L'Hardware e la Sicurezza IT - Parte 3: Meltdown e Spectre

Author : Andrea Pasquinucci

Date : 21 novembre 2018

Nei due articoli precedenti, L'Hardware e la Sicurezza IT – Parte 1: un poco di storia e L'Hardware e la Sicurezza IT – Parte 2: Attacchi Side Channel, Row Hammer ed attacchi alla Cache, sono stati riassunti brevemente i principali eventi ed alcune informazioni utili per poter svolgere una breve analisi delle vulnerabilità di sicurezza informatica dovute all'Hardware e chiamate "Meltdown" e "Spectre" [1], divulgate al pubblico il 3 gennaio 2018.

Ma prima di addentrarci in alcuni dettagli di queste vulnerabilità è necessario chiarire un poco il motivo della loro rilevanza e della loro fama. A prima vista infatti Row Hammer, descritto nell'articolo precedente, risulta essere una vulnerabilità di gravità maggiore in quanto permette di modificare dei dati (anche privilegiati) ed in alcuni casi di prendere il controllo totale del sistema. Invece Meltdown e Spectre permettono di accedere in sola lettura ad informazioni riservate presenti sul sistema. Row Hammer è però mitigato da alcuni fattori:

- si applica solamente alla memoria RAM e non tutte le memorie RAM sono vulnerabili (in linea di principio sarebbe possibile sostituire RAM vulnerabili con RAM non vulnerabili nei sistemi esistenti);
- gli attacchi Row Hammer sono molto evidenti in quanto vengono eseguite delle azioni non usuali;
- sono possibili e disponibili varie contromisure sia software che hardware che possono mitigare o impedire gli attacchi.

Meltdown e Spectre sono invece vulnerabilità hardware del disegno delle CPU. Meltdown si applica praticamente a tutti i processori Intel disegnati e prodotti dal 1995 (ad eccezione dei processori Intel Itanium e Intel Atom prima del 2013) ed in parte anche ai processori ARM. Spectre si applica a tutti i recenti processori.

È da notare anche che sia di Meltdown che di Spectre esistono molte varianti che differiscono tipicamente per dettagli tecnici di esecuzione ma che ovviamente possono rendere del tutto inefficace una particolare mitigazione. Infatti sono state prodotte delle patch software sia per i sistemi operativi sia con microcodice di aggiornamento delle CPU, patch che mitigano alcune, ma non tutte, delle molte varianti di Meltdown e Spectre. Tra le varianti vi sono anche:

- Netspectre [3] che implementa un attacco Spectre via rete in Javascript;
- Foreshadow [4] che implementa un attacco al Software Guard eXtension (SGX), estensione delle CPU Intel per gestire dati riservati (Trusted Execution Environment), e a qualunque informazione presente nella Cache L1, anche del Kernel del Sistema Operativo, di un Hypervisor o del System Management Mode (SMM) della CPU stessa.

Come vedremo, Meltdown e Spectre si basano sul disegno di funzionalità di base delle CPU moderne. Per questo soluzioni definitive richiedono il ridisegno, la produzione e la sostituzione di tutte le CPU esistenti, nei server, nelle postazioni di lavoro, negli smartphone e anche di quelle integrate nei molti piccoli dispositivi IoT presenti nella nostra vita quotidiana. Se da una parte la gravità di queste vulnerabilità non è potenzialmente grave quanto quella di Row Hammer, l'origine e la diffusione fanno di Meltdown e Spectre due vulnerabilità molto più significative e la cui risoluzione richiederà ancora parecchi anni.

Meltdown

Delle due vulnerabilità, Meltdown è la più semplice da implementare ed al contempo quella per cui, come vedremo, esistono alcune contromisure software.

La caratteristica principale di Meltdown è di violare uno dei principali assunti di sicurezza del disegno delle CPU moderne. Infatti, come abbiamo descritto nel primo articolo, uno degli assunti del disegno delle CPU moderne è la presenza di anelli di privilegi, implementati in hardware, tali per cui i processi non possono accedere a dati ed istruzioni ad un livello di privilegio maggiore rispetto a quello a cui sono eseguiti. In particolare, accede ed è eseguito a livello zero (il più alto) solo il kernel del Sistema Operativo, che controlla e gestisce tutto l'hardware, i dati e l'esecuzione di tutti i processi. La vulnerabilità Meltdown permette di accedere in lettura a tutti i dati gestiti dal kernel del Sistema Operativo da parte di un processo con privilegi bassi, evitando i controlli hardware presenti nella CPU.

L'osservazione principale su cui si basa Meltdown è che le micro-istruzioni non sono sempre eseguite dalle CPU nell'ordine previsto dal programma. Come abbiamo descritto nel primo articolo, caratteristiche comuni delle CPU moderne, sin dagli anni '60 e '70, sono le pipeline di esecuzione e l'esecuzione di istruzioni non in ordine (“out-of-order”). Nel corso degli anni, queste caratteristiche sono state sviluppate ampiamente dai progettisti e le CPU attuali sfruttano ampiamente sia l'esecuzione parallela di istruzioni di tipo diverso, sia l'esecuzione out-of-order che l'esecuzione speculativa (si veda Fig. 1). Il caso più semplice di esecuzione out-of-order è l'esecuzione speculativa in presenza di un branch nel programma: ad esempio l'esito di un test su di un valore di una variabile decide se eseguire alcune istruzioni o delle altre (il tipico IF-THEN-ELSE). In attesa che vengano calcolati i valori che permettono di valutare la condizione (IF), se sono già presenti i valori per calcolare il THEN (o l'ELSE), allora la CPU procede all'esecuzione delle istruzioni nel branch. Quando poi sarà valutata la condizione (IF) la CPU decide se mantenere il branch già calcolato nel caso di scelta corretta, oppure abbandonarlo e calcolare un altro branch. La presenza di un branch non è però l'unica situazione in cui una CPU esegue istruzioni out-of-order, ve ne sono molte altre individuate dai progettisti delle CPU e che permettono delle ottimizzazioni, alle volte anche notevoli, delle prestazioni.

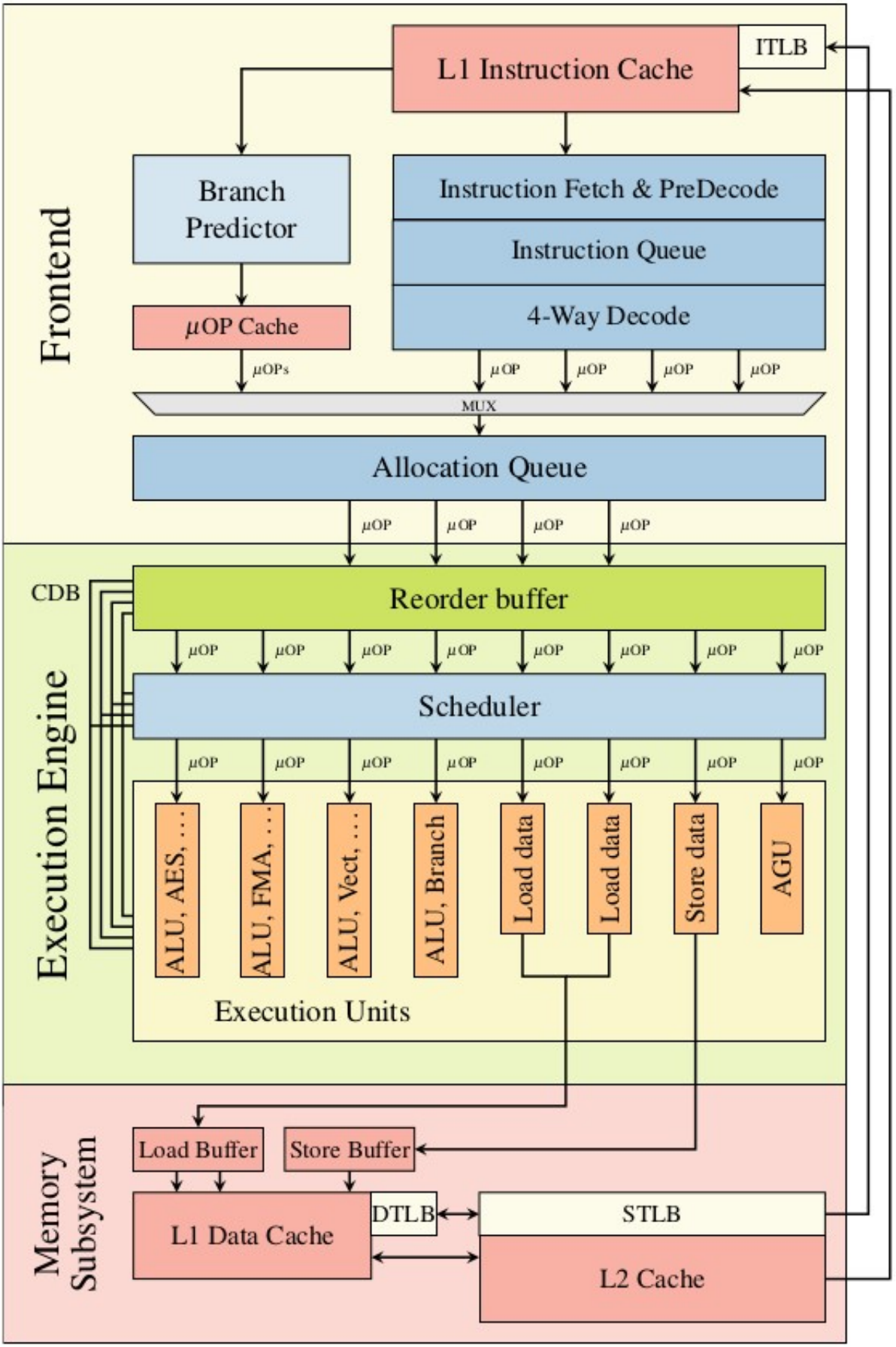
Frontend
L1 Instruction Cache
ITLB
Branch Predictor
Instruction Fetch & PreDecode
Instruction Queue
4-Way Decode
µOP Cache
µOPs
µOP
MUX
Allocation Queue
Execution Engine
CDB
Reorder buffer
Scheduler
ALU, AES, ...
ALU, FMA, ...
ALU, Vect, ...
ALU, Branch
Load data
Load data
Store data
AGU
Execution Units
Memory Subsystem
Load Buffer
Store Buffer
L1 Data Cache
DTLB
STLB
L2 Cache

Fig 1. Descrizione schematica della micro-architettura di una CPU Intel Skylake con le principali componenti per l'esecuzione out-of-order delle micro-istruzioni [2]

L'attacco Meltdown si può riassumere come segue:

1. l'attaccante è un processo utente senza alcun privilegio particolare, per cui non dovrebbe avere accesso diretto in lettura alla memoria del kernel;
2. per prima cosa viene individuata una linea di esecuzione che garantisce l'esecuzione out-of-order delle istruzioni successive;
3. tra le istruzioni da eseguire out-of-order ed in anticipo rispetto al flusso delle altre operazioni, viene inserita, come descritto nei passi seguenti, la lettura di una locazione di memoria del kernel, in linea di principio operazione vietata dall'hardware;
4. la CPU carica in un proprio registro il contenuto della locazione di memoria del kernel per prepararsi all'istruzione di lettura;
5. la lettura del registro viene eseguita nel branch out-of-order e sulla base del valore del registro viene scritto un dato nella cache;
6. in parallelo la CPU verifica se il processo in esecuzione nella branch out-of-order ha privilegi sufficienti per accedere ai dati;
7. quando la verifica dei privilegi è completata senza successo, la CPU esegue una eccezione che blocca l'esecuzione del programma.

E' chiaro che tra l'esecuzione del punto 5 e quella del punto 7 c'è una “race condition” (corsa critica), ma è possibile scrivere un flusso di istruzioni tale che 5 venga tipicamente eseguito prima di 7.

L'attacco non è però finito, perché l'informazione è presente solo nella cache della CPU, non è memorizzata dal processo attaccante e non è direttamente leggibile in quanto privilegiata. Per estrarre l'informazione dalla cache prima che la CPU la sovrascriva o la cancelli, si possono sfruttare gli attacchi alla cache descritti nel precedente articolo. In particolare la prima versione di Meltdown utilizza Flush+Reload come Covert Channel per inferire i dati presenti in cache.

Ripetendo la procedura di attacco per tutte le locazioni di memoria del kernel, il processo attaccante può leggere tutti i dati nella memoria del kernel della macchina, violando l'isolamento che dovrebbe essere garantito dalla struttura hardware in anelli del kernel.[1]

Nell'esecuzione di Meltdown c'è un passaggio cruciale: l'attaccante deve già conoscere gli indirizzi in memoria (virtuale) dei dati del kernel. In realtà, nei principali Sistemi Operativi attuali lo spazio di memoria del kernel (kernel space) è mappato anche nello spazio di memoria virtuale di ogni processo (user space). Questo rende molto più veloce e semplice la chiamata al kernel (via System Call) per eseguire azioni anche comuni ma privilegiate e controllate dal Sistema Operativo, quali ad esempio la lettura e scrittura di dati su disco e qualunque altra operazione che interagisca con l'hardware. L'assunto era che non vi fossero rischi di sicurezza, in quanto l'hardware avrebbe protetto l'accesso allo spazio di memoria virtuale riservato al kernel.

Visto che Meltdown viola questo assunto, la più semplice ed efficace contromisura per impedire Meltdown è di non mappare lo spazio di memoria del kernel nello spazio di memoria virtuale di ogni processo. Questa modifica del Sistema Operativo era già stata proposta[2] ed è stata rapidamente adottata dai produttori di Sistemi Operativi. È da notare che questa non è la soluzione definitiva per Meltdown in quanto, ad esempio, nello spazio virtuale di ogni processo devono comunque essere presenti degli indirizzi di memoria privilegiati necessari per eseguire System Call ecc. Un effetto negativo di questa contromisura è che riduce le prestazioni dei sistemi: a seconda del tipo di elaborazione e di come la contromisura è stata implementata, si possono avere rallentamenti minimi o con incrementi sino al 30% del tempo di esecuzione dei programmi.

D'altronde altre soluzioni richiedono modifiche all'hardware, anche tramite microcodice di aggiornamento delle CPU, ad esempio per modificare il flusso di esecuzione delle istruzioni out-of-order o per separare più efficacemente l'area di memoria utilizzata dal kernel da quella utilizzata dagli altri processi.

Spectre

Se Meltdown permette di accedere a tutta la memoria del Sistema Operativo, è relativamente semplice da implementare ma al contempo sono presenti delle misure di mitigazioni efficaci, al contrario Spectre è molto complesso da implementare, non dà accesso a dati privilegiati del Sistema Operativo ma di altri processi e utenti sul sistema. Inoltre Spectre è implementabile su praticamente tutte le CPU moderne ed è difficile introdurre contromisure sia software che hardware senza modificare significativamente l'architettura delle moderne CPU.

Come Meltdown, Spectre si basa sull'abuso dell'esecuzione out-of-order delle istruzioni, ma in questo caso si basa specificatamente sul processo di esecuzione speculativa di alcune istruzioni. Ad alto livello, possiamo descrivere un attacco Spectre come segue:

1. l'attaccante e la vittima sono due utenti dello stesso sistema in grado di eseguire programmi sulla stessa CPU;
2. l'attaccante conosce un programma od una libreria utilizzata dalla vittima, ad esempio una libreria che contiene delle procedure crittografiche, e vuole ottenere le chiavi crittografiche segrete utilizzate dalla vittima;
3. l'attaccante individua nel codice utilizzato dalla vittima un branch che la CPU esegue tipicamente in maniera speculativa, ovvero in anticipo rispetto alla disponibilità dei valori per la valutazione della condizione di scelta; tipicamente la CPU sceglie di eseguire il ramo di codice che di recente è stato eseguito più frequentemente;
4. l'attaccante scrive un breve programma il cui codice esegue frequentemente le stesse istruzioni macchina del branch della libreria sotto attacco, ma con dati diversi in modo da indurre la CPU a ritenere più probabile l'esecuzione da parte della vittima sul branch errato; contemporaneamente l'attaccante svuota alcune aree della cache in modo che l'esecuzione del branch errato porti al caricamento in cache solo dei dati segreti;
5. quando la CPU esegue il programma della vittima, basandosi sull'esperienza dell'esecuzione del codice malevolo, segue il branch errato e carica in cache i dati segreti, quali ad esempio delle chiavi crittografiche;

6. a questo punto l'attaccante utilizza un attacco alla cache quale Flush+Reload o Evict+Reload come Covert Channel per leggere i dati segreti dalla cache.

L'implementazione di un attacco Spectre è molto complessa e difficile, anche se vi sono esempi persino in Javascript eseguiti all'interno di un browser Web da remoto [3].

Non di meno è difficile trovare delle contromisure a questo attacco: l'esecuzione di un branch errato di un processo non dovrebbe portare ad alcuna conseguenza: infatti non vi è alcun effetto per l'esecuzione del processo vittima, i dati calcolati nel branch errato sono scartati nell'esecuzione finale, ma inevitabilmente i dati del calcolo errato compaiono in cache per un breve periodo di tempo. Il problema è come evitare che un attaccante forzi una CPU a seguire un branch errato in modo da poi poter accedere, anche indirettamente, a dati presenti in cache. A differenza di Meltdown, in un attacco Spectre non sono coinvolte aree di memoria privilegiate, per cui la CPU e il Sistema Operativo non hanno informazioni hardware da poter eventualmente sfruttare per impedire l'attacco.

Quindi la soluzione definitiva di Spectre richiede un ridisegno delle logiche hardware di esecuzione out-of-order e speculative nelle CPU di prossime generazioni, anche se per alcune varianti di Spectre sono state identificate delle contromisure sia a livello di microcodice di aggiornamento delle CPU che per i Sistemi Operativi.

L'Hardware e la Sicurezza IT

Come descritto nel primo articolo, il disegno architetturale alla base delle moderne CPU risale agli anni '60 e negli ultimi anni sono state individuate delle vulnerabilità insite nel disegno stesso che minano alcuni dei principali assunti per la sicurezza nella gestione dell'elaborazione. Come spesso accade in questi casi, non è la singola funzionalità od il singolo blocco del disegno ad essere problematico, ma una combinazione non prevista di attività possono portare alla violazione dei requisiti di sicurezza assunti.

Ad oggi le conseguenze di Row Hammer, degli attacchi alla Cache, di Meltdown e Spectre sono più teoriche che pratiche, in quanto non sono stati realmente rilevati gravi incidenti ove queste vulnerabilità fossero la causa principale della violazione della sicurezza. Ancora oggi lo sfruttamento delle vulnerabilità dei Sistemi Operativi, delle applicazioni e delle configurazioni dei sistemi, è molto più semplice e produttivo per un attaccante.

D'altra parte non possiamo minimizzare il fatto che alcuni degli assunti principali per la sicurezza dei sistemi IT, assunti basati sul disegno e le funzionalità di base dell'Hardware, non possono più essere considerati completamente veritieri. Visto anche che la risoluzione di una vulnerabilità di disegno dell'Hardware richiede tempi lunghi e alti costi (dovendo ridisegnare l'Hardware, produrlo, adattare i Sistemi Operativi al nuovo Hardware e sostituire l'Hardware esistente) è necessario iniziare sin d'ora lo studio di nuovi disegni architetturali per l'Hardware IT o delle modifiche necessarie all'attuale, e di pianificare nell'evoluzione dei sistemi IT non solo gli aspetti di prestazioni, qualità e affidabilità ma anche nuove e più estese misure di sicurezza.

Riferimenti Bibliografici

[1] Alcuni riferimenti su “Spectre and Meltdown”:

- sito ufficiale: https://meltdownattack.com/
- Wikipedia: https://it.wikipedia.org/wiki/Spectre_(vulnerabilit%C3%A0_di_sicurezza) https://it.wikipedia.org/wiki/Meltdown_(vulnerabilit%C3%A0_di_sicurezza)
- Schneier on Security: https://www.schneier.com/blog/archives/2018/01/spectre_and_mel_1.html

[2] M. Lipp et al., “Meltdown”, https://arxiv.org/abs/1801.01207v1

[3] M. Schwarz et al., “Netspectre: Read Arbitrary Memory over Network”, https://gruss.cc/files/netspectre.pdf

[4] “Foreshadow: Breaking the Virtual Memory Abstraction with Transient Out-of-Order Execution”, https://foreshadowattack.eu/

Note

[1] Questo può avere conseguenze particolarmente critiche per sistemi di virtualizzazione a Container o para-virtualizzazione, ove il kernel del Sistema Operativo è comune a tutte le istanze virtuali.

[2] Ad esempio il Linux è chiamata Kernel Page Table Isolation (KPTI) o Kernel Address Isolation to have Side-channels Efficiently Removed (KAISER), ed era stata proposta per sopperire ad alcune limitazione del Kernel Address Space Layout Randomization (KASLR), introdotto a sua volta per impedire lo sfruttamento di alcuni tipi di vulnerabilità del kernel.

Articolo a cura di **Andrea Pasquinucci**