Intel Spectre

Enrique José Rodríguez Martín
Uo257565@uniovi.es

Resumen
Este trabajo indaga en Spectre, posiblemente una de las vulnerabilidades más importantes en los últimos años y que ha afectado considerablemente no solo a los productores de hardware, si no a la confidencialidad de los datos de los usuarios y que ha conllevado a uno de los escándalos más recordados en el sector. Las consecuencias, origen, causas de la vulnerabilidad y su mitigación serán descritas en las páginas siguientes.

1. Introducción

Hoy en día la industria del "*hardware"* compite ferozmente por ofrecer a sus clientes el mayor rendimiento posible, Intel es la mayor compañía en el sector de creación de CPUs, seguido muy de lejos por su rival AMD. Datos extraídos de la web muestran un análisis del 5 de mayo de 2019 que sitúan a Intel con una cuota de mercado superior al 77%[1]; actualmente se usan numerosas técnicas para mejorar el rendimiento de nuestros procesadores, una de esas técnicas es la llamada ejecución especulativa, dicha técnica se basa en el uso de algoritmos que tratan de predecir que serie de instrucciones se van ejecutar por parte de un programa, para así cargar en memoria dichas instrucciones y conseguir un mayor rendimiento; esto se debe a que cuando se requiera dicha instrucción el sistema, si ha "adivinado" correctamente, albergará en su memoria caché la instrucción que se requerirá en próximas iteraciones ahorrando así valiosos ciclos de reloj. Esta técnica utilizada durante años por Intel y sus competidores ha resultado ser una total catástrofe a nivel de seguridad, dicha ejecución especulativa podía ser forzada de forma programática a fallar, resultando en la revelación del contenido de la memoria o los registros de la CPU. Intel intentó por varios medios ocultar temporalmente dicha falla en sus procesadores, generando aún mayor controversia cuando su CEO liquidó la mayoría de sus acciones[2] antes de la caída bursátil de la compañía[3]. Aunque se contempló durante años la posibilidad de que la ejecución especulativa podría acarrear severos problemas de seguridad, hasta principios de 2018 no se había descubierto, o al menos no se había publicado, una forma de forzar a dichos sistemas de predicción a revelar contenido de la memoria del dispositivo afectado. La noticia pese a que se intentó detener por un tiempo supuso un duro revés para estos creadores de "*hardware"* que tuvieron que implementar a la carrera soluciones para mitigar esta vulnerabilidad a la carrera. También cabe señalar que esta vulnerabilidad afectó a las principales compañías del sector de las CPUs, ya que la técnica de la ejecución especulativa no era una tecnología exclusiva y estaba ya extendida por todas ellas.

2. ¿Por qué Spectre?

Spectre es una vulnerabilidad que se relaciona íntimamente con el "*hardware"*, no con el "*software"* como normalmente ocurre con las vulnerabilidades más habituales. El nombre *"Spectre"* viene a decir en nuestro idioma espectro o fantasma, esto se debe a que como se basa en atacar la ejecución especulativa de la CPU, las medidas para mitigar esta falla de seguridad no serían sencillas y por actuar de forma invisible para los programas [4].

¿Qué es la ejecución especulativa?

En palabras de la propia Intel, debemos entender a la EE como un "scout" o explorador que avanza por delante del resto de tareas y procesos de la CPU, este va explorando según una serie de criterios, qué camino, o siendo más estrictos, qué posibles tareas ejecutará la CPU próximamente, antes siquiera que la CPU las solicite realmente, mejorando así el rendimiento del sistema. [5]

¿Cómo funciona un ataque frente a la vulnerabilidad de Spectre?

A diferencia de *"Meltdown"* donde se centra en una falla de seguridad concreta, Spectre es mucho más difícil de explotar ya que es dependiente de la empresa productora del dispositivo y del propio modelo de la CPU, pero hay que tener en cuenta que afecta a todos los CPUs modernos, de los últimos veinte años, por su diseño y como interactúa con los sistemas operativos actuales.

Los ataques de Spectre se pueden categorizar en dos tipos de ataques principalmente:

Variante 1 *"Bounds Check Bypass"*:

Este ataque no se centra en una escalada de privilegios que permita al atacante conocer el contenido de la memoria, más bien, se centra en "adiestrar" al *kernel* y engañarlo para que ejecute código que le lleve a predecir de forma incorrecta, el atacante debe contaminar el mecanismo de predicción y hacer que este falle en su ejecución especulativa para así obtener los datos que posteriormente deberá filtrar para obtener la información relevante.

A continuación, veremos como funciona esto con el ejemplo provisto por los investigadores de esta vulnerabilidad [6] [7]:

Supongamos que tenemos el siguiente código para ejecutar:

```
1  if ( x < array1_size) {
2      y = array2[array1[x] * 256] ;
3  }
```

Asumimos pues las siguientes premisas por orden de eventos:

1. El atacante controla la variable x.
2. Array1_size no está cacheada.
3. Array1 si está cacheada.
4. La CPU predice que x es menor que array1_size. Las CPUs emplean distintos algoritmos de predicción dependiendo del fabricante, ahí reside la dificultar de aprovechar esta vulnerabilidad ya que no solo depende del fabricante si no también del modelo.
5. La CPU ejecuta el cuerpo de la sentencia condicional mientras está esperando que array1_size sea cargado.

6. El atacante puede determinar el valor actual de array1[x] usando varías técnicas que muestran el contenido de la cache al atacante.

De esta forma con los conocimientos necesarios, un atacante puede hacerse con el contenido de la memoria de una máquina atacada.

Mitigación variante 1:
Para solucionar la variante 1 se utiliza la técnica de *"Static analysis and fencing"*, esta técnica analiza secuencias de código que puedan estar ligadas al control de un atacante para así este interferir con la ejecución especulativa. Fragmentos de código vulnerable pueden utilizar una *"instrucción serializable"* como *"lfence"* la cual detiene la ejecución especulativa hasta que las instrucciones hasta esa *"valla"* se hayan ejecutado. Se debe tener especial cuidado con dichas instrucciones, ya que su abuso o uso incorrecto pueden provocar serios problemas de rendimiento.

Variante 2 *"Branch Target Injection"* [8]
Esta variante se centra en los sistemas de predicción de saltos usados por los procesadores para anticipar que operaciones se ejecutarán especulativamente después de una predicción de salto. Controlando como estos predictores funcionan *("entrenamiento")*, un atacante fuerza a que se ejecuten especulativamente ciertas instrucciones y con ese código malintencionado el intruso pueda inferir ciertos datos que albergan las caches del procesador.

Este ataque se basa en la vulnerabilidad del sistema de predicción y por lo tanto solo puede afectar a ciertas instrucciones propias del juego de instrucciones del procesador, nuevamente hace que este ataque sea dependiente del procesador y del modelo, Intel por ejemplo nos informa que instrucciones de salto son las que pueden ser susceptibles de ser manipuladas [8] :

Branch Type	Instruction	Opcode
Near Call Indirect	`CALL r/m16, CALL r/m32, CALL r/m64`	FF /2
Near Jump Indirect	`JMP r/m16, JMP r/m32, JMP r/m64`	FF /4
Near Return	`RET, RET Imm16`	C3, C2 Iw

Mitigación variante 2:

Se han desarrollado dos variantes para solventar estos problemas:

1. Mecanismos de control del predictor de saltos: se introduce una interfaz entre el *"hardware"* y el *"software"* del sistema que permite a este ultimo evitar que un atacante controle el predictor.

2. El uso de *"retpoline"* desarrollado por Google posiblemente en colaboración con Intel; la idea que subyace en este sistema es prevenir que las instrucciones de salto sean usadas mediante ejecución especulativa como herramienta para obtener información, es decir, no protege directamente al kernel de filtrar información a consecuencia de un ataque.

La realización de ataques basados en la vulnerabilidad de Spectre se centraron principalmente en atacar a los buscadores web utilizando JavaScript ejecutado en el cliente, también afectó a los servicios en la nube; aunque bien es cierto que se podría utilizar de forma local con una pequeña aplicación y que se llegaron a implementar ataques que no necesitaban ejecutar ningún código en el sistema atacado, los daños a particulares fueron más o menos limitados y de severidad moderada. [8] [9]

3. Conclusión

Como conclusión podemos decir con certeza que la vulnerabilidad de Spectre en conjunción con su homóloga Meltdown han supuesto un duro revés a los desarrolladores de *"hardware"*, que han tenido que proteger a los procesadores de los últimos veinte años para subsanar estos problemas de seguridad, es poco común que una investigación de esta índole suponga un cambio radical en el desarrollo de nuevos sistemas, pero como hemos visto y ante la severidad de las vulnerabilidades, esta revelación afectará al desarrollo de nuevos procesadores, los cuales tendrán que estar protegidos ante estas fallas. En los primeros momentos tras la publicación de los parches ante las vulnerabilidades estos supusieron numerosos problemas de rendimiento en las CPUs afectadas, y de cara al futuro afectará substancialmente a los usuarios en los próximos años. Las consecuencias económicas para las empresas de hardware fueron considerables, sobre todo para Intel la cual fue la más afectada por dichas vulnerabilidades.

4. Referencias

1. https://www.cpubenchmark.net/market_share.html
2. https://arstechnica.com/information-technology/2018/01/intel-ceos-sale-of-stock-just-before-security- bug-reveal-raises-questions/
3. https://www.cnbc.com/2018/01/04/intel-stock-down-intc-could-meltdown-spectre-exploit-benefit-amd.html
4. https://mashable.com/2018/01/05/meltdown-spectre-names-cpu-bug/?europe=true
5. https://www.youtube.com/watch?v=pi2ftnlfImo
6. https://spectreattack.com/spectre.pdf
7. https://medium.com/@mattklein123/meltdown-spectre-explained-6bc8634cc0c2

8. https://software.intel.com/security-software-guidance/software-guidance/branch-target-injection
9. https://arstechnica.com/gadgets/2018/07/new-spectre-attack-enables-secrets-to-be-leaked-over-a-network/