Meltdown and Spectre - understanding and mitigating the threats

Gratuitous vulnerability logos

Jake Williams
@MalwareJake

SANS / Rendition Infosec
sans.org / rsec.us
@RenditionSec

RENDITION INFOSEC
Security by any legal means

The sky isn't falling!

Before we start, it's important to understand that while this is bad, the sky isn't falling

Media reports will likely seek to sensationalize these vulnerabilities

There are actions you can take to minimize exposure, particularly for critical workloads

Agenda

What is Meltdown?

What is Spectre?

Exploitation scenarios

How are these vulnerabilities alike (and different)

Exploit mitigations (there’s more to this than “patch”)

Closing thoughts

Meltdown

The coolest thing to happen to processor geeks since… forever.

Meltdown – the basics

Meltdown allows attackers to read arbitrary physical memory (including kernel memory) from an unprivileged user process

Meltdown uses out of order instruction execution to leak data via a processor covert channel (cache lines)

Meltdown was patched (in Linux) with KAISER/KPTI

Kernel ASLR (Address Space Layout Randomization)

Linux implements kernel ASLR by default since 4.12

The 64-bit address space is huge, you wouldn’t want to dump the whole thing

- 16EB theoretical limit, but 256TB practical limit

Randomization is limited to 40 bits, meaning that locating kernel offsets is relatively easy

Kernel ASLR (Address Space Layout Randomization)

Windows ASLR isn’t much different in that not all of the kernel is randomized

Because of the way the Windows memory manager is implemented, it is unlikely that the entirety of physical memory is mapped into a single process

Verdict: On an unpatched Windows system, most (but not all) kernel memory can be read from Windows

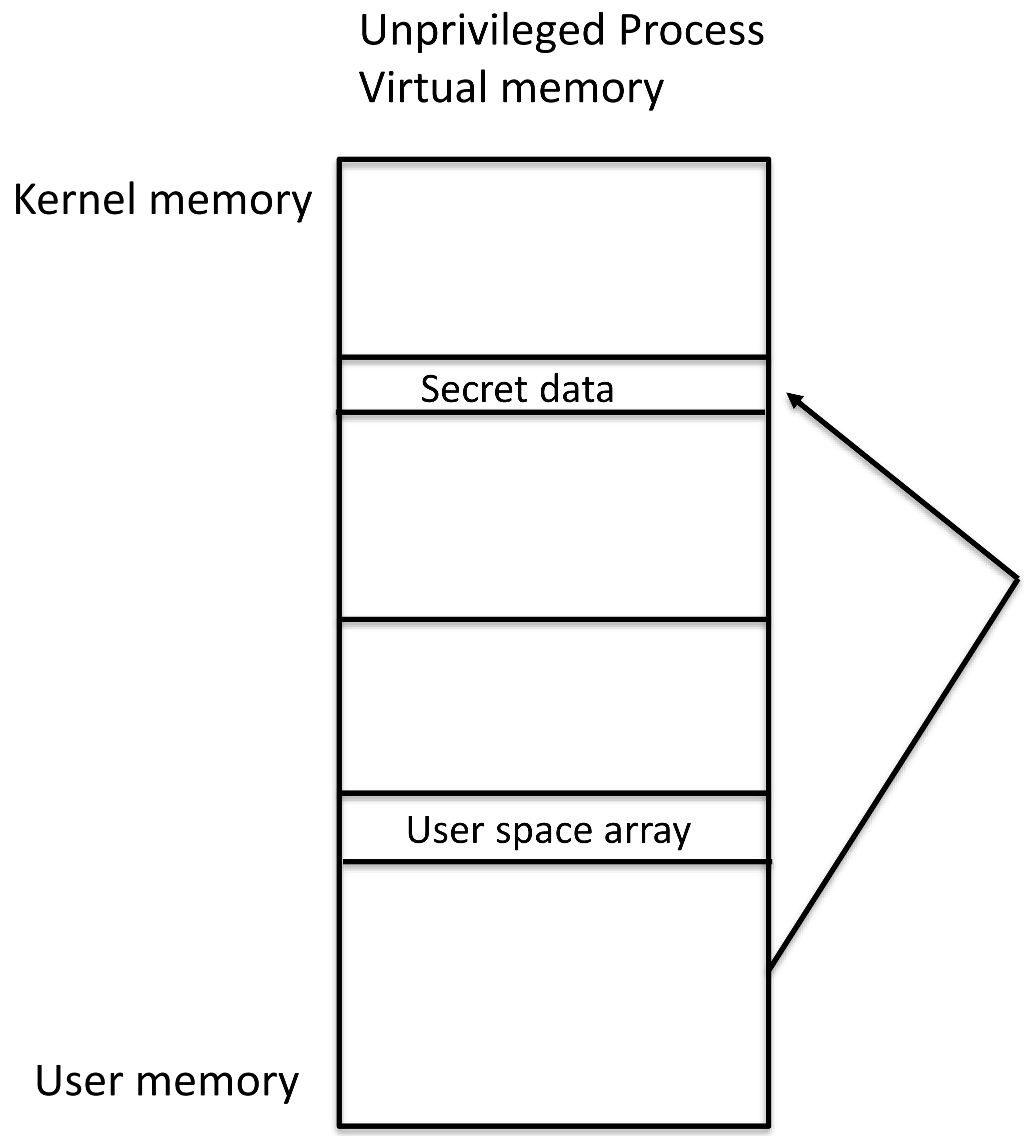
Page Tables (User and Kernel)

Page tables contain the mappings between virtual memory (used by the process) and physical memory (used by the memory manager)

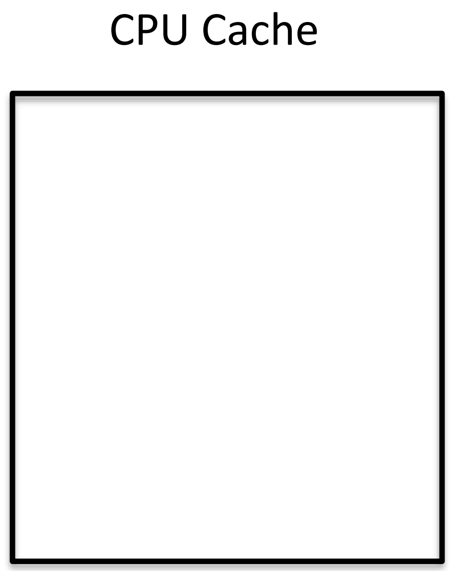
For performance reasons, most modern OS's map kernel addresses into user space processes

- Under normal circumstances, the kernel memory can't be read from user space, an exception is triggered

Meltdown attack



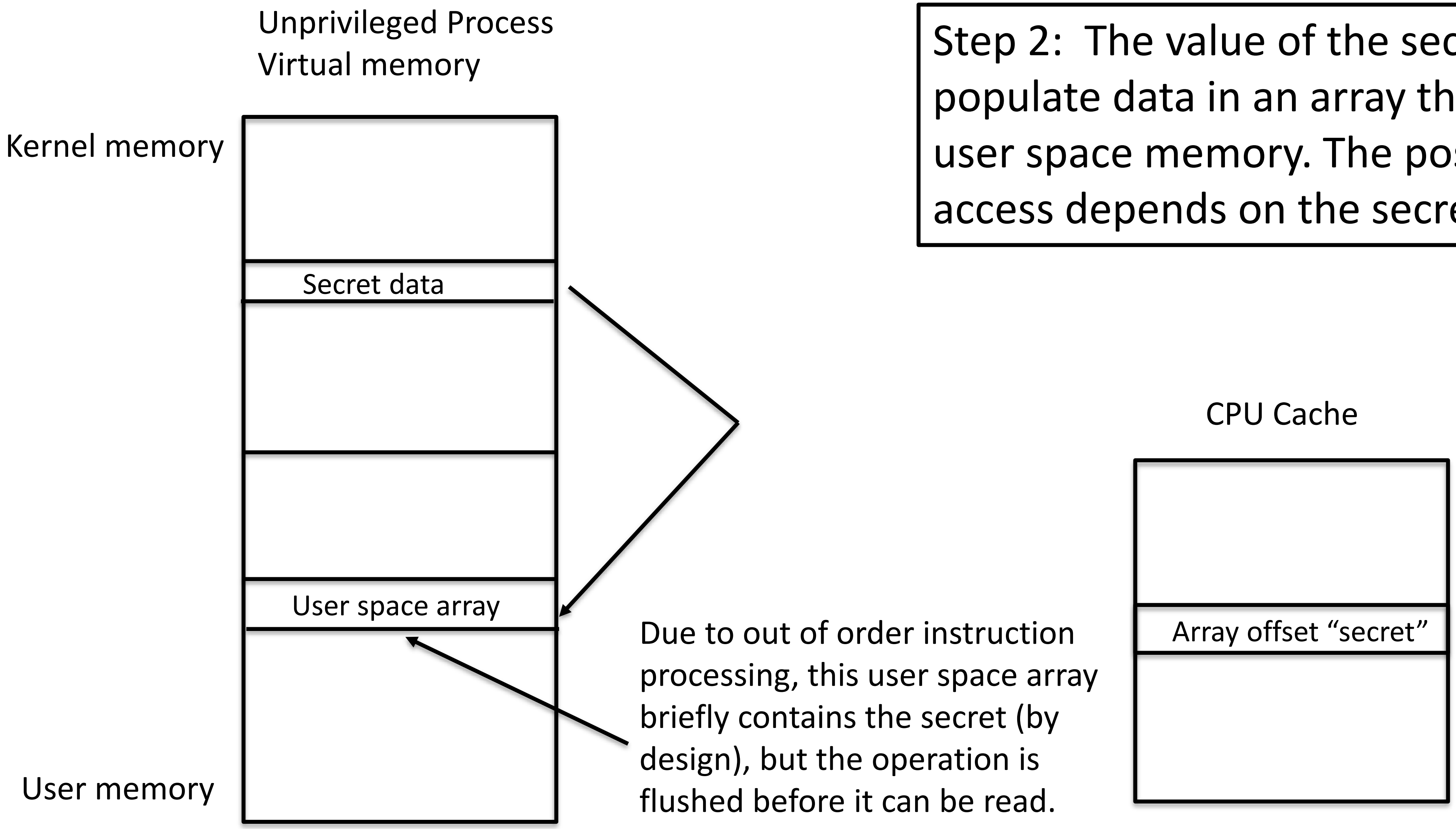
Step 1: A user process reads a byte of arbitrary kernel memory. This should cause an exception (and eventually will), but will leak data to a side channel before the exception handler is invoked due to out of order instruction execution.



Clear the elements of the user space array from the CPU cache.

Meltdown attack (2)

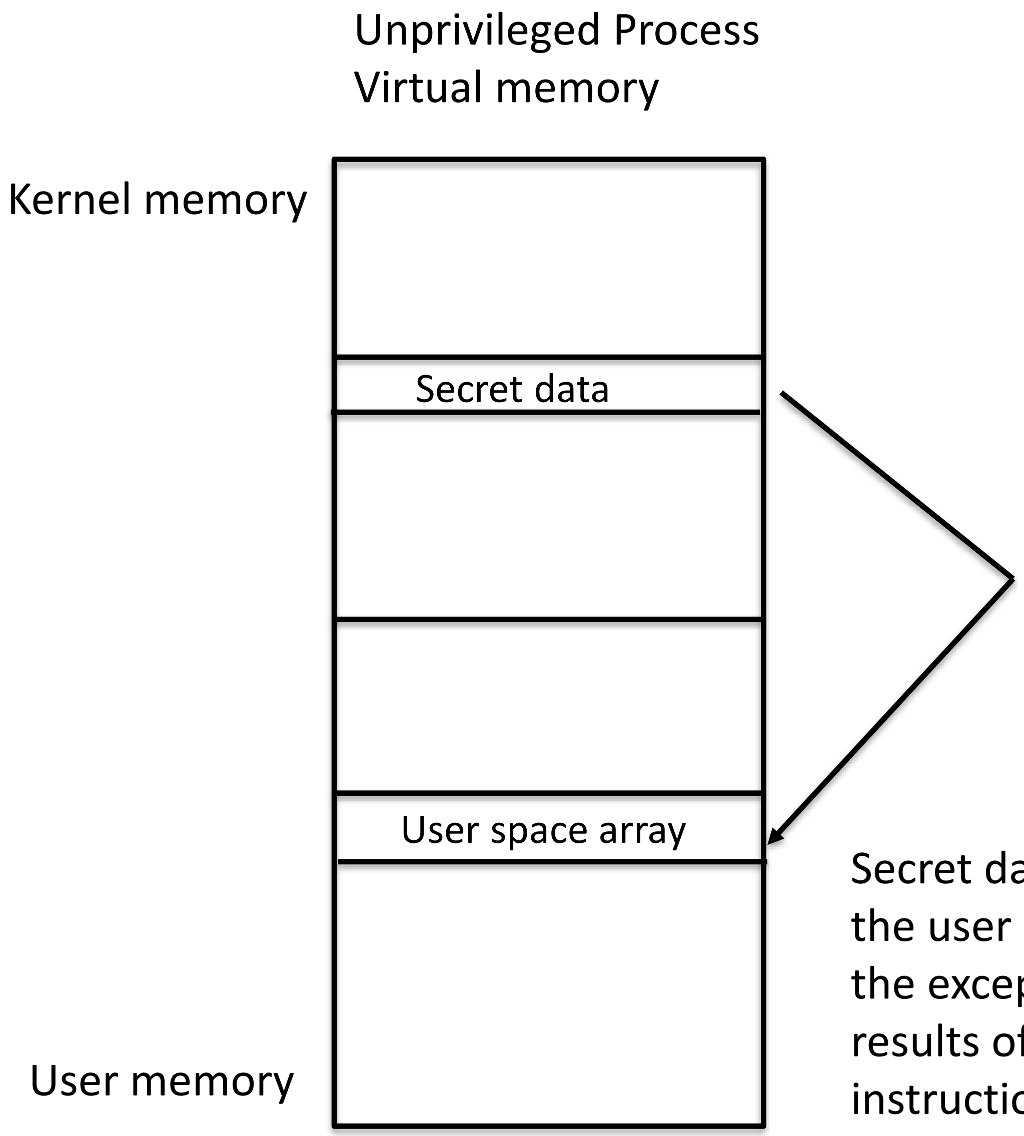
Step 2: The value of the secret data is used to populate data in an array that is readable in user space memory. The position of the array access depends on the secret value.



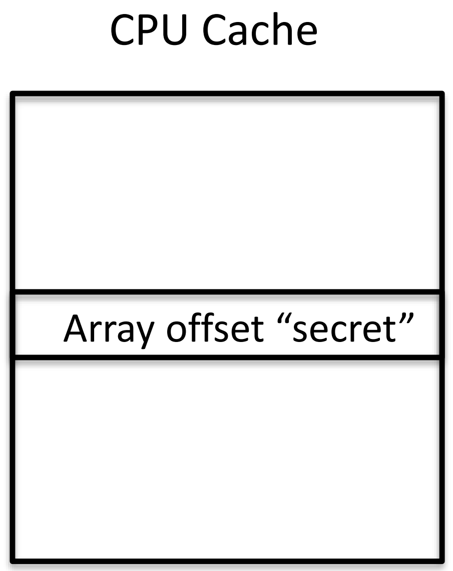
Due to out of order instruction processing, this user space array briefly contains the secret (by design), but the operation is flushed before it can be read.

Meltdown attack (3)

Step 3: An exception is triggered that discards the out of order instructions. The secret cannot be read from the user space array

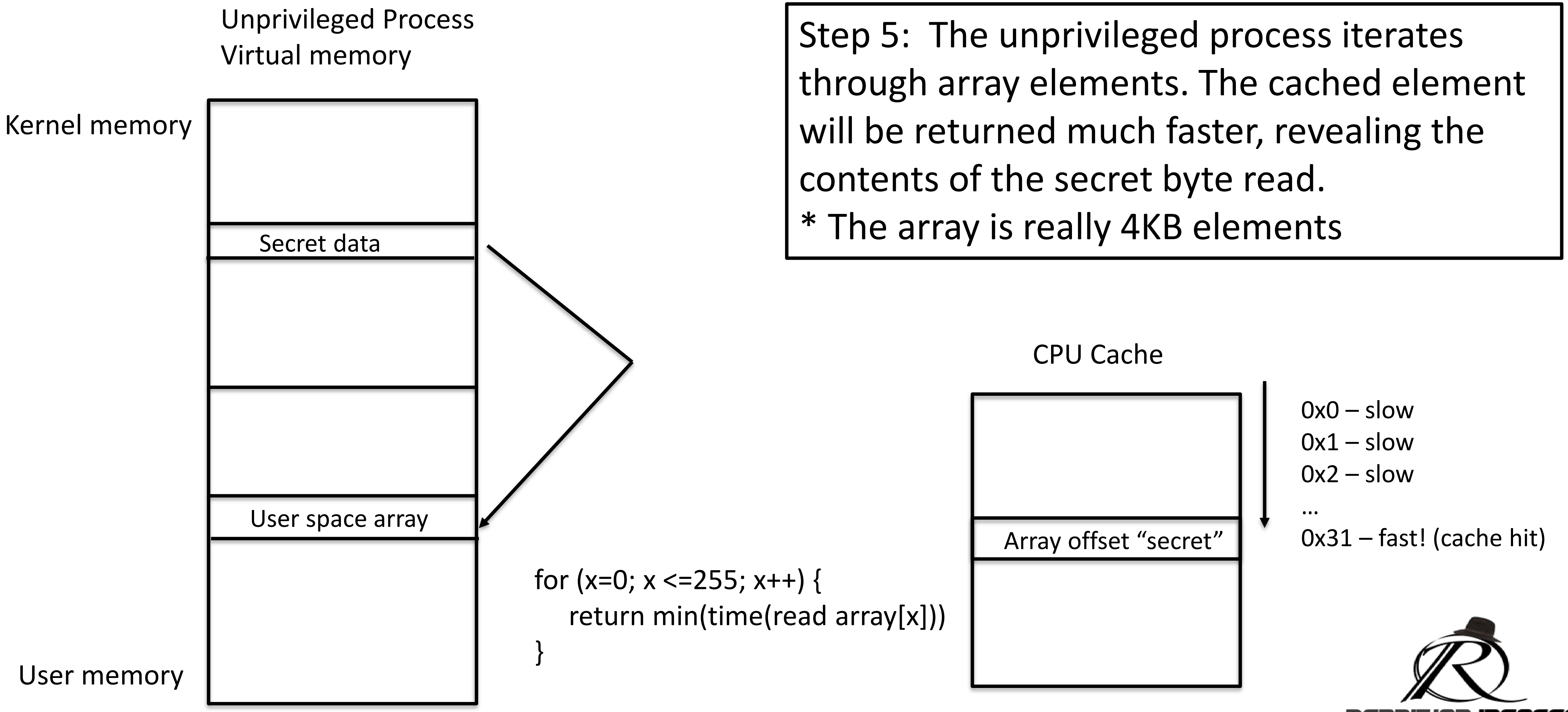


Secret data is never available in the user accessible array since the exception discards the results of the out of order instruction computations.



Meltdown attack (4)

Step 5: The unprivileged process iterates through array elements. The cached element will be returned much faster, revealing the contents of the secret byte read.
* The array is really 4KB elements



```
for (x=0; x <=255; x++) {
    return min(time(read array[x]))
}
```

Kernel Page Table Isolation

Kernel page table isolation (aka KPTI, aka the KAISER patch) removes the mapping of kernel memory in user space processes

Because the kernel memory is no longer mapped, it cannot be read by Meltdown

- This incurs a non-negligible performance impact

Kernel Page Table Isolation (2)

Technically, some kernel memory (e.g. interrupt handlers) must be mapped into user space processes

Future research will involve determining if leaking these small remnants of kernel memory can be used to dump other offsets in kernel memory

The patch **does not** address the core vulnerability, it simply prevents practical exploitation

Why was there a rumor this was Intel only?

Modern intel CPUs implement TSX, allowing for hardware transactional memory operations

- e.g. Grouping instructions for “all or nothing” execution

This allows the Meltdown attack to be performed without software exception handling routines

Are ARM and AMD processors impacted?

Meltdown exploitation is theoretically possible on both ARM and AMD, but the authors note that no practical exploitation was achieved

They note that this may be due to the need for optimization of their code – experiments confirm out of order execution is definitely occurring

Spectre

Forget what I said about Meltdown, this might be cooler…

Spectre – the basics

Spectre abuses branch prediction and speculative execution to leak data from via a processor covert channel (cache lines)

Spectre can only read memory **from the current process**, not the kernel and other physical memory

Spectre **does not** appear to be patched

Speculative Execution

Modern processors perform speculative execution

They execute instructions in parallel that are likely to be executed after a branch in code (e.g. if/else)

Of course these instructions may never really be executed, so a sort of CPU snapshot is taken at the branch so execution can be "rolled back" if needed

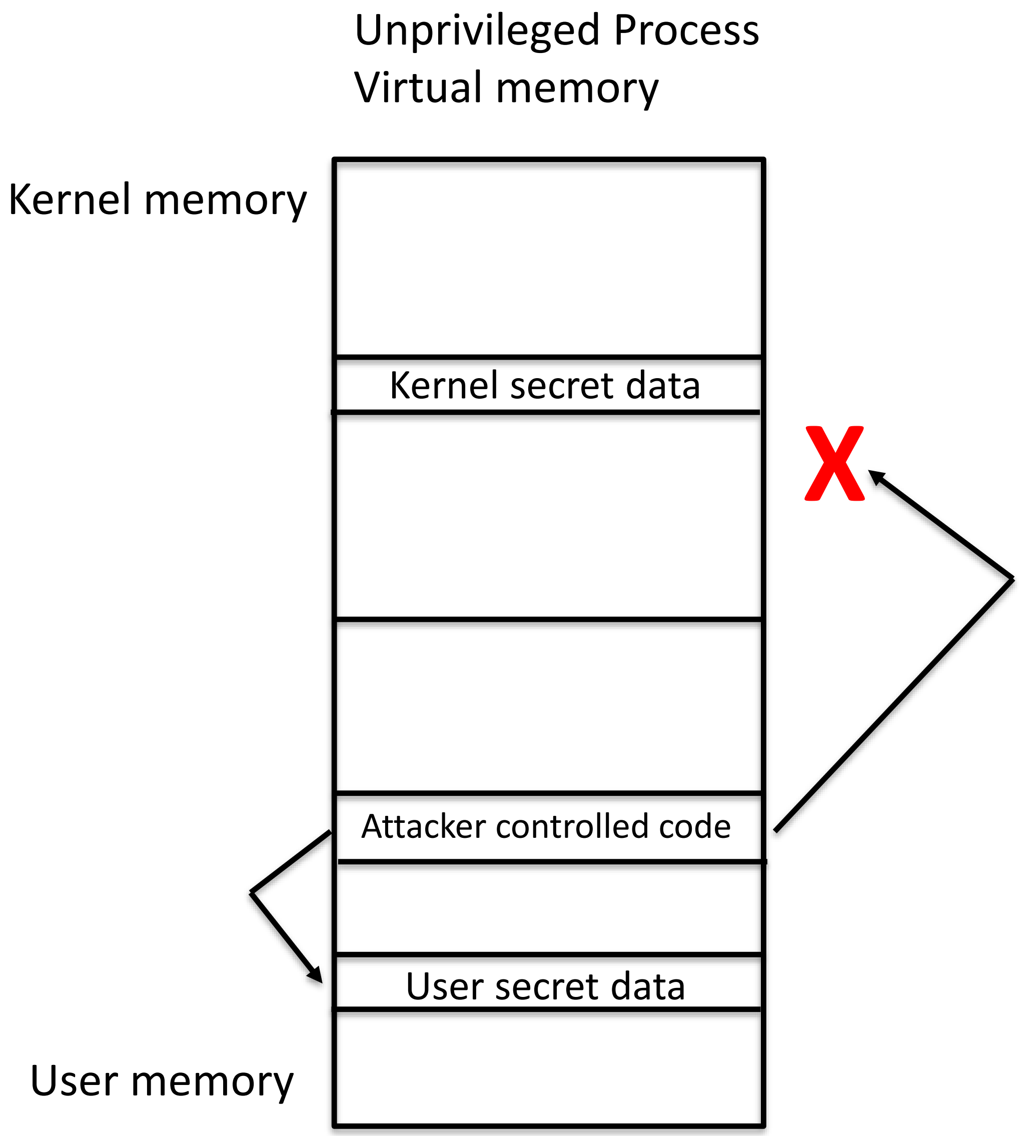
Branch Prediction

How does the CPU know which side of a branch ("if/else") to speculatively execute?

Branch prediction algorithms are trained based on current execution

The CPU "learns" which branch will be executed from previous executions of the same code

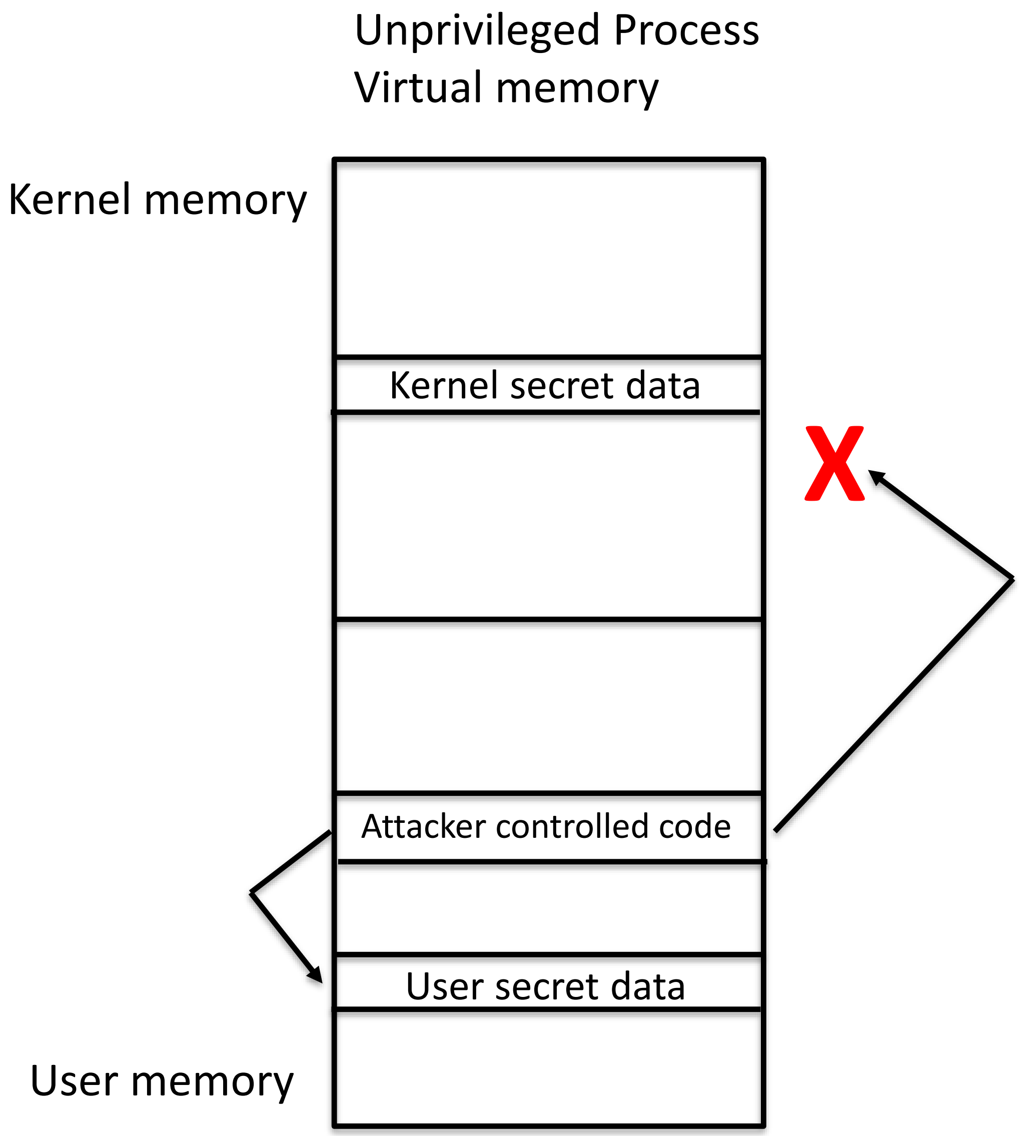
Spectre attack



The Spectre vuln does not allow an unprivileged process to read privileged memory (as we saw with Meltdown).

Spectre does allow code executing in the victim process to access data it should not have access to (e.g. outside of a JavaScript sandbox).

Spectre attack (2)



Spectre is most likely to be exploited in applications that allow users to run some code in a sandbox. Spectre will allow the attacker to escape the sandbox and leak data from elsewhere in the process.

This is most useful in a browser where one tab may contain attacker code while another tab contains sensitive information that should not be accessible to the attacker.

Isolating each tab in its own process would mitigate this type of attack.

Exploit Scenarios

How are attackers most likely to use Spectre and Meltdown?

Meltdown attacks

At this time we believe there are two primary uses for Meltdown:

1. Privilege Escalation
2. Container/Paravirtualization Hypervisor Escape

Meltdown - Privilege Escalation

On any unpatched system if an attacker can execute a process they can dump all (or most) physical memory

With physical memory, attackers could identify password hashes, execute a mimikatz style attack on Windows, or find private keys

Sure, KASLR is also bypassed (but who really cares)

Meltdown – Container Escape

Meltdown may target kernel addresses that are shared between the container and host kernel in many paravirtualization instances (e.g. Xen) and kernel sandboxes (e.g. Docker)

It is possible that attacker may leak data from the outside the container, leading to a hypervisor escape

Spectre – Exploit Scenarios

The primary exploit scenario we see for Spectre is JavaScript execution in the browser being used to read outside of the browser sandbox

There are two probable uses for this:

1. Leaking secret data from browser memory outside the JavaScript sandbox
2. Leaking addresses of user space modules to bypass ASLR (facilitating remote code execution)

Spectre – Leaking Browser Memory

The web browser contains all sorts of interesting stuff you probably don’t want other sites to be able to read

Using JavaScript (perhaps in an advertisement), Spectre attacks could be used to leak browser cache or other saved data that pertains to other sites

- I’m particularly worried about session keys for active session (this completely bypasses MFA)

Spectre – Leaking Module Addresses

A large number of browser vulnerabilities are not practically exploitable because of user space ASLR

- ASLR and DEP have substantially limited browser exploitation

Spectre can be used to determine the address of a module in memory and bypass ASLR (ushering in the new age of practical browser exploitation)

Meltdown vs Spectre

Cage match!
Two vulnerabilities enter,
All your data leaves… ☹

Meltdown vs. Spectre

	Meltdown	Spectre
Allows kernel memory read	Yes	No
Was patched with KAISER/KPTI	Yes	No
Leaks arbitrary user memory	Yes	Yes
Could be executed remotely	Sometimes	Definitely
Most likely to impact	Kernel integrity	Browser memory
Practical attacks against	Intel	Intel, AMD, ARM

RENDITION INFOSEC
Security by any legal means

Spectre and Meltdown Mitigations

This is more than (just) patching…

Patch – but patch carefully

In Linux, KPTI has an obvious performance impact

- Testing suggests less than 10%

Patching on Windows may not actually patch if you're running third party antivirus products

- Sadly, this isn't a joke…

Reports are that applying patches may BSOD

Kevin Beaumont (@GossiTheDog) has compiled a list of antivirus patch statuses

- http://bit.ly/MeltdownPatchCompat

Symantec is known to cause a BSOD

- Patch to the AV engine is expected tomorrow

Consider systems that won’t be patched

Many orgs are running legacy systems that can’t be patched (e.g. Win2k3)

For these systems, consider whether it is prudent to allow multi-user access if critical workloads are being maintained there

What about cloud?

Most (all?) major cloud hosting providers patched before the vulnerability was publicly known

- If you’re on AWS, Azure, etc. you should be good

Semi-private/smaller cloud providers are what scare me – they didn’t get the embargo information like the big guys did

Browser memory isolation

- On Chrome, you can enable site isolation – I can't think of a good reason not to do this
  - http://bit.ly/ChromeSiteIsolation
  - Thanks for @HackerFantastic for this recommendation
- This causes Chrome to load each site into its own process so even if same-origin policy is bypassed you can't steal data from another site
  - This isn't 100% safe, cache data is probably still in memory

This probably won’t be the last hardware bug

Expect to see variations on these attacks for some time to come

Multiple different researchers independently found these vulns while they were under embargo

Architect your networks expecting more vulns of this type to be discovered

Closing thoughts

A few thoughts as we close out the webcast

Closing thoughts

Attacks that impact microarchitecture of CPUs have been known for more than a decade

Most were thought to be only exploitable in very limited cases, many involving physical access

Spectre and Meltdown attacks make it clear that CPU architecture decisions need to be rethought

Closing thoughts (2)

In the long term, you should expect to see more attacks on CPU microarchitecture

This particularly will impact multi-tenant environments

Spectre and Meltdown offer true wakeup calls for those running critical workloads in shared environments (e.g. the cloud)

The End - Meltdown vs. Spectre

Thanks for attending! Please let us know if this webcast helped you get a handle on these vulnerabilities.

	Meltdown	Spectre
Allows kernel memory read	Yes	No
Was patched with KAISER/KPTI	Yes	No
Leaks arbitrary user memory	Yes	Yes
Could be executed remotely	Sometimes	Definitely
Most likely to impact	Kernel integrity	Browser memory
Practical attacks against	Intel	Intel, AMD, ARM