L'Hardware e la Sicurezza IT - Parte 2: Attacchi Side Channel, Row Hammer ed attacchi alla Cache

Author : Andrea Pasquinucci

Date : 29 ottobre 2018

La sicurezza dei sistemi IT è principalmente conosciuta in relazione ad attacchi esterni od interni che sfruttano vulnerabilità del codice od abuso di privilegi. Abbiamo tutti sentito parlare di virus, worm e malware di vario tipo che sfruttano vulnerabilità dirette di sistemi e applicazioni. Un'altra grande categoria di attacchi noti a tutti sono quelli che genericamente possiamo indicare con il termine “Phishing”, ovvero truffe e inganni che convincono l'utente di un sistema IT a fornire all'attaccante informazioni riservate od a sottrarre danaro. Inoltre una categoria di attacchi realizzabili con i sistemi IT sono quelli svolti dall'interno (“Insider attack”) ovvero da persone che abusano dei propri privilegi sui sistemi IT per danneggiare l'organizzazione per cui lavorano a proprio favore od a favore di terzi.

Nell'ambito dei sistemi IT vi è però un'altra importante categoria di minacce e attacchi anch'essa di grande rilevanza e lunga storia, sono gli attacchi “Side Channel” (ovvero condotti tramite un canale laterale) e di inferenza. Invece di sfruttare una vulnerabilità esplicita e diretta di un sistema, un attacco Side Channel sfrutta delle caratteristiche indirette del sistema per giungere comunque alla sua compromissione. Gli esempi sono molti, ma è comunque utile ricordarne alcuni dei principali.

Una delle principali famiglie di attacchi Side Channel va ora sotto il nome di TEMPEST e riguarda metodi per spiare, ovvero estrarre informazioni da un sistema IT, sfruttando l'emissione dal sistema di segnali radio, elettrici, termici, di vibrazioni ecc. [1,2]. TEMPEST è un nome in codice dell'NSA (National Security Agency, USA) che si riferisce ad un insieme di misure da implementare, di verifiche da effettuare su strumenti informatici, e relative certificazioni, per prevenire alcune classi di attacchi Side Channel. I primi studi delle minacce di tipo TEMPEST risalgono ai primi anni del 1900 quando si incominciarono a sviluppare metodi alternativi per intercettare messaggi trasmessi via telefono o radio. E' però principalmente negli anni '60 e '70 che TEMPEST fu ideato e sviluppato.[1]

Uno dei principali canali di attacchi Side Channel è quello delle radiazioni elettromagnetiche. Ad esempio è stato mostrato che è possibile ricostruire quanto mostrato su di un monitor non

schermato anche a distanza di metri e attraverso muri [3]. Analogamente sfruttando l'emanazione di radiazioni elettromagnetiche è stato possibile intercettare quanto in transito attraverso modem non schermati, ma anche attraverso cavi di connessione. Inoltre smart-card, schede FPGA, CPU, chip crittografici ecc. emettono radiazioni elettromagnetiche che, se non schermate, possono permettere l'intercettazione delle informazioni.

E' possibile intercettare informazioni anche monitorando l'utilizzo di corrente elettrica da parte dell'hardware ed in alcuni casi monitorando l'emissione di onde sonore o di emissioni termiche.

Recentemente è stato mostrato [4] come la ricostruzione di quanto presente su di un monitor, anche di uno smartphone, è possibile analizzando i rumori acustici di fondo emessi dagli schermi e registrati da comuni microfoni presenti nei nostri dispositivi quotidiani.

Le principali misure di sicurezza TEMPEST sono

1. La schermatura dei sistemi
2. La modifica dei processi software o dei componenti hardware in modo da non emettere radiazioni che possano portare o far dedurre informazioni.

Un altro esempio recente [6] riguarda lo studio dei chip dedicati alle comunicazioni Wireless, come WiFi e Bluetooth. Questi chip includono sullo stesso substrato sia una componente ricetrasmittente analogica che una componente digitale utilizzata anche per la cifratura del traffico. E' stato notato che la presenza sullo stesso substrato di silicone di entrambe le componenti porta in molti chip alla presenza di interferenze fra le due componenti, generate dalla componente digitale e amplificate dalla componente analogica. Ne risulta quindi che studiando il “rumore” della trasmissione radio è possibile identificare i segnali emessi dalla componente digitale del chip durante la cifratura / decifrazione del traffico. In alcuni casi questo fenomeno può portare alla identificazione della chiave segreta di cifratura anche a qualche metro di distanza.

Vi sono anche attacchi Side Channel non direttamente fisici come i precedenti, ma implementati tramite monitoraggio delle applicazioni IT o dell'attività dell'Hardware stesso. Una classe importante di questi attacchi è quella che utilizza la misura dei tempi di elaborazione in particolare quando applicati alla crittografia. Sin a partire da [5] è stato mostrato come conoscendo l'algoritmo di cifratura e misurando i tempi di esecuzione dell'algoritmo crittografico, sia possibile da parte di una terza parte presente sul sistema, risalire alle chiavi di cifratura. Questo può essere evitato se l'algoritmo crittografico esegue le elaborazioni in tempo costante, ovvero in modo indipendente dai dati in ingresso (incluse le chiavi), od in alcuni casi con una modifica dell'algoritmo chiamata “Bliding” che introduce dei dati casuali all'interno dell'elaborazione, poi rimossi, in modo da rendere inintelligibile il risultato dell'attacco tramite misura dei tempi di elaborazione.

Row Hammer

A partire dal 2015 con lo studio di Kim et al. [7] sulla vulnerabilità poi chiamata Row Hammer, cresce grandemente l'interesse sia accademico che dei professionisti di sicurezza informatica, a

partire dai componenti del Project Zero di Google, all''interazione tra funzionalità hardware in particolare delle CPU e sue componenti, e le violazioni del modello di sicurezza ad anelli descritto nella prima parte di questo articolo.

L'osservazione alla base dell'effetto Row Hammer è che la miniaturizzazione delle componenti hardware, in particolare delle DRAM, può portare ad errori dovuti a disturbi provenienti da locazioni vicine.

Semplificando il più possibile la descrizione del processo fisico, si può dire che in una cella DRAM il valore 0 o 1 è rappresentato dalla presenza o meno di carica elettrica in un condensatore. Ogni cella che rappresenta 1 bit è composta da un condensatore e da un transistor, e le celle sono organizzate in righe. Le celle di una riga vengono lette / scritte insieme in un'unica operazione (si veda la Figura 1). Le operazioni di lettura in realtà rimuovono fisicamente le cariche dai condensatori che poi devono essere ricaricati, ovvero riscritti, per mantenere il dato. Ma i condensatori, soprattutto quelli così piccoli, si scaricano velocemente, per cui è necessario rinfrescare i dati, ovvero rileggerli/riscriverli, molto frequentemente. Ad esempio DDR3 DRAM hanno un periodo di refresh di 64 millisecondi, mentre sono necessari circa 50 nanosecondi per leggere / scrivere una riga di memoria.

Per ingrandire la capacità delle memorie, e ridurne il costo, è necessario ridurne le dimensioni, ma questo vuol dire avvicinare tra loro le celle. I produttori di DRAM si sono sempre preoccupati[2] di minimizzare i fenomeni in cui effetti elettromagnetici di una cella influiscono sulle celle limitrofe, chiamati anche Errori di Disturbo (“Disturbance Errors”). Questi errori sono stati ridotti sia con misure fisiche di isolamento delle celle, sia con il processo di rinfrescamento dei dati, sia con l'introduzione di memorie con ECC (Error Correction Code), in grado di correggere 1 o 2 errori per riga / parola. Le ECC DRAM sono però più costose e un po' più lente delle memorie non-ECC e sono utilizzate tipicamente solo in server di alto livello.[3]

I produttori di DRAM hanno sempre considerato errori prodotti casualmente o da fenomeni fisici esterni, ma non hanno però considerato che qualcuno potesse causare volontariamente degli errori di disturbo per violare la sicurezza del sistema. Invece questo è proprio quello che Kim et al. si sono proposti di fare nello studio [7].

Quello che Kim et al. hanno notato è che leggendo un bit[4] alcune celle nelle righe limitrofe perdono carica elettrica ad una velocità leggermente più veloce di quanto succede normalmente. Se il bit viene letto alcune volte nell'arco di un periodo di rinfrescamento, la perdita di carica elettrica nelle righe limitrofe è minima e non produce alcun effetto. Ma se invece si ripete la lettura dello stesso bit o della stessa riga, centinaia di migliaia di volte nell'arco di un periodo di rinfrescamento, allora alcune celle possono scaricarsi completamente, cambiando quindi il loro valore.

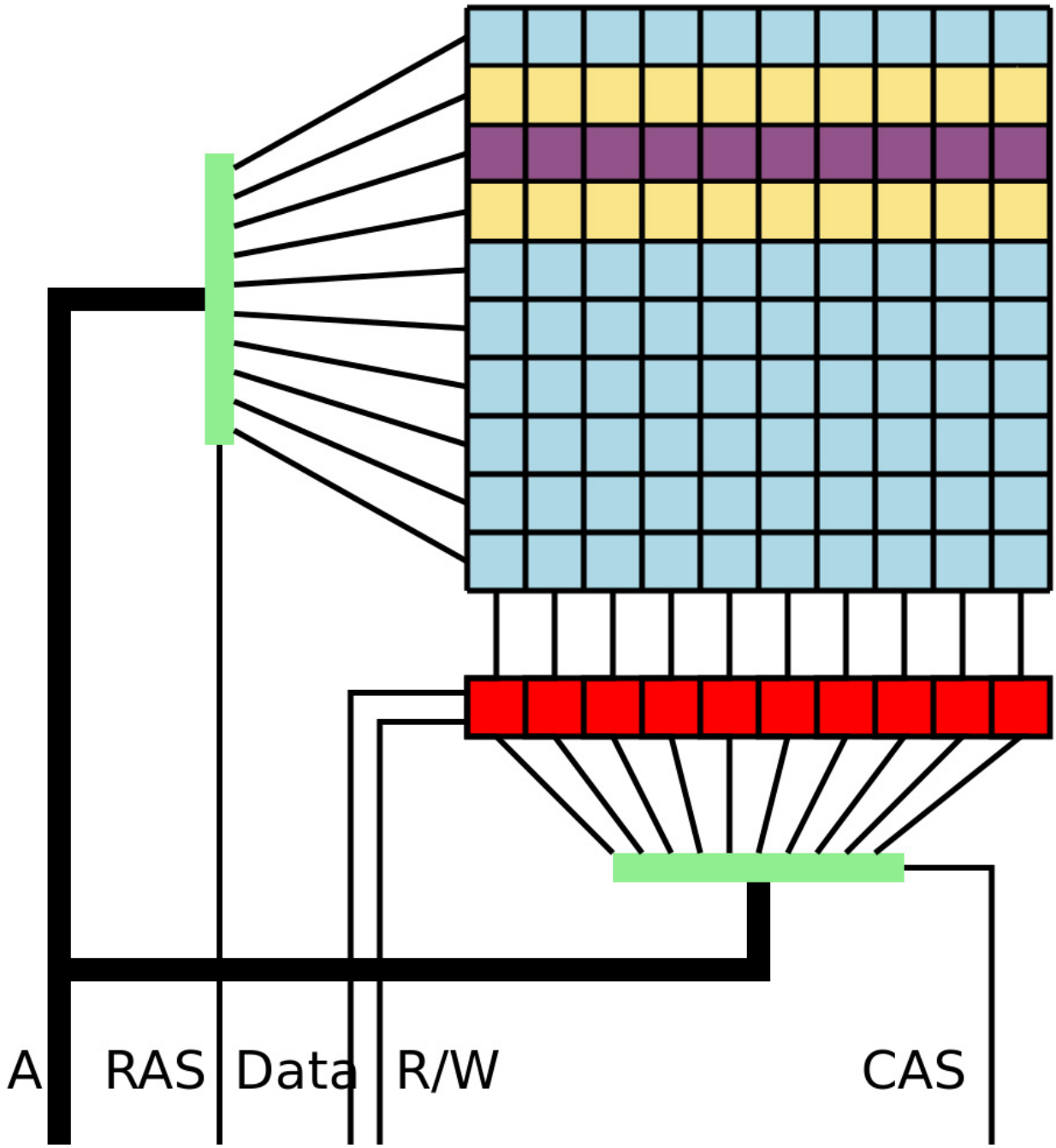

Fig. 1 Esempio di attacco Row Hammer: la lettura rapida delle righe gialle può indurre il cambio di valore di alcune celle viola, la riga rossa indica il “row buffer” [Wikipedia]

Per riuscire a leggere migliaia di volte lo stesso bit, od un bit nella stessa riga, nell'arco di un periodo di rinfrescamento è necessario però aggirare le misure di miglioramento delle prestazioni presenti nell'hardware. In particolare all'interno di ogni DRAM sono presenti dei “row buffer” (anche chiamati “sense-amplifier”) che mantengono copia di una riga in modo che letture immediatamente successive siano servite dal quest'ultimo e non direttamente dalla riga. Inoltre, come descritto nella prima parte di questo articolo, una volta letto un dato dalla RAM, questo viene caricato nelle varie Cache presenti nella CPU, in modo che letture

immediatamente successive dello stesso dato non accedano per nulla alla RAM ma ad una Cache e siano molto più veloci.

E' possibile, anche se non facile, scrivere programmi che aggirino le Cache, i “row buffer” ecc. In [8] sono riportati alcuni riferimenti ad articoli che descrivono tecniche per implementare attacchi Row Hammer. Sono stati ideati anche attacchi in javascript all'interno di browser, via network, che utilizzano le GPU, su smartphone ecc.

La difficoltà di implementazione degli attacchi non ha però evitato di trovare modi di renderli molto pericolosi. L'idea principale è quella di modificare un bit in memoria che garantisce la sicurezza del sistema: ad esempio se si riesce a modificare un bit che indica che un settore di memoria è privilegiato, allora qualunque processo può accedere a quei dati riservati. Modificando quindi opportuni bit in memoria è possibile sia accedere a dati riservati al kernel o di altri utenti e processi, ed anche accedere direttamente all'anello 0 (kernel mode) senza passare da System Call e Gate, e così via. Potenzialmente quindi gli attacchi Row Hammer possono completamente aggirare qualunque misura di sicurezza e permettere all'attaccante qualunque azione sul sistema. Le possibili conseguenze di un attacco Row Hammer includono: l'accesso amministrativo al sistema operativo, l'accesso a chiavi crittografiche e dati riservati, l'accesso da una macchina virtuale alle applicazioni e dati di un'altra macchina virtuale sullo stesso Host ecc.

Per implementare un attacco Row Hammer il problema principale quindi è quello di identificare esattamente i bit in memoria fisica[5] DRAM che si vogliono modificare, ed ideare una procedura in grado di modificare solo quelli. La difficoltà di esecuzione di questi attacchi li rende quindi generalmente poco efficaci, ma come è stato dimostrato, sicuramente possibili.

Contromisure per Row Hammer

Bisogna innanzitutto notare che non tutte le memorie DRAM sono vulnerabili ad un attacco Row Hammer, dipende non solo dal modello di memoria ma in alcuni casi anche dalla singola memoria.

In generale la soluzione definitiva per eliminare la vulnerabilità è quella di produrre nuove memorie DRAM che per costruzione non siano suscettibili a Row Hammer. Questo però non è semplice in quanto le richieste del mercato sono di maggiore capacità, minore dimensione e minor costo delle memorie, e non è facile sviluppare nuove memorie che siano al contempo anche non vulnerabili a Row Hammer. Inoltre le memorie ECC, come già indicato, garantiscono di correggere errori ad 1 o 2 bit per parola / riga, mentre gli attacchi Row Hammer tipicamente includono più bit e quindi possono aver successo anche su queste memorie.

Come mitigazione Hardware, in alcune nuove CPU e memorie sono state introdotte istruzioni che permettono di monitorare la frequenza di accesso alle righe di memoria e, se questa supera certe soglie, di forzare l'esecuzione di un rinfrescamento, il che impedisce l'effetto Row Hammer.

Sono anche state proposte ed implementate molteplici altre misure di protezione. Tra le più

semplici misure di protezione vi sono le seguenti:

- il dimezzamento del tempo di refresh, ad esempio ogni 32 millisecondi, riduce l'efficacia di molti tipi di attacchi Row Hammer, potenzialmente a costo di rallentare l'accesso alla memoria;
- l'introduzione di procedure per verificare se le memorie sono vulnerabili ad attacchi Row Hammer, in modo da implementare misure di protezione solo su queste;
- visto che l'attacco Row Hammer esegue delle attività molto particolari, l'introduzione di misure di monitoraggio degli accessi alla RAM o dell'utilizzo delle Cache (che devono essere svuotate o aggirate dall'attaccante) o di Buffer di memoria, per identificare e bloccare i processi che eseguono un attacco Row Hammer;
- analogamente i sistemi operativi possono limitare l'accesso o l'uso di istruzioni utili ad implementare Row Hammer, come le istruzioni *clflush* e *kmalloc heap* in linux;
- infine è possibile che il sistema operativo separi fisicamente nelle memorie i dati con privilegi diversi o di diversi utenti (ad esempio per macchine virtuali) in modo che un attaccante non possa leggere dati in una riga confinante ad una con altri privilegi, anche se questo può introdurre molte complicazioni alla già non semplice gestione della memoria.

Attacchi alla Cache

Come abbiamo visto, i dati sono gestiti non solo nella RAM ma anche in una gerarchia di Cache, tipicamente 3, ciascuna più piccola della precedente ma più veloce. L'ultima cache, usualmente la terza "L3" anche detta Last Level Cache "LLC", quella più lontana dalla CPU e più vicina alla RAM, è quella di maggior interesse in quanto tipicamente contiene tutti i dati presenti anche nelle Cache inferiori e per tutti i Core presenti sul sistema. Sono stati ideati alcuni tipi di attacchi che sfruttano la caratteristica principale della Cache LLC, ovvero quella di essere più veloce della RAM. I più noti tra questi attacchi sono "Prime + Probe", "Flush + Reload" e "Flush + Flush" [9]. L'idea di base di questi attacchi è di misurare il tempo di lettura di un dato, se il dato è già presente in Cache il tempo di lettura è molto inferiore al caso in cui il dato non sia presente in Cache ed il sistema operativo debba leggerlo nella RAM e copiarlo nella Cache.

Sono state ideate due classi di attacchi, entrambe violano la riservatezza dei dati ma non permettono la modifica degli stessi e l'aumento di privilegi di un processo. Gli attacchi alla Cache non sono così distruttivi come Row Hammer, ma possono comunque avere conseguenze molto gravi.

Il primo tipo di attacco permette di creare un "Covert Channel", ovvero un canale di comunicazione non previsto, non facilmente rilevabile e che può violare le politiche di sicurezza del sistema. Un esempio eclatante è la realizzazione di una connessione Secure Shell (SSH) tra due macchine virtuali eseguite sullo stesso Host, macchine virtuali tra le quali non dovrebbe esserci alcuna comunicazione [10]. Questo canale di comunicazione è stabilito tramite un attacco alla Cache dell'Host che ospita le due macchine virtuali.

L'idea di base di questo tipo di attacco assomiglia un poco all'utilizzo del codice Morse, ovvero

alla codifica dell'informazione in segnali brevi e lunghi, rispettivamente presenza o assenza di un dato nella Cache LLC.

Si può descrivere brevemente l'attacco come segue: l'attaccante è in grado di installare su entrambe le macchine virtuali il proprio codice (eventualmente diffuso anche tramite malware). L'attaccante deve identificare un'area fisica della Cache da utilizzare per l'attacco. Questo è forse il punto più complesso dell'attività in quanto, come descritto, l'accesso alla Cache è tramite la memoria virtuale, mentre la Cache è organizzata secondo la memoria fisica. Inoltre i due programmi sono eseguiti su due macchine virtuali distinte per cui il mapping tra memoria virtuale e fisica effettuato sia dal sistema operativo che dall'Hypervisor usualmente differisce.[6] Nondimeno da un'analisi dell'hardware e di come sistemi operativi e Hypervisor gestiscono la memoria e le Cache, è possibile in alcuni casi trovare procedure tecniche per identificare le stesse aree della Cache LLC in entrambe le macchine virtuali. Una volta trovata una procedura per scrivere, leggere o rimuovere dati da un'area fisica della LLC, i due programmi eseguono con la stessa periodicità ma in maniera necessariamente asincrona le seguenti azioni:

- se il programma che invia il messaggio deve inviare un 1, legge/scrive dei dati (casuali) nella LLC un numero concordato di volte in rapida successione
- altrimenti se il programma che invia il messaggio deve inviare uno 0, non scrive nulla nella LLC per lo stesso periodo di tempo
- il programma che riceve il messaggio richiede, con la stessa periodicità, la lettura di uno stesso set di dati che ha caricato in memoria (RAM) e misura il tempo di ogni lettura.

Quando il processo che invia il messaggio invia un 1, cancella dalla LLC i dati del processo che riceve il messaggio, quindi quando il processo che riceve il messaggio legge i suoi dati deve attendere che questi siano ri-caricati dalla RAM nelle Cache: il tempo di lettura sarà quindi alto. Invece quanto il processo che invia il messaggio invia uno 0, l'area della LLC non viene sovrascritta, quindi il processo ricevente trova direttamente in Cache i propri dati e la lettura è molto più veloce. I tempi di lettura del processo ricevente sono quindi alti, per un 1, e bassi, per uno 0.

Ovviamente vi sono molte complicazioni nell'esecuzione di questa semplice procedura, ad esempio i programmi sono interrotti nell'esecuzione dal sistema operativo per l'esecuzione di Interrupt, od altri programmi possono utilizzare la stessa area della LLC.[7] Possono quindi venire introdotti molti errori nel canale di trasmissione. La presenza di rumore ed errori in canali di comunicazione è però un argomento ben noto delle reti di comunicazione e si possono adottare protocolli di correzione degli errori che permettono di rimuovere quasi completamente gli errori e realizzare un canale di comunicazione funzionale, a costo di rallentare la velocità di invio dei dati.

Il secondo tipo di attacco permette di leggere dati riservati utilizzati da un altro programma in esecuzione sullo stesso sistema fisico. Tipicamente si considera come attacco esemplare la lettura della chiave segreta di de/cifratura utilizzata da un altro programma in esecuzione anche su di un'altra macchina virtuale, ma sempre sullo stesso Host.

L'esempio più semplice di questo tipo di attacco è quello di due utenti sulla stessa macchina

(virtuale o fisica), uno che utilizza una chiave segreta per de/cifrare dei dati ed il secondo utente che intercetta la chiave segreta del primo. L'attacco si basa sull'utilizzo di librerie condivise, ovvero entrambi gli utenti utilizzano la stessa libreria crittografica per de/cifrare i dati. Essendo una libreria condivisa, il sistema operativo mantiene in memoria fisica RAM una sola copia della libreria, e mappa la stessa copia fisica nelle due distinte memorie virtuali dei due processi, quello che esegue la de/cifrazione e quello che esegue l'attacco. Supponiamo inoltre che la libreria implementi l'algoritmo RSA utilizzando la procedura matematica chiamata “Square-and-multiply” (questa procedura non è più utilizzata dalle librerie crittografiche attuali).[8] La caratteristica principale di nostro interesse di questa procedura matematica è che è iterativa sui bit della chiave segreta, ripetuta per ogni blocco di dati da de/cifrare, e che esegue due operazioni distinte nel caso in cui un bit della chiave segreta sia 1 o 0. Utilizzando la stessa libreria crittografica, l'attaccante può agire come segue:

- per prima cosa l'attaccante rimuove dalla Cache le istruzioni della libreria crittografica, utilizzando l'istruzione *clflush* o caricandovi altri dati
- poi l'attaccante attende un tempo sufficiente a che il processo vittima esegua un ciclo della procedura “Square-and-multiply”
- infine l'attaccante misura il tempo di lettura (o esecuzione) di una delle due operazioni della libreria.

Se il processo vittima ha utilizzato la stessa operazione dell'attaccante, il tempo misurato è breve perché il processo vittima ha già caricato l'operazione in LLC, altrimenti il tempo misurato è notevolmente maggiore perché l'operazione deve essere caricata dal processo attaccante dalla RAM.

Visto che l'utilizzo delle due operazioni è legato alla presenza di un 1 o di uno 0 nella chiave segreta, è possibile, ripetendo molte volte le misure ed a meno di rumori ed errori come nel caso precedente, individuare i bit della chiave segreta anche con precisioni superiori al 70%.

Contromisure per attacchi alla Cache

Le contromisure contro gli attacchi alla Cache sono in parte simili a quelle adottate per limitare gli attacchi Row Hammer. Con le tecnologie di oggi è difficile immaginare architetture fisiche diverse dall’attuale con la gerarchia di Cache tra la RAM e i Core e quindi risulta molto difficile eliminare il fenomeno fisico alla base di questi attacchi, ovvero la differenza di tempi di accesso alla Cache rispetto alla RAM. Quello che invece è possibile fare è limitare o controllare l’accesso dei processi alle istruzioni che misurano con altissima precisione i tempi di esecuzione di blocchi di codice, e l’accesso ad istruzioni, come *clflush*, che permettono di gestire le Cache od indirizzarne i contenuti. Queste misure possono essere implementate sia in hardware che nei Sistemi Operativi, ad esempio limitando l’esecuzione di alcune istruzioni solo in kernel mode.

Altre misure includono l’eliminazione o forte diminuzione dell’utilizzo della *deduplication*, e una diversa gestione delle librerie comuni (shared libraries) e delle aree comuni delle memorie. Questo impedirebbe ad esempio a diverse macchine virtuali di utilizzare le stesse aree fisiche della Cache, evitando l’attacco Covert Channel qui sopra descritto, ma al contempo rendendo

più complesso e meno efficiente l'utilizzo stesso della Cache.

Infine per quanto riguarda gli algoritmi crittografici, questi devono comunque garantire elevati livelli di sicurezza e devono essere scritti in modo da eseguire il codice in tempo costante indipendentemente dagli ingressi, sia i dati che le chiavi. Le librerie crittografiche più recenti soddisfano questi requisiti, almeno per quanto riguarda gli attacchi noti.

Riferimenti Bibliografici

- [1] National Security Agency (NSA), "TEMPEST: A Signal Problem", 1972, declassificato 2007-09-27, https://www.nsa.gov/news-features/declassified-documents/cryptologic-spectrum/assets/files/tempest.pdf
- [2] si veda ad esempio J.M. Atkinson, "Tempest 101", Granite Island Group, http://www.tscm.com/TSCM101tempest.html
- [3] si veda ad esempio M.G. Kuhn, "Electromagnetic eavesdropping risks of flat-panel displays", International Workshop on Privacy Enhancing Technologies, 88-107, 2004, https://www.cl.cam.ac.uk/~mgk25/pet2004-fpd-slides.pdf
- [4] D. Genkin, M. Pattani, R. Schuster, E. Tromer, "Synesthesia: Detecting Screen Content via Remote Acoustic Side Channels", preprint 2018/08/21, https://www.cs.tau.ac.il/~tromer/synesthesia/
- [5] P.C. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems". CRYPTO 1996
- [6] G. Camurati et al., "Screaming Channels: when electromagnetic side channels meet radio transceivers", preprint 2018, http://s3.eurecom.fr/tools/screaming_channels/
- [7] Y. Kim, R. Daly, J. Kim, C. Fallin, J.H. Lee, D. Lee, C. Wilkerson, K. Lai, O. Mutlu (June 24, 2014). "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors" 2014/06/24, IEEE, https://users.ece.cmu.edu/~yoonguk/papers/kim-isca14.pdf
- [8] Rowhammer.js: https://arxiv.org/abs/1507.06955
  Nethammer: https://arxiv.org/abs/1805.04956
  Another Flip: https://arxiv.org/abs/1710.00551
  Flip Feng Shui: https://www.usenix.org/system/files/conference/usenixsecurity16/sec16_paper_razavi.pdf
  Glitch: https://www.vusec.net/projects/glitch/
  RAMpage: https://vvdveen.com/publications/dimva2018.pdf
- [9] Prime+Probe: http://palms.ee.princeton.edu/system/files/SP_vfinal.pdf
  Flush+Reload: https://eprint.iacr.org/2013/448.pdf
  Flush+Flush: https://gruss.cc/files/flushflush.pdf
  Cache attack in Javascript: https://arxiv.org/pdf/1502.07373.pdf
- [10] C. Maurice et al., "Hello from the Other Side: SSH over. Robust Cache Covert Channels in the Cloud", https://gruss.cc/files/hello.pdf

Note

- [1] Oggi il termine corretto è EMSEC, ovvero “Emissions Security”, anche se TEMPEST è ancora utilizzato come termine generico.
- [2] Sin dagli anni '70, ad esempio già con la DRAM Intel 1130.
- [3] ECC DRAM furono introdotte principalmente per correggere errori indotti da radiazioni elettromagnetiche esterne e radiazioni di background quali particelle alpha, neutroni ecc. ad esempio emessi dal sole, che possono interferire con una cella e cambiarne il valore.
- [4] Come indicato, la lettura di un bit induce la lettura / annullamento / riscrittura di una intera riga di memoria.
- [5] Si ricordi la breve descrizione della gestione della memoria riportata nella prima parte di questo articolo, con la presenza di memoria fisica e virtuale, pagine, segmenti ecc.
- [6] L'identificazione di una stessa area della Cache può essere semplice nel caso in cui l’Hypervisor utilizzi la “*deduplication*” (mappa alla stessa memoria fisica di blocchi di memoria uguali), misura utile a ridurre l’utilizzo della memoria.
- [7] La gestione della LLC è comunque in carico al Sistema Operativo, in questo caso Host, ed i processi non possono utilizzare aree della LLC in maniera esclusiva.
- [8] Un simile attacco è possibile quando l'algoritmo AES è implementato tramite “T-tables”, anche questa implementazione non è più utilizzata dalle librerie crittografiche attuali.

Articolo a cura di **Andrea Pasquinucci**