



MELTDOWN



SPECTRE

# MELTDOWN & SPECTRE

Presentado por:  
Eduardo Valdez  
Sistemas Operativos  
Grupo 2  
Semestre 2018-2

## MELTDOWN & SPECTRE

- Antecedentes históricos
  - ¿Qué son?
- ¿Qué *software/hardware* afecta?
  - ¿Cómo funcionan?
  - Mitigación

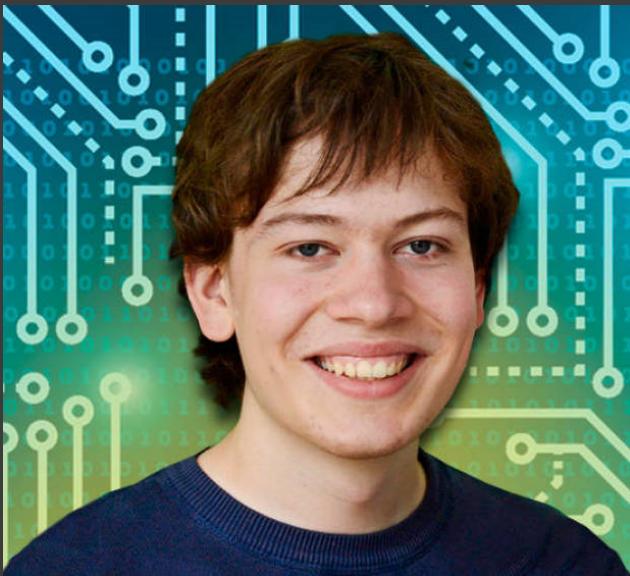


MELTDOWN

- Antecedentes históricos

El 8 de mayo de 1995, un documento titulado: *<<La arquitectura del procesador Intel 80x86: trampas para los sistemas seguros>>* y publicado en el simposio de 1995 del IEEE referente a seguridad y privacidad advertía de un canal de temporización encubierto en la caché de la CPU.

- Antecedentes históricos



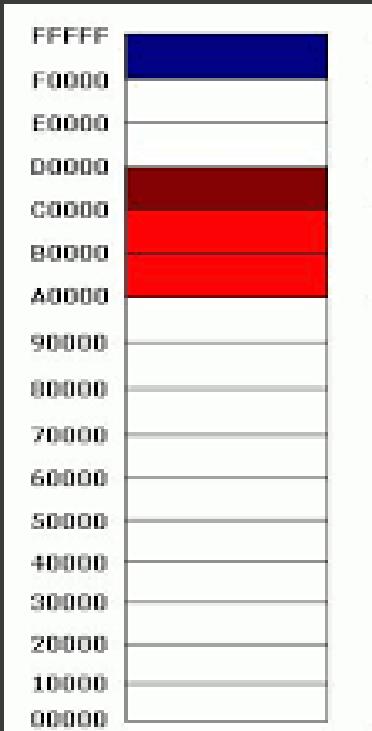
La vulnerabilidad *Meltdown* fue descubierta de forma independiente por Jann Horn del Proyecto Cero de Google, así como por otros investigadores de distintas universidades.

- ¿Qué es *Meltdown*?

*Meltdown* es una vulnerabilidad en la arquitectura de los microprocesadores modernos que explota, principalmente, los efectos secundarios de la ejecución fuera de orden

- ¿Qué es *Meltdown*?

*Meltdown* no explota ninguna vulnerabilidad de *software*, si una de *hardware*, ya que explota la información de canal lateral disponible en la mayoría de los microprocesadores modernos.



Esto permite a un atacante que pueda ejecutar código en el microprocesador vulnerable obtener un volcado completo del espacio de direcciones del *kernel*, incluyendo cualquier memoria física mapeada.

- ¿Qué software/hardware afecta?

Intel introdujo la ejecución especulativa en sus microprocesadores a partir de la familia P6 con el microprocesador Pentium Pro IA-32 en 1995.

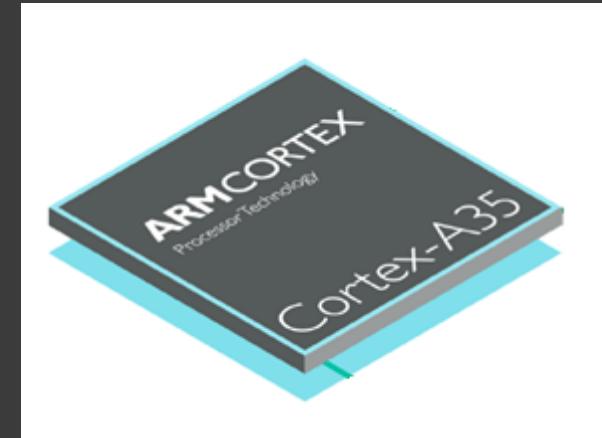
Google ha manifestado que todos los microprocesadores de Intel desde 1995 con ejecución fuera de orden son potencialmente vulnerables a *Meltdown*.



Se excluye los microprocesadores Itanium, así como los Intel Atom de antes de 2013.

- ¿Qué software/hardware afecta?

Una gran parte de los teléfonos móviles Android actuales de gama media utilizan *Cortex-A53* y *Cortex-A55* de 8 núcleos y no están afectados ni por *Meltdown* ni por *Spectre*, ya que no realizan ejecución fuera de orden.



Esto incluye también los sistemas con chip *Snapdragon 630*, *Snapdragon 626* y *Snapdragon 625* de *Qualcomm*, así como todos los microprocesadores *Snapdragon 4xx* basados en núcleos A53 y A55.



- ¿Qué software/hardware afecta?

Los sistemas *Raspberry Pi* tampoco son vulnerables a *Meltdown* ni a *Spectre*.



- ¿Cómo funciona *Meltdown*?

Las CPU de los actuales equipos informáticos emplean una variedad de técnicas para proporcionar altos niveles de seguridad y eficiencia.

En el caso de *Meltdown*, hay 4 particularidades que resultan especialmente relevantes:

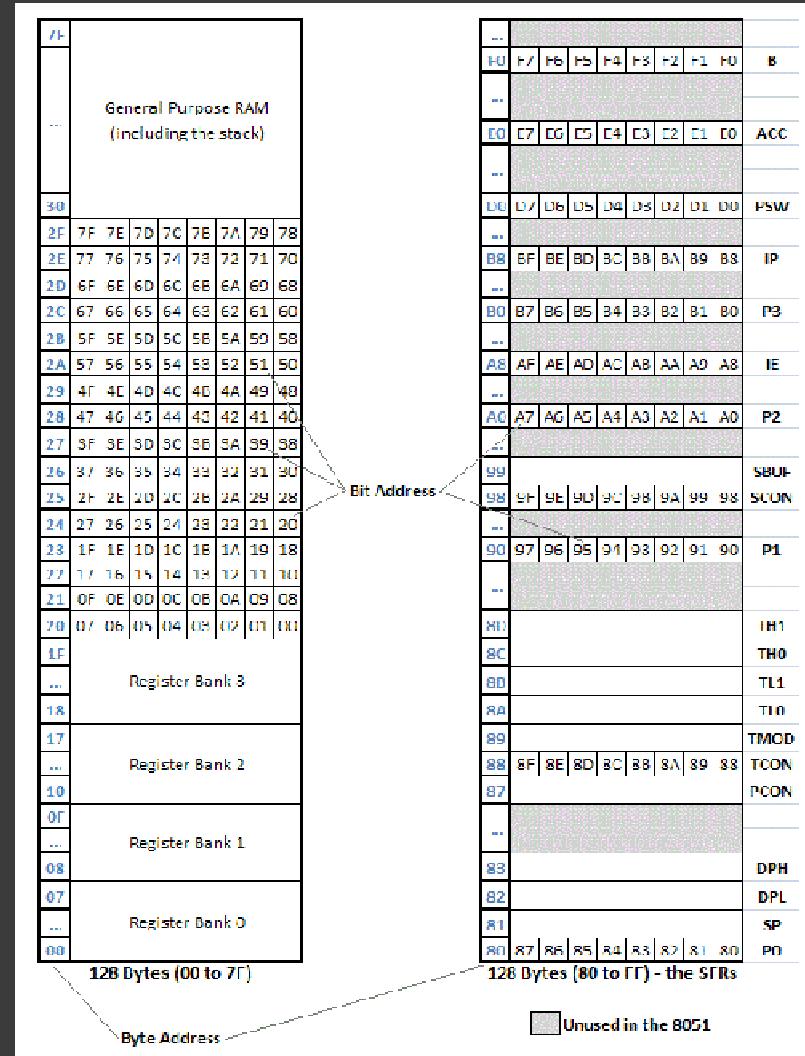


- ¿Cómo funciona *Meltdown*?

**Memoria virtual (paginada), también conocida como mapeo de memoria.**

- ¿Cómo funciona Meltdown?

Se utiliza para que los accesos a la memoria resulten más eficientes, y para controlar qué procesos pueden acceder a determinadas áreas de la memoria.

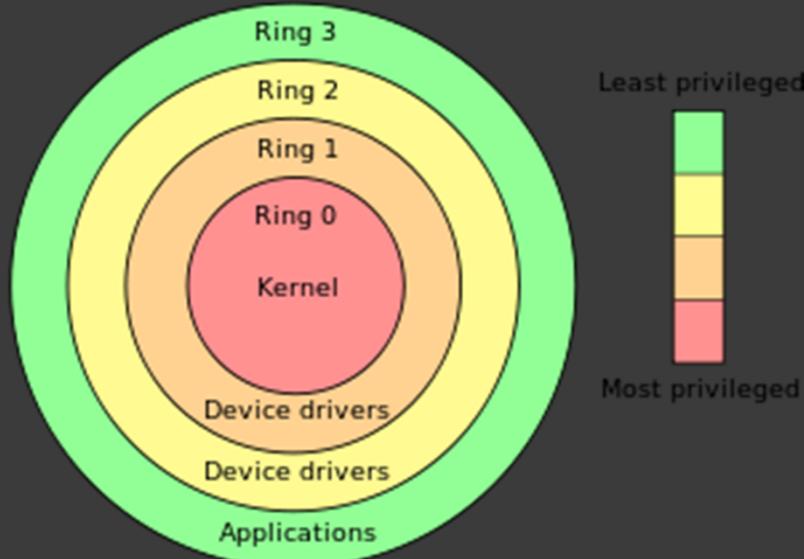


- ¿Cómo funciona *Meltdown*?

**Los dominios de protección o anillos de protección.**

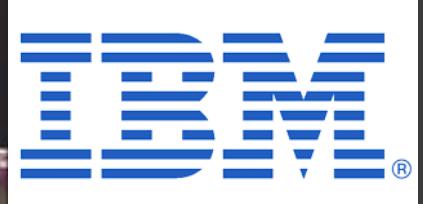
- ¿Cómo funciona *Meltdown*?

Proporcionan un medio por el cual el sistema operativo puede controlar qué procesos están autorizados a leer determinadas áreas de la memoria virtual.



- ¿Cómo funciona *Meltdown*?

En 1967, Robert Tomasulo desarrolló un algoritmo que permitía la programación dinámica de instrucciones para permitir la ejecución fuera de orden.

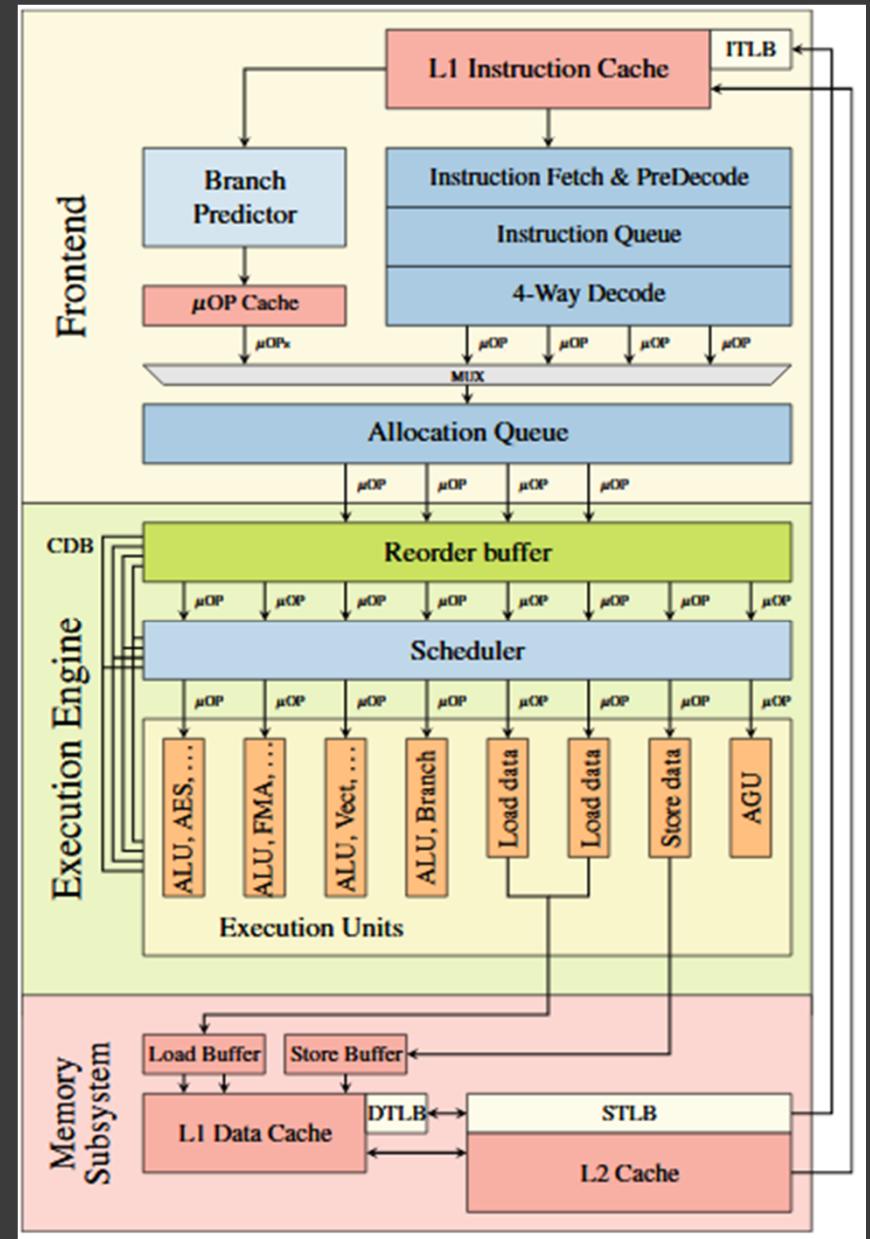


- ¿Cómo funciona *Meltdown*?

En la arquitectura Intel, la segmentación (*pipeline*) consta del *Front-end*, el motor de ejecución *Execution Engine* (*back-end*) y el subsistema de memoria (*Memory Subsystem*).

Las instrucciones x86 se obtienen por el *Front-end* de la memoria y se decodifican en micro-operaciones ( $\mu\text{OP}$ ) que se envían continuamente al motor de ejecución *Execution Engine* (*back-end*).

La ejecución fuera de orden se implementa dentro del motor de ejecución *Execution Engine* (*back-end*).



- ¿Cómo funciona *Meltdown*?

**Segmentación de instrucciones y ejecución especulativa**

- ¿Cómo funciona *Meltdown*?

Se utiliza para permitir que las instrucciones se ejecuten del modo más eficiente posible —si es necesario, permitiéndoles correr fuera de orden o en paralelo a través de las múltiples unidades de procesamiento de la CPU— siempre que el resultado final sea correcto.



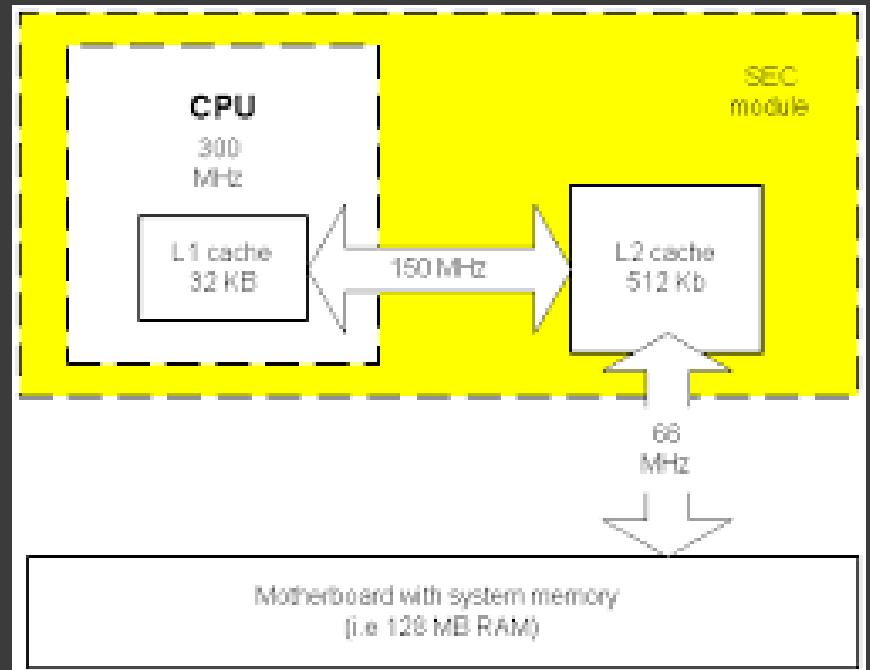
Esto maximiza la eficiencia al hacer que se usen al máximo todas las unidades de ejecución del microprocesador.

- ¿Cómo funciona *Meltdown*?

## La caché de la CPU

- ¿Cómo funciona *Meltdown*?

Es una pequeña cantidad de memoria alojada en la CPU que se utiliza para garantizar el alto rendimiento del microprocesador, acelerar los accesos a memoria y facilitar la ejecución de instrucciones de un modo eficiente.



- ¿Cómo funciona *Meltdown*?

Por lo general, los mecanismos anteriormente descritos se consideran seguros, y proporcionan la base sobre la que se fundamentan la mayoría de sistemas operativos y microprocesadores modernos.



*Meltdown* se aprovecha de la forma en que interactúan estas mecánicas, para saltarse los controles de privilegios fundamentales de la CPU y poder así acceder a información sensible del sistema operativo y de otros procesos.

- ¿Cómo funciona *Meltdown*?

AHORA SI,

¿CÓMO FUNCIONA?

- ¿Cómo funciona *Meltdown*?

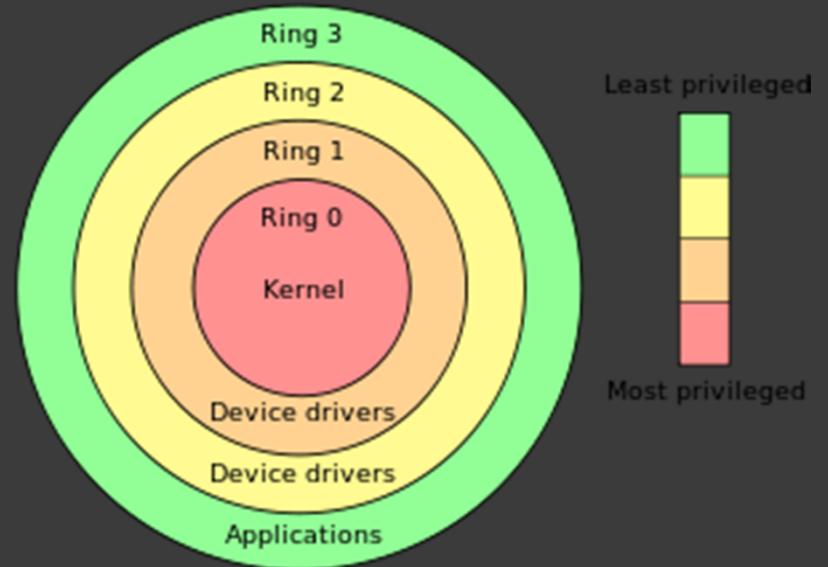
1.- Para una lectura no autorizada, la unidad de ejecución recibirá el dato de que la instrucción no ha superado la comprobación de privilegios, y abandonará la instrucción para proceder a atender a la siguiente.

Este procedimiento es completamente seguro.

- ¿Cómo funciona *Meltdown*?

2.- En los primeros estados de la ejecución de la instrucción, el planificador de la CPU consideró dos eventos:

- Una comprobación de privilegios
- Los primeros pasos de la ejecución de la instrucción.



Mientras estaba esperando a que la comprobación de privilegios se completase, la unidad de ejecución empezó su trabajo y solicitó la información.

- ¿Cómo funciona *Meltdown*?

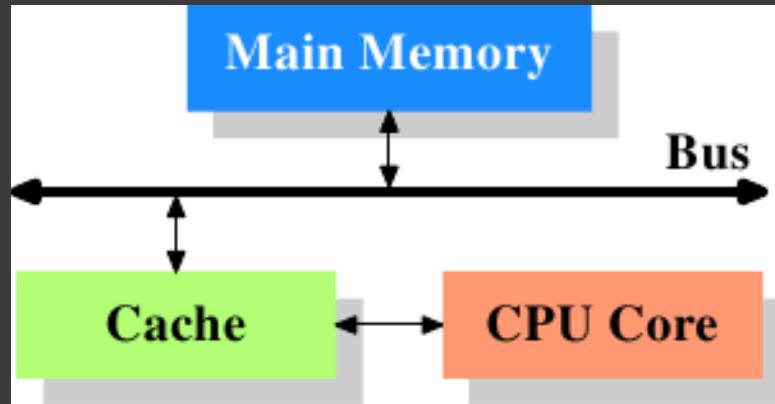
Esto no tiene efecto alguno y la seguridad está garantizada, puesto que la información leída nunca se pone a disposición de otros procesos hasta completarse la comprobación de privilegios.



Sin embargo, la información ya ha sido solicitada por la unidad de ejecución y obtenida por el controlador de memoria con el fin de estar listo para procesarla, y si bien la unidad de ejecución descarta la información al fallar la comprobación de privilegios, pero...

- ¿Cómo funciona *Meltdown*?

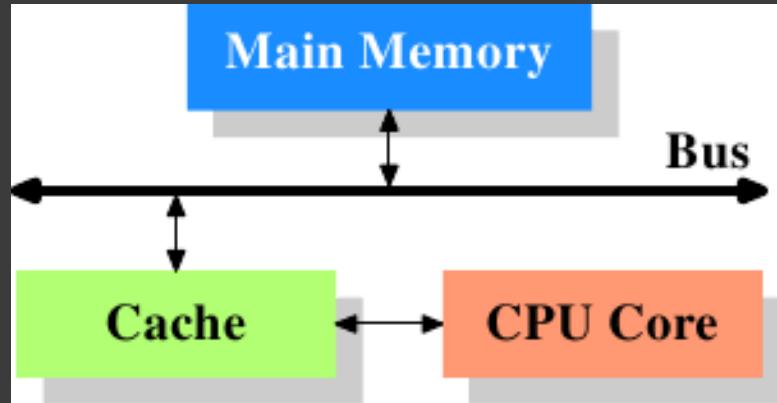
¡La caché de la CPU, de hecho, se actualizó!



Algo habitual al leer información de la memoria, y se actualiza por si esa información fuese necesaria una segunda vez.

- ¿Cómo funciona *Meltdown*?

3.- La caché de la CPU no es legible por parte de un proceso no autorizado, puesto que es algo interno de la CPU.



Pero, usando un ataque coordinado a la caché “una forma de ataque de canal lateral”, resulta posible para un proceso maligno determinar si la información de una dirección específica se encuentra en la caché de la CPU.

- ¿Cómo funciona *Meltdown*?

El proceso maligno puede utilizar esta diferencia temporal para detectar cuál de ambos casos ha tenido lugar, y saber así si la dirección ya estaba en la caché de la CPU.

- ¿Cómo funciona *Meltdown*?

Meltdown emplea esta técnica de forma secuencial para leer cualquier dirección que le interese a alta velocidad, y dependiendo de cuáles sean los otros procesos concurrentes.



El resultado puede contener contraseñas, información encriptada y cualquier otro tipo de información sensible, de cualquier dirección de cualquier proceso que exista en su mapa de memoria.

- Mitigación

Daniel Gruss propuso KAISER, una modificación que su objetivo es evitar ataques de canal lateral, y por tanto también evita *Meltdown*, ya que garantiza que no haya una asignación válida para el espacio del *kernel*.



KAISER estará disponible en las próximas versiones del *kernel* de Linux bajo el nombre *kernel page-table isolation (KPTI)* aislamiento de tablas de páginas del núcleo.

- Mitigación

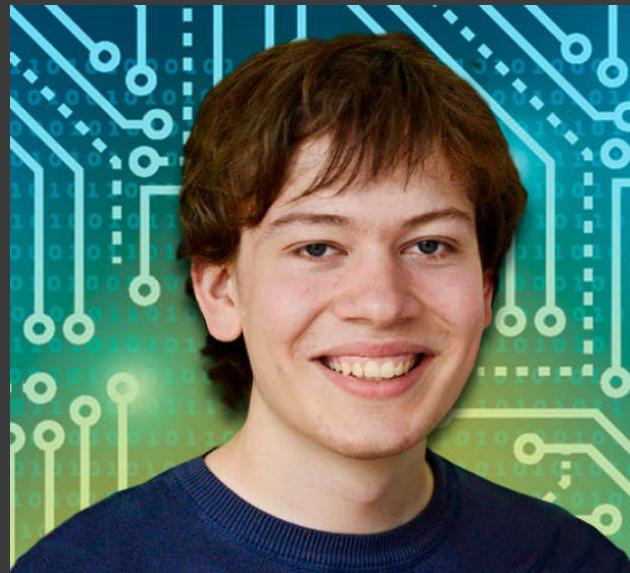
Se ha indicado que la implantación de KAISER puede conllevar una reducción en el rendimiento de la CPU, que alcanzaría según algunos investigadores hasta un 30% dependiendo del uso, aunque Intel consideró este último extremo una exageración.



SPECTRE

- Antecedentes históricos

Spectre fue descubierto de forma independiente por Jann Horn del Proyecto Cero de Google y por Paul Kocher en colaboración con Daniel Genkin, Mike Hamburg, Moritz Lipp y Yuval Yarom.



- ¿Qué es Spectre?

*Spectre* es una vulnerabilidad que afecta a los microprocesadores modernos que utilizan predicción de saltos.

- ¿Qué es Spectre?

Un **predictor de saltos** (*branch predictor*) es un circuito digital utilizado en los microprocesadores para reducir ciclos de parada en la segmentación (*pipeline*).

Los saltos condicionales introducen retardo en estos microprocesadores, ya que normalmente no se evalúa la condición del salto hasta pasadas varias etapas, lo que hace que se puedan introducir instrucciones en la segmentación (*pipeline*) que no deben de ser ejecutadas, teniendo que convertirse posteriormente en NOP, y decrementando así el rendimiento.

La predicción es posible anotando el comportamiento del programa en saltos anteriores.

- ¿Qué software/hardware afecta?

Desde 2018, casi todos los sistemas informáticos están potencialmente afectados por *Spectre*, incluidos los equipos de escritorio, portátiles y dispositivos móviles. Específicamente, se ha demostrado que *Spectre* funciona en procesadores Intel, AMD, los basados en ARM y los de IBM.



- ¿Cómo funciona *Spectre*?

La idea básica es buscar en el código existente lugares donde la especulación tiene "contacto" con datos de otro modo inaccesibles, manipular el proceso para ponerlo en un estado en el que la ejecución especulativa tenga que tocar esos datos.

- Mitigación

### Solución temporal a nivel de *software*

Puesto que es posible la explotación a través de Javascript incrustado en los sitios web, Chrome 64 incluirá por defecto mitigaciones contra el ataque, y los usuarios de Chrome 63 pueden mitigar manualmente el ataque activando la función *Strict Site Isolation*:

(`chrome://flags#enable-site-per-process`).

A partir de Firefox 57.0.4, Mozilla está reduciendo la resolución de los temporizadores JavaScript para ayudar a prevenir ataques sincronizados, y está trabajando y planificando técnicas de ofuscación de tiempos para futuras versiones.

