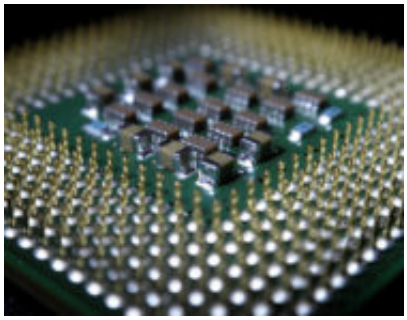
L'Hardware e la Sicurezza IT - Parte 1: un poco di storia

Author : Andrea Pasquinucci

Date : 25 settembre 2018



Il 3 gennaio 2018 sono state divulgate informazioni riguardo due vulnerabilità di sicurezza informatica chiamate “**Spectre**” e “**Meltdown**” [1] la cui origine è in alcune funzionalità dei microprocessori. “Spectre” e “Meltdown” sono pertanto vulnerabilità presenti nell'Hardware e risolvibili definitivamente solo con la sostituzione o modifica dell'Hardware stesso, anche se ovviamente sono possibili alcune mitigazioni software.

Per capire come si è arrivati a “Spectre” e “Meltdown”, perché sono così importanti e pericolosi, e apprezzare i principi su cui si basano, è utile partire dalle origini e riassumere velocemente un po' di storia dell'informatica.

La Nascita dell'Hardware Moderno

È negli anni '60 che vedono la luce i primi elaboratori la cui struttura di base è la progenitrice di quelli attuali. Sino ad allora gli elaboratori, oltre ad essere estremamente costosi e di enormi dimensioni, avevano quasi esclusivamente tre grandi clienti: i militari, le grandi aziende e le università. Questi elaboratori, che ora chiamiamo genericamente Mainframe, funzionavano secondo l'approccio chiamato “Batch”: i programmatori preparavano un “Job”, ovvero l'esecuzione di un programma od un gruppo di programmi, e caricavano nell'elaboratore[1] il codice da eseguire e tutti i dati necessari all'esecuzione. Nell'elaboratore risiedeva permanentemente un programma, il Sistema Operativo, con i compiti di caricare i Job, lanciarne l'esecuzione e riprendere il controllo dell'elaboratore al termine dell'esecuzione di un Job per avviare il Job successivo.

E' importante notare che solo un Job è caricato ed eseguito nell'elaboratore in ogni istante: durante l'esecuzione un Job ha a disposizione e controlla tutto l'hardware dell'elaboratore, non vi è condivisione alcuna delle risorse. Non essendoci condivisione di risorse né connessione ad altri elaboratori, non vi sono problematiche o possibili minacce di sicurezza. Gli unici possibili rischi sono dovuti a errori interni ai programmi stessi che possono portare a risultati errati o ad errori tali da far abortire l'esecuzione.

Ma questo approccio all'esecuzione dei programmi non è efficiente: tra un Job e l'altro la CPU, ai tempi la risorsa più costosa, poteva rimanere inutilizzata per molto tempo in attesa del caricamento e scaricamento di dati e istruzioni. Per ovviare a questo si introdussero, in Hardware, “pipeline” di esecuzione e CPU SuperScalari. Le “pipeline” sono componenti Hardware con lo scopo di caricare, preparare per l'esecuzione, e scaricare dati e istruzioni, mentre le CPU SuperScalari hanno molteplici unità di esecuzione, ad esempio una per eseguire calcoli su numeri interi, una su numeri a virgola mobile (floating point), una su numeri booleani ecc., in modo da poter eseguire in parallelo calcoli diversi.

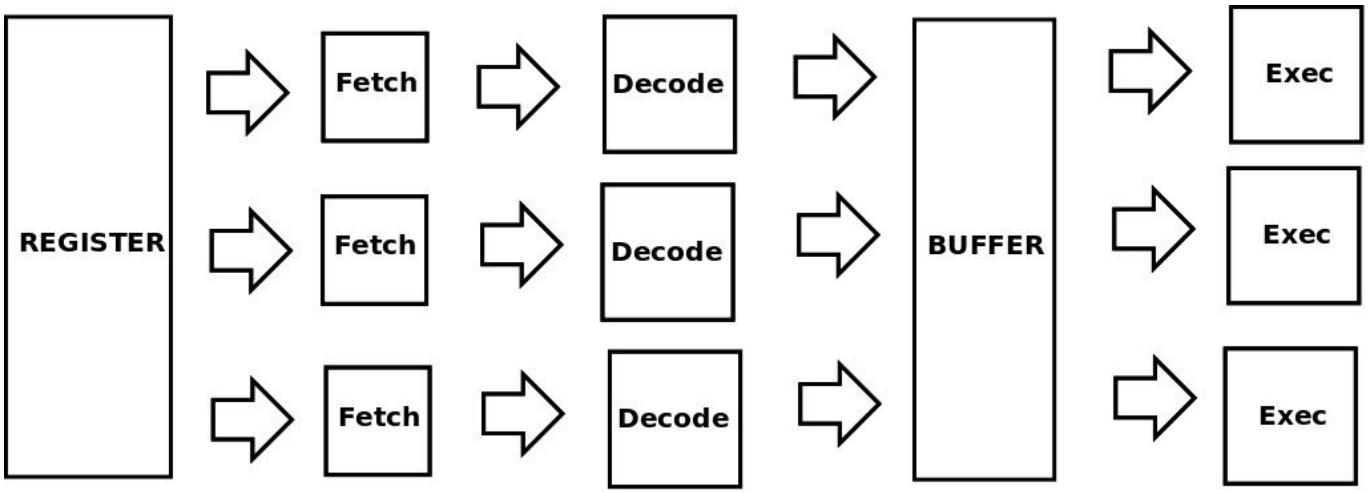


Fig. 1 – Modello di Pipeline [A.Pasquinucci]

Per gestire questi sistemi è necessario che il ruolo del Sistema Operativo aumenti: da puro sistema di carico/scarico di dati e istruzioni diventa il gestore delle code e dell'elaborazione, anche se sempre di un solo Job alla volta.

Si noti anche una caratteristica che diventerà importante per alcuni tipi di vulnerabilità “Spectre” e “Meltdown”: nelle CPU SuperScalari le istruzioni non sono sempre eseguite nell'ordine previsto dal programma, spesso anzi alcuni tipi di istruzioni, ad esempio calcoli su numeri a virgola mobile, sono eseguite in anticipo non appena disponibili i dati necessari. Questo è fatto perché se poi il calcolo risulta necessario, allora si è guadagnato del tempo perché il risultato è già pronto, altrimenti se il calcolo non è utilizzato dal programma, viene eliminato. La gestione dell'esecuzione delle istruzioni fuori dall'ordine del programma è demandata principalmente ai circuiti Hardware che fanno sì che per il Sistema Operativo ed i programmi l'esecuzione risulti sempre nell'ordine dato. La logica adottata in Hardware è che è meglio tenere impegnate le CPU anche con calcoli che nel caso non saranno utili, piuttosto che tenerle ferme: alcune volte ci sarà un grande guadagno di tempo, altre volte il guadagno sarà minimo, ma in ogni caso non si perderà mai tempo.

Modelli di Sicurezza IT

Prima di proseguire con l'evoluzione degli elaboratori, è utile rammentare quale fosse ai tempi l'approccio alla sicurezza informatica. Visto che gli elaboratori erano per lo più mono-processo,

mono-utente e non erano connessi tra di loro, le principali esigenze di sicurezza provenivano dai militari. Il principale modello di sicurezza militare è quello formalizzato e dimostrato da Bell - La Padula nel 1973, un modello di sicurezza a multi-livello: rispecchiando la gerarchia militare, il modello richiede che ognuno possa leggere solo i documenti al proprio livello ed ai livelli inferiori, mentre possa inviare nuovi documenti solo al proprio livello ed ai livelli superiori (write-up, read-down). Bell e La Padula dimostrarono che questo modello garantisce la riservatezza delle informazioni. Biba nel 1977 dimostrò che il modello opposto (write-down, read-up) garantisce l'integrità delle informazioni.

Vi sono altri modelli per la sicurezza delle informazioni: ad esempio il controllo di accesso discrezionale (DAC) prevede che ogni utente possa decidere chi può accedere e come alle proprie informazioni, usualmente tramite la configurazione di liste di controllo accesso (ACL). Questo approccio è stato adottato ad esempio dai Sistemi Operativi di tipo Unix.

Più in generale si è giunti a formulare il concetto di sistemi “di fiducia” (“Trusted”) che prevedono che vi sia un insieme di regole di sicurezza (“Trusted Computing Base”, TCB) gestite dal responsabile della sicurezza, e che all'interno del Sistema Operativo vi sia un processo sicuro (“Reference Monitor”) che verifica che l'esecuzione di ogni istruzione soddisfi la TCB. Questi concetti sono anche formulati nei requisiti e certificazioni di sicurezza quali gli Orange Book e i Common Criteria [2, 3].

Più avanti sarà necessario approfondire alcuni aspetti dell'approccio ai sistemi Trusted, TCB e Reference Monitor. Per ora però è importante ricordare l'approccio alla sicurezza come descritto dai sistemi a multi-livello.

Multiprocesso e Multiutente

Ritornando alla nostra breve storia, il passo successivo nell'evoluzione degli elaboratori è l'esecuzione contemporanea di più programmi, ovvero di più Job ognuno di un utente diverso. Questo è il passo cruciale per la nascita degli elaboratori moderni ed il principale attore di questo passo è il Sistema Operativo. L'evoluzione del Sistema Operativo lo porta ad adempiere principalmente a due compiti:

1. **Macchina Estesa** (o macchina virtuale): la scrittura di programmi direttamente in linguaggio macchina o in linguaggi appena più evoluti richiede comunque la conoscenza dell'Hardware e del suo funzionamento, questo rende molto complessa la scrittura dei programmi; invece il Sistema Operativo può interporre un’interfaccia applicativa, che semplifica e nasconde le complessità dell'Hardware rendendo più semplice la scrittura dei programmi; non solo, permette anche di eseguire lo stesso programma su Hardware diversi;
2. **Gestore delle Risorse**: se si vogliono eseguire più programmi allo stesso tempo, è necessario che il Sistema Operativo, che ha il compito di caricarli e scaricarli dalla CPU, li gestisca durante tutto il tempo della loro esecuzione, non solo all'inizio ed alla fine, ed in maniera molto precisa e dettagliata.

E' necessario approfondire come il Sistema Operativo svolge il compito di Gestore delle

Risorse. Prima di tutto, all'interno di una CPU (e per ora consideriamo solo elaboratori con una CPU ciascuno) viene eseguito un solo programma alla volta. Vengono però caricati in memoria più programmi e la CPU ne alterna l'esecuzione. Il Sistema Operativo decide quale programma deve essere eseguito, per quanto tempo, e verifica che durante l'esecuzione un programma non tocchi i dati degli altri programmi ne utilizzi periferiche Hardware in modo inappropriato. L'alternanza di esecuzione dei programmi permette di rendere molto più efficiente l'utilizzo delle risorse, ad esempio mentre un programma legge o scrive dei dati, un altro viene eseguito nella CPU. Il Sistema Operativo quindi gestisce l'accesso a tutte le risorse Hardware da parte dei programmi in modo che questo sia efficiente e che al contempo l'integrità, riservatezza e disponibilità dei dati siano garantite. Questo deve essere fatto dal Sistema Operativo in modo tale che durante l'esecuzione un programma possa comunque disporre di tutte le risorse Hardware, come se fosse eseguito da solo, in assenza di altri programmi.

Multics

Nel 1965 MIT, AT&T, IBM e GE decisero di sviluppare in comune un sistema operativo 'sicuro' chiamato Multics. Questo sistema operativo doveva essere:

1. Progettato top-down
2. Capace di supportare almeno 1000 utenti contemporaneamente
3. 'Reliable'
4. 'With sufficient control of access to allow selective sharing of information'

ed avere molte altre caratteristiche di sicurezza militare. Il progetto non ebbe successo commerciale, uno dei motivi fu che le richieste di sicurezza intrinseca al sistema, garantite da uno sviluppo controllato top-down, portarono ad un’eccessiva complicazione del software rispetto alle piattaforme hardware disponibili in quegli anni. Così nel 1969 AT&T convinse i partner a chiudere lo sviluppo di Multics, anche se Multics venne sviluppato in maniera indipendente ancora per una decina di anni.

Multics fu però un passaggio molto importante nella storia dei Sistemi Operativi, anche per essere il padre di Unix (e la somiglianza dei nomi non è casuale).

Ma come Multics avrebbe potuto soddisfare questi requisiti? Come poteva un Sistema Operativo gestire le risorse, garantire l'isolamento tra programmi ed al contempo permettere che un programma utilizzasse tutte le risorse del sistema?

La soluzione che si adottò allora anche per Multics, e che tuttora è alla base dell'Hardware IT, è simile all'approccio alla sicurezza multi-livello citato precedentemente. In particolare Multics fu progettato con 8 livelli, chiamati Anelli di Protezione (“Protection Rings”) ed implementati in Hardware (vedi Fig. 2). L'anello più interno, convenzionalmente numerato con 0, è l'anello con maggiori privilegi, il livello massimo di sicurezza, mentre l'anello più esterno, il 7, ha il livello minore di privilegi. Ogni programma viene caricato ed eseguito solo ed esclusivamente in un livello, ma può chiamare altri programmi che sono eseguiti in altri livelli (tipicamente superiori). E' l'Hardware che isola i livelli, e non permette ad un programma eseguito ad un certo livello di accedere direttamente alle risorse di un altro livello. In ogni istante nell'Hardware vi è un

indicatore che segna a quale livello sono eseguite le istruzioni. Per accedere a risorse gestite da un altro livello, un programma effettua una richiesta, ovvero una chiamata ad una funzione speciale (System Call) che attiva una chiamata Hardware dedicata (Call Gate). Queste chiamate Hardware permettono di accedere a specifiche routine ad un determinato livello: se i parametri passati con la richiesta sono corretti, allora la richiesta è accettata e viene eseguito il comando al nuovo livello. Ad esempio per leggere e scrivere su disco, un programma in esecuzione ad un livello esterno esegue una System Call al Sistema Operativo a livello 0 chiedendo di leggere/scrivere i dati. E' quindi il Sistema Operativo stesso che esegue la lettura / scrittura e poi ritorna i dati e l'esecuzione al programma al procedente livello.

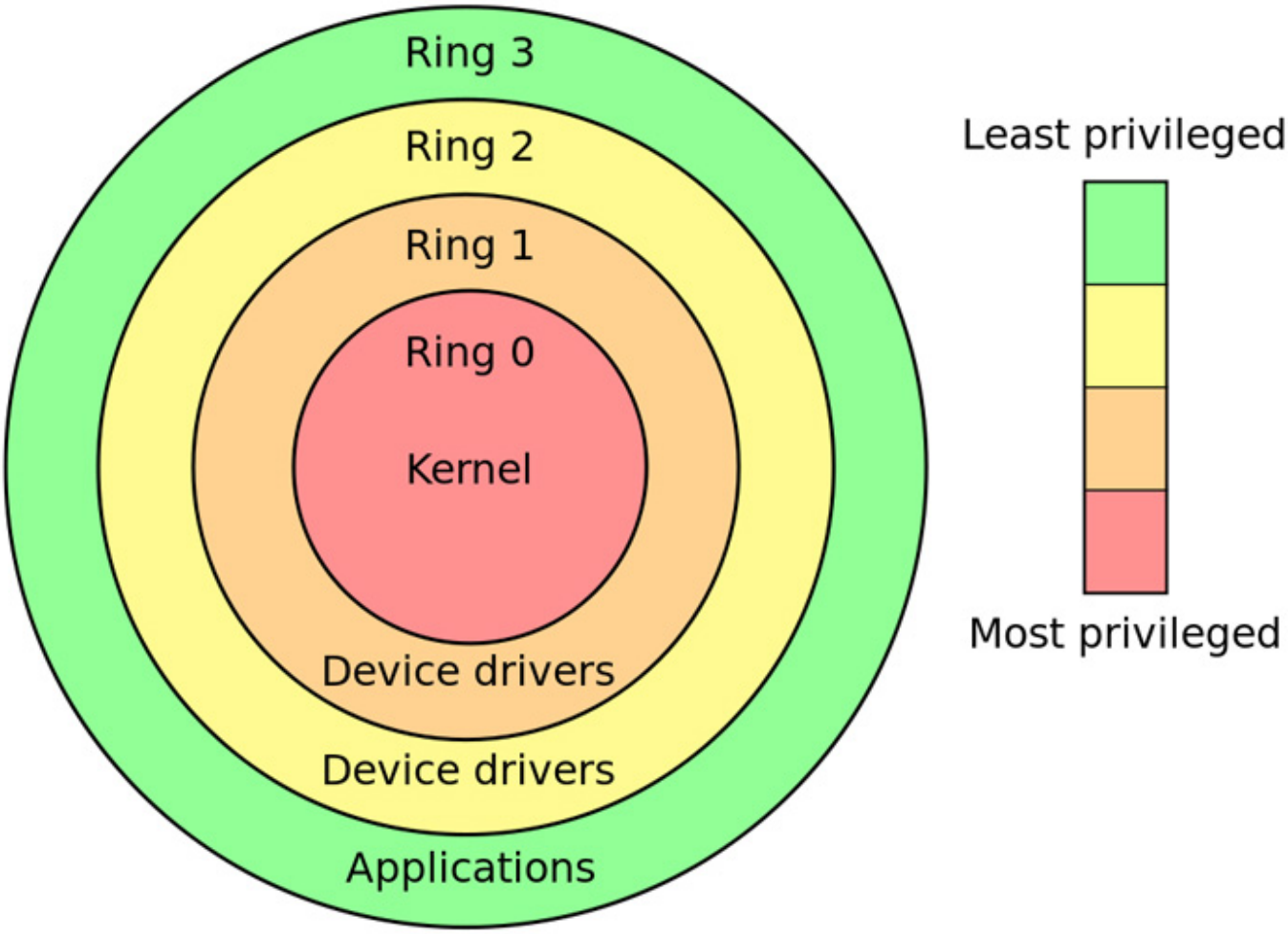


Fig. 2 – Anelli di protezione per le CPU Intel x86 [Wikipedia]

In questo modo l'Hardware garantisce che il Sistema Operativo, a livello 0, gestisca direttamente tutte le risorse, sia in grado di fornire un’interfaccia applicativa che virtualizza e semplifica l'accesso alle risorse, garantendo allo stesso tempo l'integrità, riservatezza e disponibilità dei dati. Senza la separazione tra Sistema Operativo e altri programmi fornita dall'Hardware, non sarebbe possibile garantire la sicurezza dell'esecuzione contemporanea di programmi né la separazione tra programmi e dati di utenti diversi.

La maggior parte delle moderne CPU adotta una struttura ad Anelli di Protezione, a partire ad esempio dalle CPU Intel x86 che hanno 4 anelli. In generale però gli attuali Sistemi Operativi non sfruttano tutti gli Anelli, ma ne usano solo due, tipicamente quello più privilegiato e quello meno privilegiato. Si dice che il Sistema Operativo opera in “**kernel mode**” (ring 0) mentre tutti gli altri programmi sono eseguiti in “**user mode**”. Ci sono alcuni motivi per cui in pratica si usano solo due anelli tra cui la semplicità di gestione del codice, la portabilità del codice su diversi modelli di CPU, la maggiore facilità nel gestire molti privilegi in software piuttosto che in Hardware.

La ricerca di maggiori prestazioni insieme all'estensione delle capacità di calcolo delle CPU, all'esplosione del numero di periferiche e di utilizzi degli elaboratori, ha portato ad una crescita quasi incontrollata della dimensione dei Sistemi Operativi eseguiti in Kernel Mode. Ovviamente le dimensioni e la complessità sono tra i primi nemici della sicurezza, e questo si riscontra quotidianamente nelle vulnerabilità di sicurezza che si scoprono nei Sistemi Operativi.

La gestione della Memoria Dinamica

La presenza di Gate e System Call non è però sufficiente a raggiungere gli obiettivi del progetto Multics, è anche necessario che durante l'esecuzione ogni programma possa accedere, anche solo virtualmente, a tutto l'Hardware, in particolare a tutta la memoria dinamica. La gestione della memoria (principalmente la RAM, Random Access Memory, e le Cache) è un argomento molto complesso ma per i nostri scopi è necessario farne almeno un accenno di cui avremo assolutamente bisogno per capire il funzionamento di “Spectre” e “Meltdown”.

L'idea di partenza è quella di dare ad ogni programma accesso a tutta la memoria in maniera virtuale. Ogni programma può quindi allocare qualunque indirizzo (virtuale) della memoria, questi indirizzi però non sono reali, vengono tradotti in tempo reale negli indirizzi di memoria realmente utilizzati da una componente Hardware chiamata Unità di Gestione della Memoria (MMU) con l'ausilio di tabelle Hardware di conversione (ad esempio TLB, LTD, GDT ecc.). In questo modo un programma pensa di utilizzare tutta la memoria, ma in realtà solo l'Hardware ed il Sistema Operativo sanno veramente quali locazioni di memoria fisica sono utilizzate dal programma, ed in quale memoria: RAM, Cache (e di quale livello) od anche disco nel caso di swap.

L'approccio di Multics ed anche di molti dei processori moderni quali ad esempio quelli della famiglia Pentium Intel, è quello di utilizzare un processo di gestione della memoria chiamato Segmentazione con Paginazione (“Segmentation with Paging”) che implementa sia il mapping tra indirizzi virtuali e fisici che la gestione della memoria fisica, con lo spostamento delle pagine di memoria da una memoria all'altra a seconda della necessità di uso.

Infatti le memorie fisiche sono organizzate in maniera gerarchica: dallo swap su disco, memoria più lontana dalla CPU e più lenta, alla RAM, ai vari livelli di Cache di cui L1 è quello più vicino alla CPU, più veloce ma tipicamente piccolo (vedi Fig. 3).

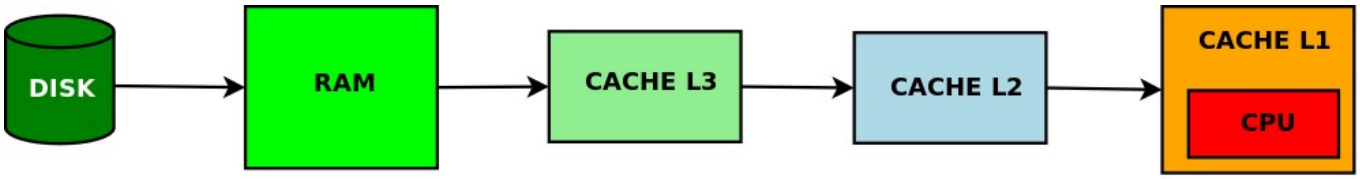


Fig. 3 – Gerarchia delle memorie [A.Pasquinucci]

Il Sistema Operativo fa sì che vengano caricati nella Cache L1 le istruzioni ed i dati che servono all'esecuzione del programma. Si definisce “Cache Hit” quando il processore trova nella Cache L1 i dati o le istruzioni di cui ha bisogno. Quando invece il processore non trova nella Cache L1 i dati o le istruzioni di cui ha bisogno per proseguire l'elaborazione, si dice che si verifica un “Cache Miss”.

Nel caso di Cache Miss, il Sistema Operativo deve dapprima liberare sufficiente spazio nella Cache per poter caricare i nuovi dati, e questo richiede trovare spazio nelle Cache inferiori, nella RAM o su disco, ove scrivere i dati. Poi trovare nelle Cache inferiori, nella RAM o su disco i dati da caricare e copiarli nella Cache L1. Dopodiché può essere ridato il controllo al programma che può procedere nell'esecuzione. E' chiaro che nel caso di Cache Miss i tempi di esecuzione di una istruzione sono molto più lunghi, anche 100 volte, che nel caso di Cache Hit.

Tutto ciò è ulteriormente complicato dal fatto che, come indicato inizialmente, le “pipeline” di esecuzione del codice sono molteplici e non è detto che le istruzioni vengano eseguite nell'ordine previsto dal programma, e questo è fatto per utilizzare al massimo le CPU.

Per “Spectre” e “Meltdown” è necessaria qualche ulteriore informazione tecnica su come è gestita la memoria di un elaboratore. Come già indicato, la memoria è tipicamente organizzata in Pagine e Segmenti. Una Pagina è un’unità di memoria sia fisica che virtuale, spesso di 4KB, che viene gestita dall'Hardware come un blocco unico con un unico indirizzo. Questo semplifica l'indirizzamento della memoria e le attività di copia dei dati tra le diverse Cache e memorie. I Segmenti invece sono spazi di memoria virtuali, gestiti dall'Hardware, che permettono ad un programma di avere a disposizione più di uno spazio di memoria con indirizzi consecutivi che partono da 0. I Segmenti possono avere dimensioni (virtuali) differenti a seconda delle necessità del programma. Un tipico esempio di uso di un Segmento è quello delle librerie condivise (“Shared Libraries”): alcune librerie di sistema, ad esempio quelle grafiche, sono spesso utilizzate da molti programmi contemporaneamente. In assenza di Segmenti, ogni programma deve caricare nel proprio spazio virtuale una copia della libreria, mentre se l'Hardware supporta i Segmenti, il Sistema Operativo può caricare una libreria in un Segmento (read-only) ed includere il Segmento nello spazio virtuale di memoria di tutti i programmi che lo utilizzano, velocizzando le operazioni e riducendo l'utilizzo di memoria. Visto che un Segmento può essere molto grande, in CPU Intel x86 i segmenti sono di 4GB dimensione superiore a quella della Cache L1, è necessario suddividere in Pagine i Segmenti in modo da gestire il caricamento dei dati nelle diverse memorie.

Un ultima osservazione, nelle attuali CPU a 64 bit, l'utilizzo dei Segmenti non è più veramente

necessario poiché lo spazio di memoria a 64 bit indirizzabile è veramente enorme e i dati possono essere distribuiti direttamente in blocchi di memoria con indirizzo di base differente. I Segmenti sono comunque utilizzati per la protezione della memoria, ad esempio indicano se un'area di memoria può essere acceduta solo in “kernel mode” od anche in “user mode”.

Sicurezza e Hardware

L'Hardware ha quindi un ruolo fondamentale ed indispensabile nel garantire la sicurezza, intesa come riservatezza, integrità e disponibilità dei dati e delle risorse. In particolare l'Hardware:

- isola il Sistema Operativo da tutti gli altri programmi;
- permette solo al Sistema Operativo di accedere all'Hardware ed a tutte le risorse del sistema;
- obbliga tutti gli altri programmi ad accedere alle risorse tramite il Sistema Operativo che può quindi sia gestire le risorse fisiche (integrità e disponibilità) sia verificare i permessi di accesso alle risorse assumendo il ruolo di Reference Monitor (riservatezza);
- gestisce, insieme al Sistema Operativo, la concomitanza di esecuzione di più programmi contemporaneamente ed in particolare della memoria dinamica.

Cosa succederebbe se un programma potesse anche solo leggere i dati di un altro programma in esecuzione, senza passare attraverso il Sistema Operativo? Dovremmo sicuramente definire questa una vulnerabilità Hardware.

Note

[1] I programmi e i dati erano tipicamente caricati per mezzo di schede cartacee perforate lette da opportune unità di input dell'elaboratore.

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] The Orange Book, DoDD 5200.28-STD, pubblicato nel 1983 e aggiornato 1985 dal National Computer Security Center (NCSC), parte della National Security Agency (NSA), http://csrc.nist.gov/publications/secpubs/rainbow/std001.txt

[3] “Common Criteria for Information Technology Security Evaluation” (anche noto come “Common Criteria” o CC), ISO/IEC 15408,

http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=50341

[4] A.S. Tanenbaum, “Modern Operating Systems”, Prentice Hall

A cura di: **Andrea Pasquinucci**