>> Jorge Dávila

¿Por qué hemos llegado a Spectre y Meltdown?

El final del año 2017 no terminó con doce uvas, sino con dos joyas de lo que podrían denominarse hecatombes tecnológicas. En los albores de 2018 han terminado siendo reales y han sido bautizados los ataques Spectre y Meltdown. Ambos son fallos tan fundamentales que es difícil que los digieran los miles de millones de usuarios de ordenadores de este planeta. Sin embargo, merece la pena hacer el esfuerzo clarificador de bajar a los detalles más íntimos de los procesadores que controlan el mundo, para entender lo naïves que pueden ser las piedras angulares sobre las que todo lo demás está construido.

Si fuera cierto aquel comentario machista de que los hombres sólo saben hacer las cosas una detrás de otra (*subescalares*[1]), y que en cambio las mujeres pueden hacer varias cosas a la vez (*superescalares*[2]), entonces las Unidades Centrales de Procesado (CPU) de todos nuestros ordenadores son claramente femeninas.

Los primeros ordenadores tenían que ser físicamente recableados para realizar tareas diferentes lo que les valió la denominación de «Ordenadores de Programa Fijo». Los primeros que tuvieron un procesador central (CPU) como tal, son los denominados "Ordenadores con Programa Almacenado", cuyos orígenes están en la Máquina Universal de Turing[3] descrita en 1936.

La idea de un ordenador con programa almacenado ya estaba presente en el diseño del primer ordenador civil, el **ENIAC**[4], pero esta característica se omitió inicialmente para que el aparato pudiera estar listo a tiempo. Los primeros ordenadores con programa almacenado fueron la **Máquina Experimental de Pequeña Escala de Mánchester**[5], un pequeño prototipo de ordenador que ejecutó su primer programa el 21 de junio de 1948, y el **Manchester Mark I**[6], que lo hizo la noche del 16 al 17 junio de 1949.

En cuanto a cómo construir un ordenador digital, en su momento hubo dos iniciativas inicialmente distintas. Por una parte estaba la que se conoce como **Arquitectura von Neumann**, que es un diseño que plasmó John von Neumann en el primer borrador del informe sobre la máquina **EDVAC**[7] en 1945, y por otro lado, la denominada **Arquitectura Harward**.

En el informe del EDVAC se describía un ordenador digital compuesto de una Unidad Central de Procesado (CPU[8]) con una Unidad Lógica y Aritmética (ALU[9]), Registros del Procesador[10]; una Unidad de Control (CU[11]),Registros de Instrucciones[12], un Contador de Programa[13], Memoria[14] para almacenar tanto datos como instrucciones y mecanismos de Entrada y Salida (IO[15]).

Con el tiempo, el significado de Arquitectura von Newman ha evolucionado hasta referirse a cualquier ordenador de programa almacenado en el que las instrucciones y los datos no puedan ser accedidos simultáneamente porque comparten un bus común, lo cual supone un cuello de botella que frena las prestaciones del sistema.

Por otro lado está la **Arquitectura Harward**, que tiene pistas y medios de almacenamiento que son distintos y están físicamente separados, para las instrucciones y los datos del programa que se ejecuta. De este modo, el acceso a ambos puede ser simultáneo. Este término proviene de la computadora **Harvard Mark I** basada en relés y cuyas instrucciones del programa estaban almacenadas en cintas de papel perforadas de 24 bits de ancho, y los datos en interruptores electromecánicos.

Ya que la cache es una memoria de respuesta mucho más rápida que la memoria principal, un atacante puede medir el tiempo que tarda en determinado acceso y con ello saber si está viniendo de la memoria RAM o de la cache. Esa información puede ser utilizada para leer los datos contenidos en esa memoria.

Arquitectura Harvard modificada

Hoy en día la mayoría de los procesadores mantienen las líneas de instrucciones y datos separadas para un mejor rendimiento pero, en realidad, lo que implementan es una **arquitectura Harvard modificada**, en las que esa separación se relaja y se pueden cargar programa y datos desde un mismo origen como, por ejemplo, una memoria caché.

La historia de la tecnología de cachés está muy relacionada con el concepto de **Memoria Virtual** que vino a unificar las diferentes memorias que había en los primeros ordenadores (años 60) en un espacio único de direcciones de memoria que era el utilizado por todos los programas. Las tecnologías de almacenamiento en aquellos años podían ir desde memorias de semiconductor, a discos o tambores

[1] Ver https://en.wikipedia.org/wiki/Scalar_processor
[2] Ver https://en.wikipedia.org/wiki/Superscalar_processor
[3] Ver https://en.wikipedia.org/wiki/Universal_Turing_machine
[4] Ver https://en.wikipedia.org/wiki/ENIAC
[5] Ver https://en.wikipedia.org/wiki/Manchester_Small-Scale_Experimental_Machine
[6] Ver https://en.wikipedia.org/wiki/Manchester_Mark_1
[7] Ver https://es.wikipedia.org/wiki/EDVAC
[8] Ver https://en.wikipedia.org/wiki/Central_processing_unit
[9] Ver https://en.wikipedia.org/wiki/Arithmetic_logic_unit
[10] Ver https://en.wikipedia.org/wiki/Processor_register
[11] Ver https://en.wikipedia.org/wiki/Control_unit
[12] Ver https://en.wikipedia.org/wiki/Instruction_register
[13] Ver https://en.wikipedia.org/wiki/Program_counter
[14] Ver https://en.wikipedia.org/wiki/Computer_memory
[15] Ver https://en.wikipedia.org/wiki/Input/output

magnéticos pasando por tapetes de núcleos de ferrita [16]. La memoria virtual resolvió el problema de las distintas naturalezas de la memoria, pero todavía quedaba resolver el problema de la disponibilidad y los tiempos de respuesta.

La Memoria Virtual permite al sistema operativo ejecutar muchos programas a la vez, de modo que cada programa tenga su propio espacio de direcciones virtuales que están vinculadas (*mapped*) a páginas de la memoria física.

Los primeros ordenadores eran tan lentos (kHz) que las memorias *cache* realmente no eran necesarias. Estrictamente hablando. La primera *cache* de datos se utilizó en el IBM System/360 Modelo 85 de 1968,y reducía el tiempo de acceso a datos e instrucciones a una cuarta parte.

Cache: características

La *cache* es la forma más cara de memoria en un ordenador. Generalmente está construida con memoria RAM estática [17] que es mucho más complicada (necesita más electrónica) que la dinámica [18] que se utiliza en la memoria central (RAM). Otra diferencia está en su ubicación; con procesadores corriendo a varios GHz (10^9 Hz) la *cache* tiene que estar en la misma pieza de silicio en la que está la CPU. Enviar y recibir datos a través de los buses de datos e instrucciones supone un tiempo intolerable para la dinámica de las CPUs modernas.

Desde la década de los 80, la diferencia de velocidad entre lo que pasa dentro de la CPU y la memoria principal (RAM externa) ha estado creciendo. Parte del problema son las dimensiones físicas de la instalación. Las *caches* que están dentro de un procesador se encuentran, como mucho, a algún milímetro del procesador al que sirven. Sin embargo, la memoria principal se encuentra a centímetros de distancia.

Meltdown explota distintos efectos colaterales de la ejecución especulativa en los procesadores modernos para leer posiciones arbitrarias en la memoria del kernel donde se pueden encontrar de todo, desde datos personales a contraseñas y claves criptográficas.

Teniendo en cuenta que la velocidad de la luz es un límite no superado para la transferencia de información, la luz recorre un milímetro en $3{,}3 \times 10^{-12}$segundos, lo que permitiría frecuencias de hasta 30,3GHz (10^9 Hz). Sin embargo, la luz recorre 20 centímetros en $6{,}7 \times 10^{-10}$ segundos, y eso sólo permite frecuencias de 1,5 GHz.

El procesador no es consciente de la existencia de las distintas *caches* y, de hecho, su mantenimiento lo realiza una unidad independiente denominada **Memory Management Unit** (**MMU**). Estas unidades son sistemas lógicos complejos que necesitan correr al menos tan deprisa como la propia CPU para poder así alimentarla eficazmente con datos e instrucciones.

Dado que los programas de ordenador suelen utilizar numerosas veces los mismas datos e instrucciones, la presencia cerca del procesador de una memoria *cache* con suficiente capacidad y velocidad, permite que en esos reiterados accesos los datos no tengan que venir de la lejana memoria principal, sino de la inmediatez de los distintos niveles de *cache* con los que cuentan los procesadores modernos.

Avance de la arquitectura

Por otra parte, la arquitectura de la CPUs ha ido avanzando prodigiosamente no sólo por la miniaturización y complejidad crecientes, sino además por la mejora en los paradigmas de operación. Al principio las cosas se hacían paso a paso, y solo se ejecutaba una instrucción por lo que el procesador se quedaba sin hacer nada mientras esperaba que le llegaran los datos o instrucciones que necesitaba (*subescalar*), con lo que se quedaba muy lejos de ejecutar una operación en cada ciclo de reloj.

Para aumentar la velocidad la única posibilidad era introducir en el diseño de las CPUs comportamientos menos secuenciales y más paralelos. Cuando se habla de paralelismo en este entorno, se refiere a dos técnicas de diseño: el **Paralelismo a nivel de Instrucciones** (**ILP** [19]) que busca aumentar el número de instrucciones que se ejecutan simultáneamente en la CPU, es decir, aumentar la utilización de toda la capacidad ejecutora que haya en la pieza de silicio; y el **Paralelismo a nivel de Tareas** (**TLP** [20]), que pretende aumentar el número de hilos (*threads* [21]) de procesado que una CPU puede ejecutar simultáneamente.

En esta última dirección está el **Simultaneous Multi-Threading** (**SMT** [22]) que permite a los programas preparados para ello, ejecutar múltiples hilos y procesarlos en paralelo dentro de un único procesador. El truco consiste en simular dos procesadores lógicos dentro de un único procesador físico para así aprovechar mejor sus unidades de cálculo manteniéndolas ocupadas durante un mayor porcentaje de tiempo. Este proceder conlleva una mejora en la velocidad de las aplicaciones de aproximadamente un 60%.

Con este ataque se puede leer la memoria de otro proceso o máquina virtual ejecutándose en la nube sin tener ningún permiso o privilegio para ello. Este ataque afecta a millones de clientes y virtualmente a cualquier usuario de un ordenador personal.

Saltarse la protección

Tanto la **MMU** como el hardware encargado de la **Memoria Virtual** trabajan con el sistema operativo para proveer **la protección del espacio de memoria**; es decir, evitar que otros programas lean o escriban en el espacio de memoria que no les corresponden, y **no dejar que nadie interactúe con el núcleo (*kernel*) del sistema operativo**. Esta protección es la que se han saltado lateralmente los ataques **Spectre** y **Meltdown**.

Entender lo que ocurre, incluso en los sistemas de

[16] Ver https://en.wikipedia.org/wiki/Magnetic-core_memory
[17] Ver https://en.wikipedia.org/wiki/Static_random-access_memory
[18] Ver https://en.wikipedia.org/wiki/Dynamic_random-access_memory
[19] Ver https://en.wikipedia.org/wiki/Instruction-level_parallelism
[20] Ver https://en.wikipedia.org/wiki/Task_parallelism
[21] Ver https://en.wikipedia.org/wiki/Thread_(computing)
[22] Ver https://en.wikipedia.org/wiki/Simultaneous_multithreading

cache más sencillos, es algo realmente difícil. En un procesador moderno hay múltiples niveles de *cache*, algunos son independientes y otros se comparten entre diferentes núcleos de la misma CPU. Además también hay componentes dedicados a la **ejecución especulativa** (**SE**[23]), que traen de la memoria a la memoria *cache*, datos que pueden ser utilizados antes que el núcleo procesador esté listo para utilizarlos. Esa sección es la clave del ataque **Meltdown**.

La ejecución especulativa es una técnica de optimización en la que un sistema desarrolla una actividad que puede llegar a no ser necesaria. El trabajo se adelanta antes de que se sepa si va a ser necesario o no. En el caso de que, llegada la hora, ese procesado sea útil, el sistema se evita el retraso de tener que hacerlo en ese momento. Si llegado ese momento, lo hecho no resulta útil, se descarta el resultado de ese esfuerzo y se deshacen los cambios que hayan podido haberse realizado. Este fallo no penaliza el tiempo de ejecución ya que el trabajo adelantado se hizo en paralelo con otros cálculos.

Tanto **Spectre** como **Meltdown** utilizan la memoria *cache* en un ataque lateral de tiempo[24]. Ya que la *cache* es una memoria de respuesta mucho más rápida que la memoria principal, un atacante puede medir el tiempo que tarda en determinado acceso y con ello saber si está viniendo de la memoria RAM o de la *cache*. **Esa información puede ser utilizada para leer los datos contenidos en esa memoria**.

Explotación de efectos colaterales

La seguridad de los ordenadores depende esencialmente del aislamiento de los distintos espacios de memoria, en particular, en que los rangos de direcciones asignados al núcleo del sistema operativo no sean accesibles desde el espacio de usuario. **Meltdown** explota distintos efectos colaterales de la ejecución especulativa en los procesadores modernos para leer posiciones arbitrarias en la memoria del kernel donde se pueden encontrar de todo, desde datos personales a contraseñas y claves criptográficas. Este ataque funciona en diferentes arquitecturas de Intel desde al menos 2010 y potencialmente, procesadores de otras marcas también están afectados.

La causa esencial de Meltdown es el hardware. El ataque es independiente del sistema operativo, y no se basa en ninguna debilidad software. Meltdown rompe cualquier medida de seguridad que esté basada en el supuesto aislamiento en el espacio de direcciones de memoria. Con este ataque se puede leer la memoria de otro proceso o máquina virtual ejecutándose en la nube sin tener ningún permiso o privilegio para ello. Este ataque afecta a millones de clientes y virtualmente a cualquier usuario de un ordenador personal.

El único mecanismo de defensa que permite ser inmune a Meltdown es el denominado **KAISER**[25]. Es una característica del núcleo de Linux que hace más robusto el aislamiento de memoria aleatorizando el espacio de direcciones de memoria (**KASLR**[26]).

La **predicción de bifurcaciones** (*branchprediction*[27]) y la ejecución especulativa se usan en los procesadores modernos para maximizar su velocidad de cómputo. Por ejemplo, si el destino de una bifurcación depende del valor leído en la memoria, mientras el valor no llega, la CPU trata de adivinar ese resultado y continúa la ejecución en una de las bifurcaciones posibles. Cuando el valor de la memoria llega a la CPU, ésta ya puede aceptar o descartar lo que ha calculado mientras ese valor llegaba.

Actualmente, la lógica hardware de la ejecución especulativa no se mantiene dentro del espacio del usuario que lo ejecuta, por lo que puede acceder a la memoria y los registros de otro, de su víctima, y realizar así operaciones con ellos y en ellos.

Los ataques Spectre consisten en conseguir de la víctima que haga operaciones especulativas y, a través de ellas, el atacante extraer información confidencial de lo que está haciendo. Esta posibilidad invalida todas las asunciones de seguridad basadas en la separación de procesos como son la compartimentalización, la compilación just-in-time, y otras contramedidas que se han propuesto contra los ataques a través de canales laterales.

Ya en 1995, varios autores llamaron la atención sobre los efectos negativos para la seguridad que tenían las arquitecturas de las familias de procesadores 80x86. Curiosamente, este análisis se hizo bajo el auspicio del programa Trusted Products Evaluation Program de la NSA. Está claro que durante más de veinte años los "arquitectos de procesadores" de Intel, no han tenido a bien atender aquella advertencia.

Spectre: operaciones especulativas

Los ataques **Spectre** consisten en conseguir de la víctima que haga operaciones especulativas y, a través de ellas, el atacante extrae información confidencial de lo que está haciendo. Las técnicas a utilizar son las habituales en los ataques por canales laterales, como son las de inducir fallos (*faultattacks*[28]), o la denominada "programación orientada por el retorno" (ROP[29]).

Esta posibilidad invalida todas las asunciones de seguridad basadas en la separación de procesos, como son la compartimentalización (*containerization*), la compilación *just-in-time* (JIT[30]), y otras contramedidas que se han propuesto contra los ataques a través de canales laterales.

Los ataques Spectre y Meltdown tienen éxito en microprocesadores de Intel,

[23] Ver https://en.wikipedia.org/wiki/Speculative_execution
[24] Ver https://en.wikipedia.org/wiki/Timing_attack
[25] Ver https://en.wikipedia.org/wiki/Kernel_Page_Table_Isolation
[26] Ver https://en.wikipedia.org/wiki/Address_space_layout_randomization#KASLR
[27] Ver https://en.wikipedia.org/wiki/Branch_predictor
[28] Ver https://en.wikipedia.org/wiki/Fault_injection
[29] Ver https://en.wikipedia.org/wiki/Return-oriented_programming
[30] Ver https://en.wikipedia.org/wiki/Just-in-time_compilation

AMD y ARM, que son utilizados en miles de millones de dispositivos. Aunque en algunos casos sea posible utilizar contramedidas específicas e improvisadas para algún procesador, las soluciones correctas requieren acertados cambios en los diseños de los procesadores, así como nuevas arquitecturas de los conjuntos de instrucciones (ISA[31]). Y todo ello requiere tiempo y mucho esfuerzo.

El método más sencillo y efectivo para mitigar los efectos de Spectre y Meltdown es **deshabilitar las *caches***. Esto hace que el ordenador se vuelva increíblemente lento si lo comparamos con lo que estamos acostumbrados. Los parches que se están desarrollando no son tan extremistas, pero cambian el modo en el que la CPU trabaja con la memoria *cache,* especialmente en lo que se refiere a los **cambios de contexto**[32] entre el espacio de usuario y el espacio del *kernel*.

Tarea increíblemente compleja

Los movimientos de datos e instrucciones en una CPU multinúcleo moderna y el diseño de política de *cache* que lleva asociado **es una tarea increíblemente compleja**. Los fabricantes de procesadores llevan años y cuentan con toneladas de experiencia sobre cómo los procesadores mueven los datos y cómo pueden hacerlo de forma más rápida y fiable.

Cambios en el micro código de un procesador es algo que se hace sólo cuando es absolutamente necesario y después de muchas horas de pruebas. Un cambio a salto de mata, como ocurre con los prometidos parches de Intel y AMD, y que todavía tienen que venir, seguro que traerán problemas asociados que pueden ser graves. Algunos de ellos ya se han manifestado en la forma de importantes pérdidas de eficiencia, "pantallazos azules" y "cuelgues" de sistemas operativos y de algunas aplicaciones.

Nada realmente nuevo

Lo más sorprendente es saber que todo esto no es realmente nuevo y que ya en 1995, varios autores[33] llamaron la atención sobre los efectos negativos para la seguridad que tenían las arquitecturas de las familias de procesadores 80x86.Curiosamente, este análisis se hizo bajo el auspicio del programa **Trusted Products Evaluation Program** (**TPEP**) de la **NSA**. Está claro que durante más de veinte años los "arquitectos de procesadores" de Intel, no han tenido a bien atender aquella advertencia.

Desde el punto de vista social, este incidente ha puesto de manifiesto una de las características básicas de nuestro mercado capitalista: el acceso a la información privilegiada y la no competencia lícita. En cuanto el CEO de Intel, **Brian Krzanich**, se enteró del incidente que nos ocupa, vendió casi un millón de acciones[34] de la compañía que dirige para evitarse posibles pérdidas millonarias en el corto y medio plazo y, según parece, no hay nada ilegal en ello.

Las grandes corporaciones nos tienen acostumbrados a que, cuando pasa algo serio, se celebra algún baile peristáltico[35] en el que lo único que se hace es cambiar a sus directivos de sitio. En la mayoría de los casos, esas reorganizaciones terminan siendo sólo una sesión de trile[36] donde todos los ejecutivos terminan cobrando más, los clientes pagando las consecuencias y sin que nada sustancial cambie.

La compañía protagonista principal de este desaguisado, Intel, y como respuesta al mismo, ha movido a algunos de sus directivos[37] para formar un grupo (supuestamente) dedicado a la seguridad de sus productos[38]; algunos quieren ver en ello un cambio de verdad pero sólo el tiempo lo dirá y yo, por el momento, lo dudo.

El mensaje inicial de Intel fue claramente desafiante y minimizaba el posible impacto de los ataques, a la vez que acusaba a muchos de "sobreactuar", de dramatizar. Desde entonces ha ido aportando pocade información muy técnica (nada útil para el gran público) y prefiere optar por hacer desconocidas actualizaciones directas del microcódigo que controla sus procesadores. Esas "soluciones rápidas" deberían ser sólo para aquellos usuarios que sean más lanzados y no teman vivir en el filo de la navaja. Reacciones tan poco maduras no deberían salir al mercado planetario ya que pueden llegar a ser peores que lo que las motivó.

Todo esfuerzo técnico en estos momentos es bienvenido pero hay que tomarlo siempre con mil precauciones, ya que el primer mundo informatizado necesita encontrar la manera de evitar, no sólo los actuales, sino los futuros desastres del tipo **Meltdown** y **Spectre**. Es necesario cambiar la forma en la que se hacen las cosas y que las compañías que nos inundan de artefactos y soluciones hagan las cosas bien, y no se tire para adelante con sistemas no suficientemente seguros, como se hace siempre.

En lugar de monetizar su fracaso, el CEO de Intel y muchos más, debería rendir cuentas, legales y económicas, ante los millones de usuarios cuyas máquinas se han creído el mito de que había intimidad (compartimentalización) dentro de los procesadores y sistemas virtuales que construyen nuestro presente y acunan nuestro futuro. ■

En lugar de monetizar su fracaso, el CEO de Intel y muchos más, debería rendir cuentas, legales y económicas, ante los millones de usuarios cuyas máquinas se han creído el mito de que había intimidad (compartimentalización) dentro de los procesadores y sistemas virtuales que construyen nuestro presente y acunan nuestro futuro.

[31] Ver https://en.wikipedia.org/wiki/Instruction_set_architecture
[32] Ver https://en.wikipedia.org/wiki/Context_switch
[33] Sibert, O.; Porras, P.A.; Lindell, R.: "The Intel 80x86 Processor Architecture: Pitfalls for Secure Systems". Proceedings IEEE Xplore Conference: Conference: Security and Privacy, pp. 211-222. 1995.
[34] Ver https://www.cnbc.com/2018/01/04/intel-ceo-reportedly-sold-shares-after-the-company-already-knew-about-massive-security-flaws.html
[35] Ver https://en.wikipedia.org/wiki/Peristalsis
[36] Ver https://es.wikipedia.org/wiki/Trile
[37] Ver http://www.oregonlive.com/silicon-forest/index.ssf/2018/01/intel_reorganizes_amid_fervor.html
[38] Ver https://www.anandtech.com/show/12264/intel-forms-product-assurance-and-security-group-amid-meltdown-and-spectre-fallout

Jorge Dávila
Consultor independiente
Director
Laboratorio de Criptografía
LSIIS – Facultad de Informática – UPM
jdavila@fi.upm.es