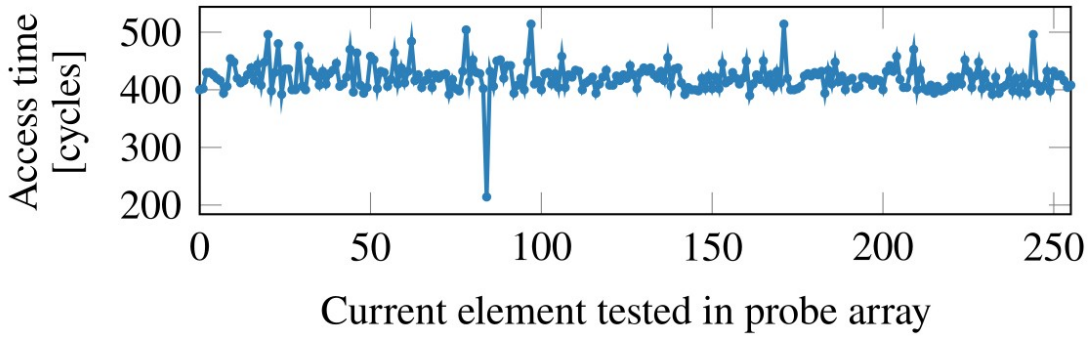
```
i1: R1 ← MEM[address of secret]
i2: R2 ← R1 * 64
i3: R3 ← MEM[probe + R2]
```



- i1 leads to protection fault, the program stops executing
- But the fault is only detected later
- i2, i3 are invalidated (R1, R2, R3 are not changed)
- **But the R1-th element of the probe array was loaded into the cache!**

IMPLEMENTATION

- **The R1-th element of the probe array was loaded into the cache!** (even though its content, of course, hasn't changed)
- The elements of the probe array are 64 bytes each
  → for every possible R1 value, a different block is loaded into the cache
- We can measure what R1 might have been!
- We go through the probe array and measure the access time:



→ The value of R1 was 84

MITIGATION

- Kernel memory can be dumped at speed 100-500 KB/s
- KPTI (KASLR, KAISER)
  - Kernel pages are not included in the page table of the task

    → Page tables are switched at each kernel call
  - Implies up to 30% overhead
  - AMD CPUs are not affected

FURTHER ATTACKS AGAINST SPECULATIVE EXECUTION

- Meltdown (2017):
  - Exploits how exception handling works
  - Accesses unauthorized data via the task's own page tables
  - Relatively easy to patch (≈)
- Spectre (2017):
  - Exploits speculative execution due to conditional branching
  - Goal: trick the CPU into speculatively executing attacker-controlled code in a foreign address space
    - Spectre v1: misprediction of a branch condition
    - Spectre v2: misprediction of a branch target (BTB – Branch Target Buffer – is deliberately "mistrained")
  - Gains access to secrets via a foreign task's page table
  - Very difficult to mitigate
- More and more speculative mechanisms are being found to be vulnerable!

DEPARTMENT OF
NETWORKED SYSTEMS
AND SERVICES

MŰEGYETEM 1782