

CSE 451: Operating Systems

Winter 2026

Module 19

Meltdown

Gary Kimura

Reading Kernel Memory from User Space

- Three parts
 - What is my cache? Pipelining and branch prediction.
 - Use timing to determine if something is in the cache or not.
 - A simple coding example to determine the value of a byte.
- A more complete description is in “Meltdown: Reading Kernel Memory from User Space”, Communications of the ACM, June 2020, Vol 63 No. 6, Pages 46-56.

Determining the value of a byte

- Consider this snippet of code to read a byte and use its value as an index into an array.

```
if (spec_cond) { // teach the HW that this is true
    unsigned char value = *(unsigned char *)ptr; // pointer to the secret
    maccess(&data[(value & 0xff)*0x100]); // data is in my address space (stride of 256)
}
```

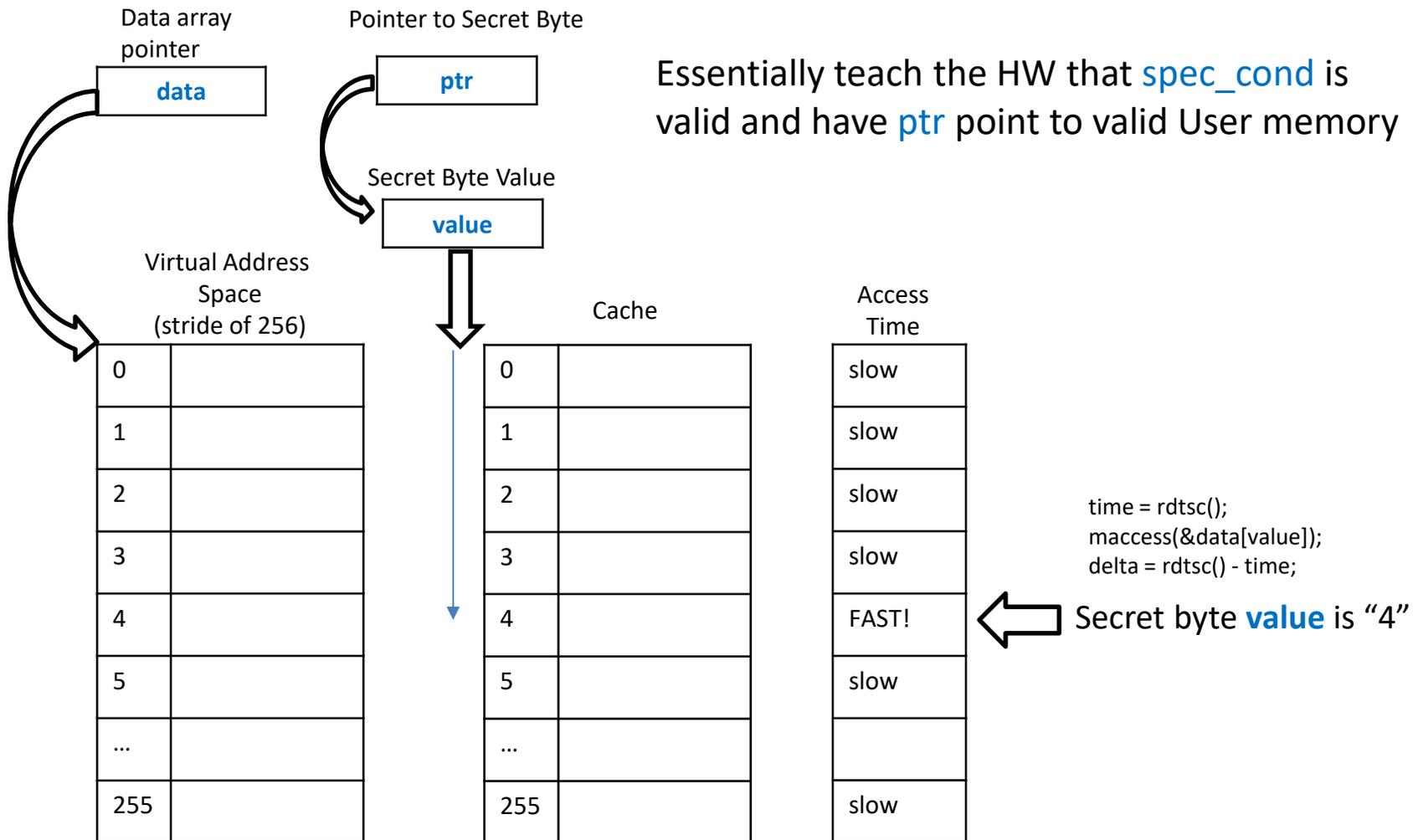
- Now determine what data item was accessed based on what is now in the cache. For each potential value record how long it takes to access the data.

```
time = rdtsc();
maccess(&data[value]);
delta = rdtsc() - time;
```

- The value with the fastest delta time exposes the secret.

Illustration

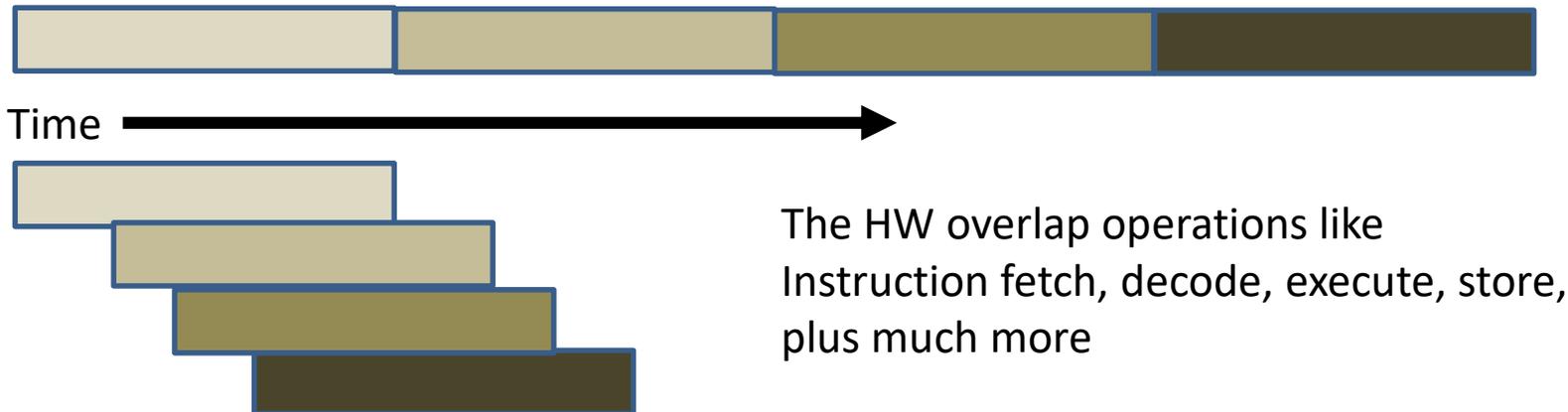
```
if (spec_cond) { // teach the HW that this is true
    unsigned char value = *(unsigned char *)ptr; // pointer to the secret
    maccess(&data[(value & 0xff)*0x100]); // data is in my address space (stride of 256)
}
```



What can Ptr point to?

- What if ptr is a legal address? – works fine
- What if ptr is an illegal address? – raise an exception

Instruction Pipeline and Branch Prediction



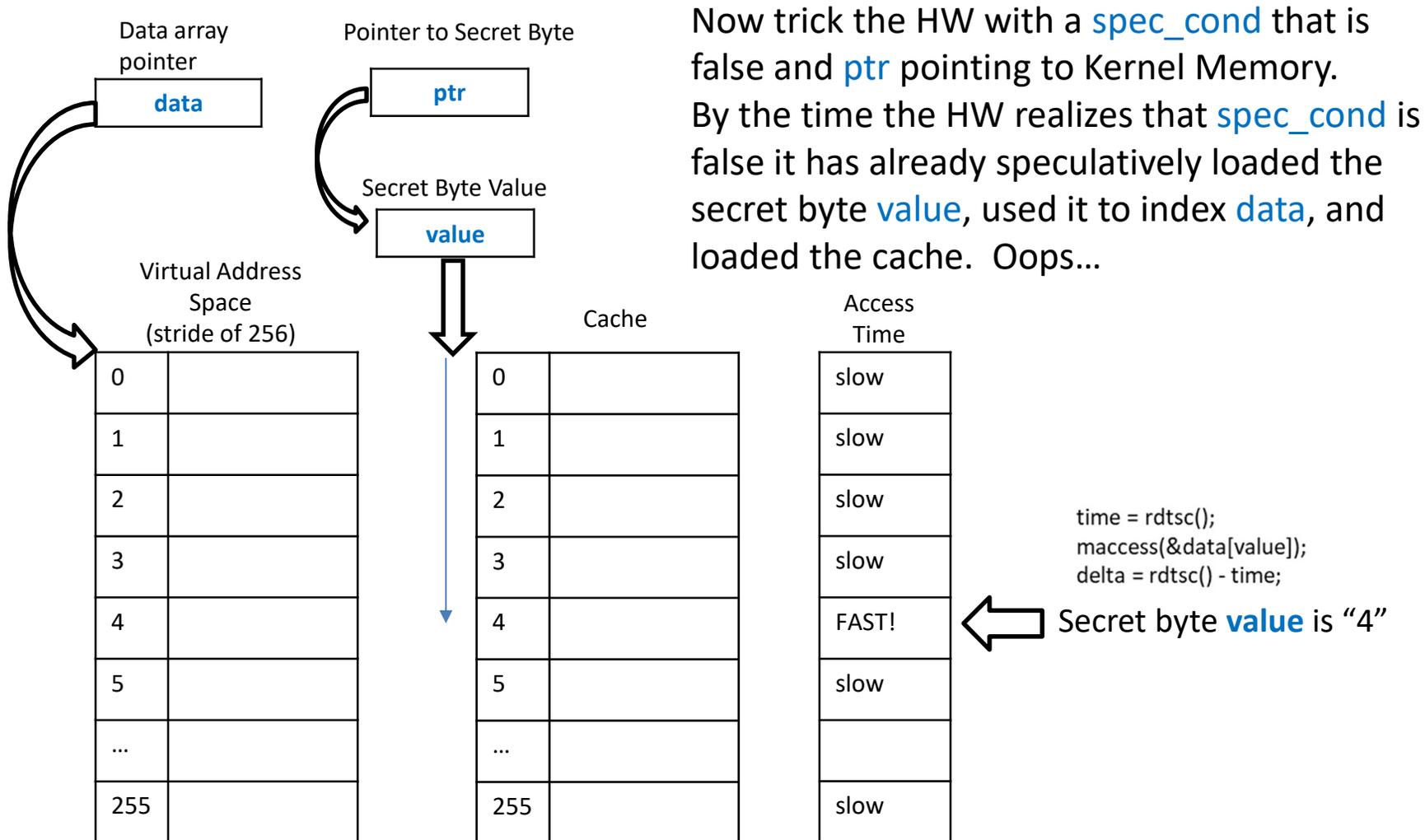
- This is a big optimization, if we can keep the pipeline full.
- What about branches? Unconditional and Conditional
- HW keeps track if a conditional branch is or isn't taken, and keeps the pipeline full on where it believes the branch will go.
- And if its prediction is wrong it stops executing the instructions in the pipeline.
- No harm because it doesn't allow those instructions to complete, however Meltdown exploited a side affect

Putting it all together

1. Teach the HW to predict the branch (may not be necessary if the HW automatically does pipelining across branches).
2. Flush the cache, e.g., CFLUSH().
3. “Trick” the HW to do load the cache based on the secret value. But don’t raise an exception now by having spec_cond be FALSE.
4. Scan through memory to see what part of data was loaded into the cache.
5. Rinse and repeat.

Illustration

```
if (spec_cond) { // teach the HW that this is true
    unsigned char value = *(unsigned char *)ptr; // pointer to the secret
    maccess(&data[(value & 0xff)*0x100]); // data is in my address space (stride of 256)
}
```



Possible fixes

Caches and Optimizations work great, except when they don't, and especially when they leak information!

1. Remove Cache, or branch prediction, or pipelining.
2. Don't rely on protection bit in the PTE to stop users from accessing kernel data, i.e., redo how the Operating System uses page tables.

Now a word about updates

How do we update the operating system?

FRANK 'N' ERNEST

