RIDL : Rogue In-Flight Data Load

Analyse de la classe de vulnérabilités MDS

Baptiste Rébillard, Leandro Rodriguez, Aurélien Pouilles, Killian Marty

INSA Toulouse

16 decembre 2025

Description de la faiblesse (MDS)

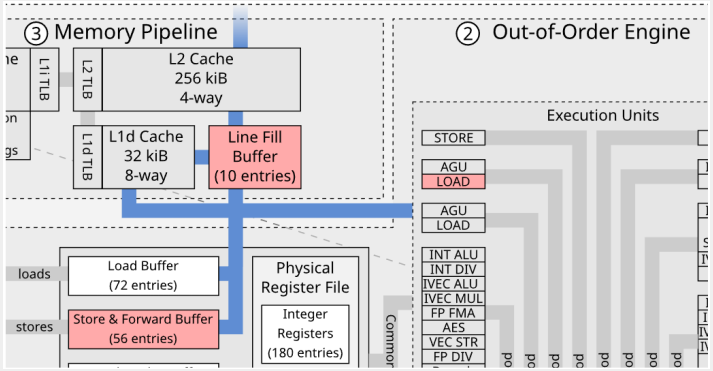


Figure: Buffers partagés[1]

RIDL appartient à la classe *Microarchitectural Data Sampling* (MDS).

1. **Exécution Spéculative :** Pour optimiser les performances, le CPU exécute des instructions à l'avance (spéculation).

[1] from: https://mdsattacks.com/slides/slides.html

Description de la faiblesse (MDS)

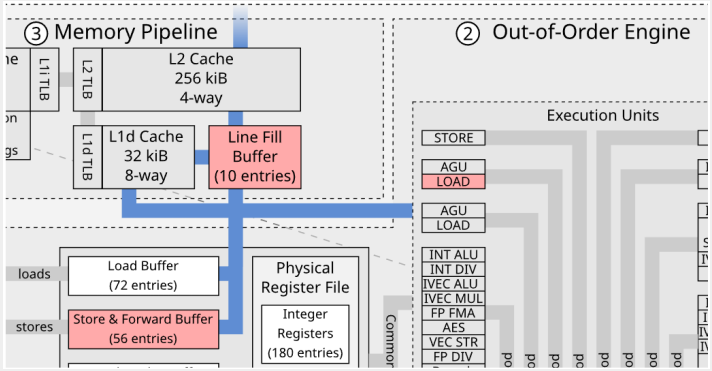


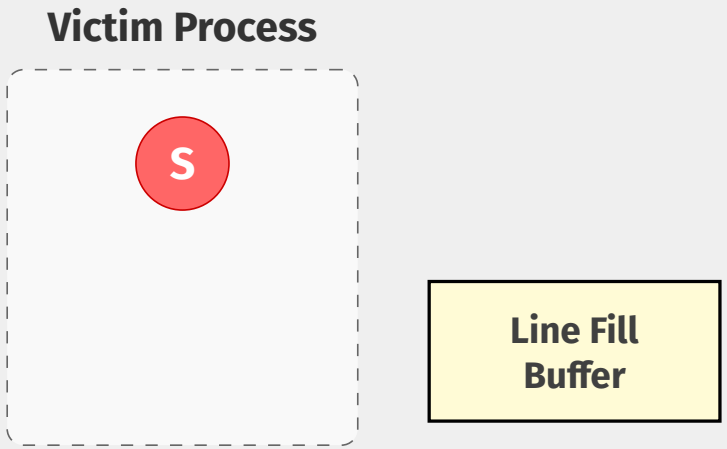
Figure: Buffers partagés[1]

RIDL appartient à la classe *Microarchitectural Data Sampling* (MDS).

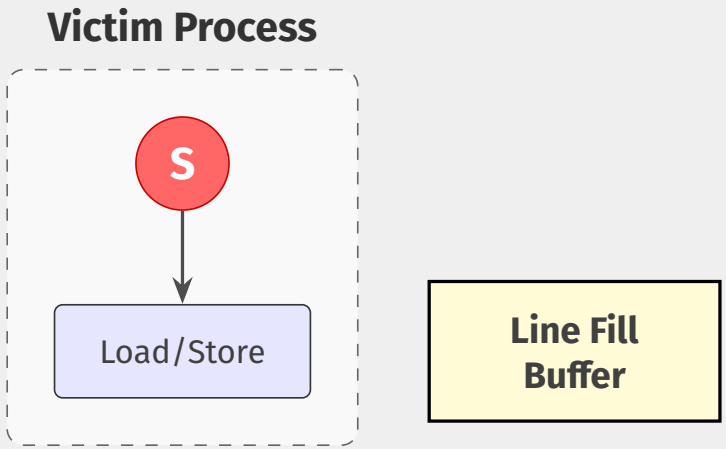
1. **Exécution Spéculative :** Pour optimiser les performances, le CPU exécute des instructions à l'avance (spéculation).
2. **Données "In-Flight" :** Lorsqu'une lecture spéculative est effectuée, si la donnée n'est pas prête, le CPU peut "attraper" des données traînant dans les Line Fill Buffers.

[1] from: https://mdsattacks.com/slides/slides.html

Anatomie de l'attaque RIDL (Flux complet)

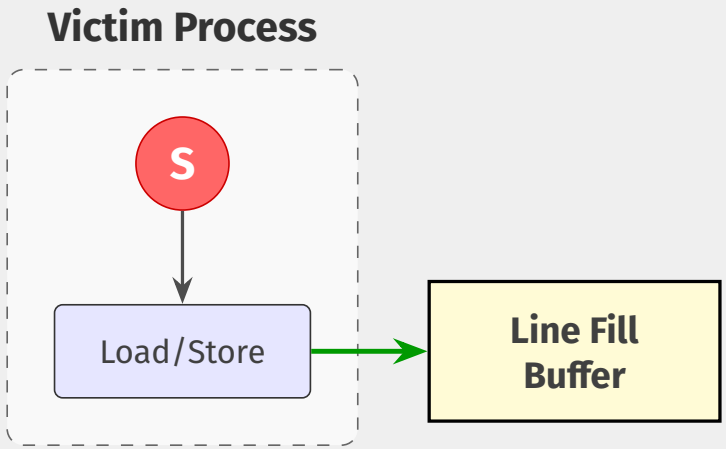


Anatomie de l'attaque RIDL (Flux complet)



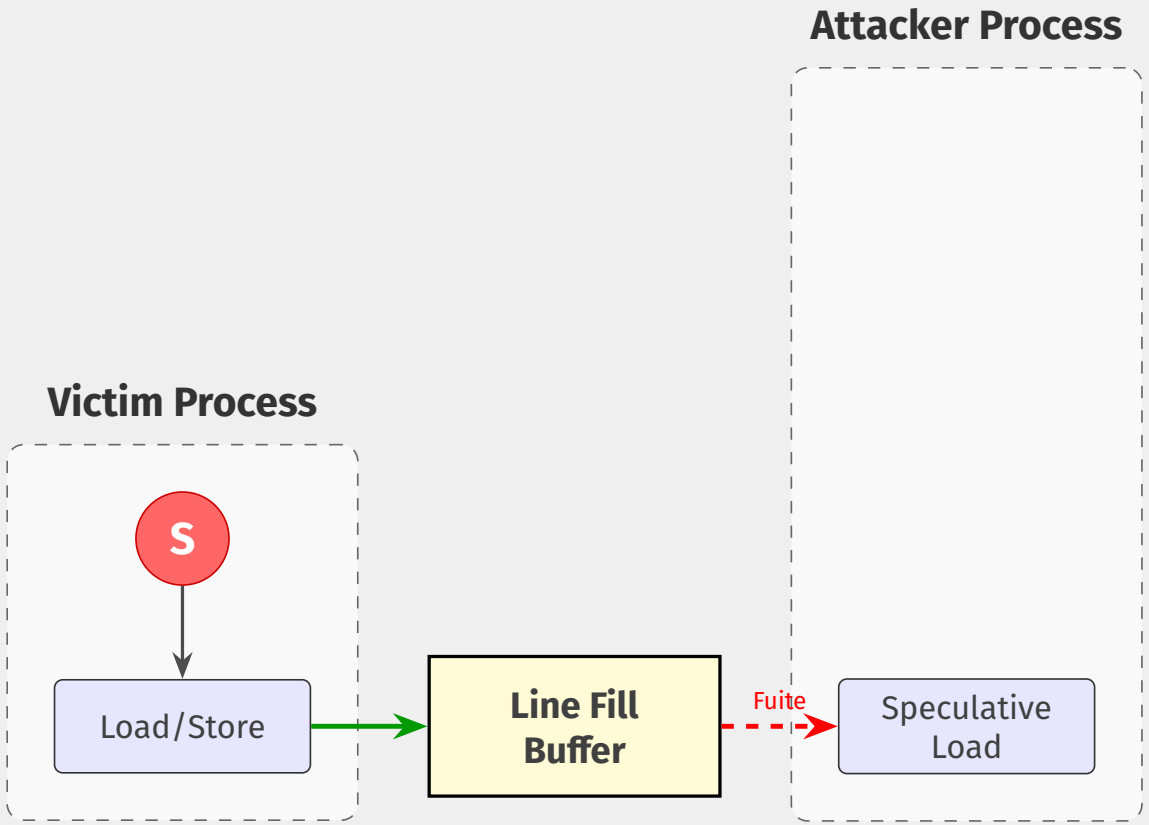
- La victime charge des données en mémoire.

Anatomie de l'attaque RIDL (Flux complet)



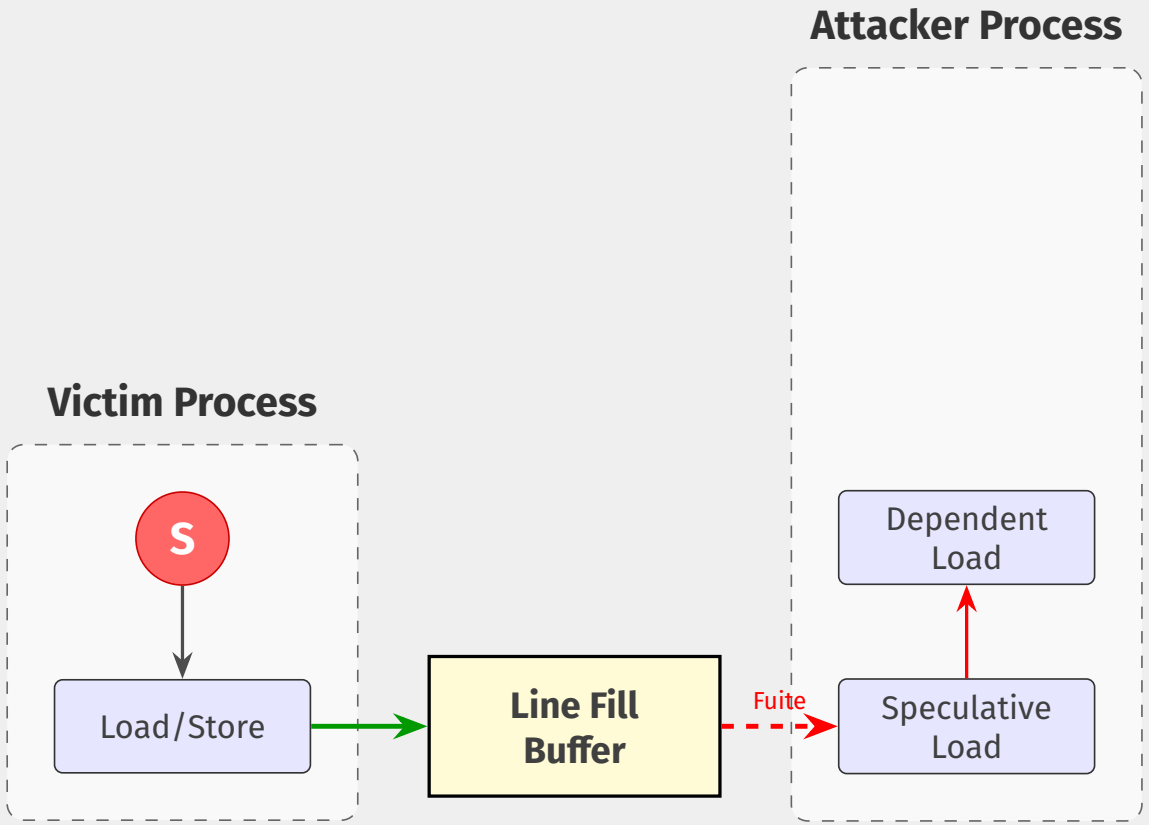
- La victime charge des données en mémoire.
- Ces données transitent par les buffers partagés (optimisation).

Anatomie de l'attaque RIDL (Flux complet)



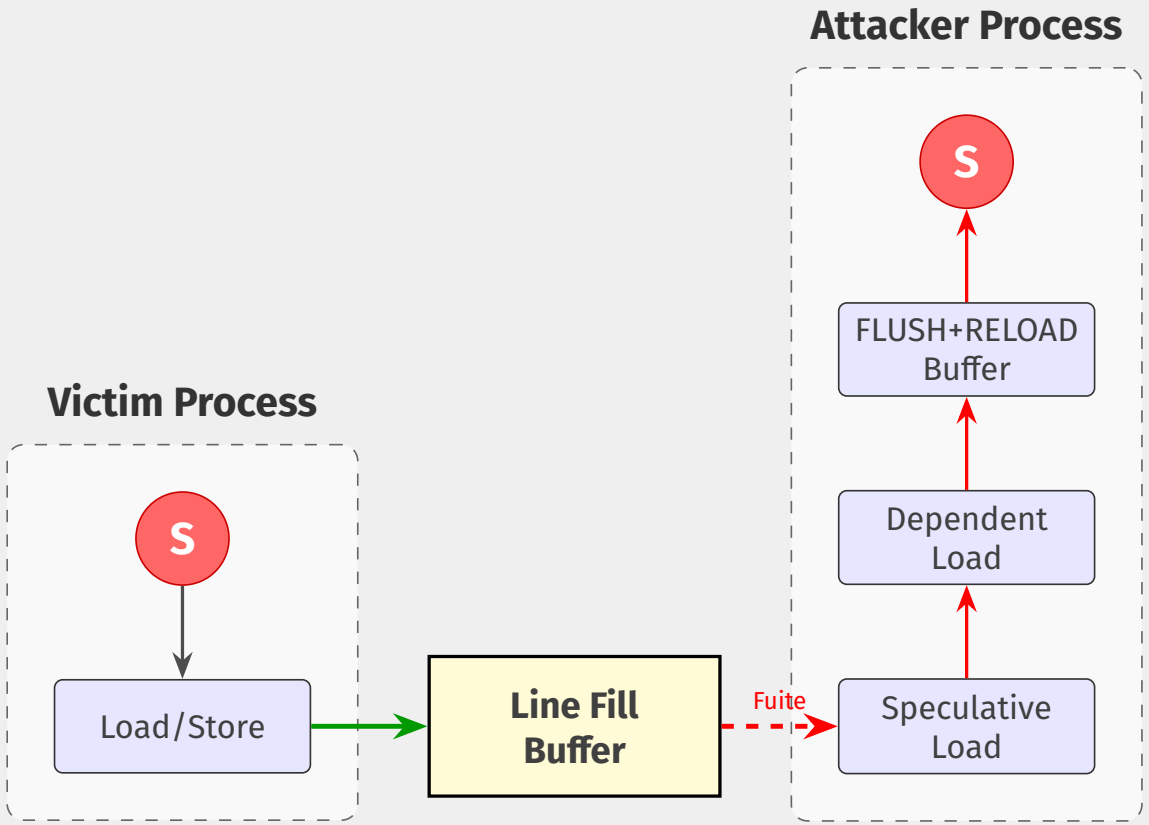
- La victime charge des données en mémoire.
- Ces données transitent par les buffers partagés (optimisation).
- L'attaquant capte la donnée par une lecture spéculative.

Anatomie de l'attaque RIDL (Flux complet)



- La victime charge des données en mémoire.
- Ces données transitent par les buffers partagés (optimisation).
- L'attaquant capte la donnée par une lecture spéculative.
- Un **Load Dépendant** encode cette valeur volatile dans le Cache.

Anatomie de l'attaque RIDL (Flux complet)



- La victime charge des données en mémoire.
- Ces données transitent par les buffers partagés (optimisation).
- L'attaquant capte la donnée par une lecture spéculative.
- Un **Load Dépendant** encode cette valeur volatile dans le Cache.
- Après rollback, l'attaquant retrouve la valeur par F&R.

Comment obtenir précisément ce qu'on veut ?

On leak de l'information mais on ignore ce que c'est… Puisque le LFB contient du bruit, l'attaquant doit **stimuler** la victime puis **filtrer** le résultat.

- **Stimulation par Appels Système (Syscalls) :**
  L'attaquant force le noyau à manipuler le secret via des fonctions standard (ex: `passwd` pour charger `/etc/shadow`, ou `ssh` pour les clés).

Comment obtenir précisément ce qu'on veut ?

On leak de l'information mais on ignore ce que c'est… Puisque le LFB contient du bruit, l'attaquant doit **stimuler** la victime puis **filtrer** le résultat.

- **Stimulation par Appels Système (Syscalls) :**
  L'attaquant force le noyau à manipuler le secret via des fonctions standard (ex: `passwd` pour charger `/etc/shadow`, ou `ssh` pour les clés).
- **Entraînement du Prédicteur (Style Spectre) :**
  L'attaquant entraîne le prédicteur de branchement pour forcer le CPU à charger spéculativement l'adresse cible dans le LFB (ex: via `copy_from_user`).

Comment obtenir précisément ce qu'on veut ?

On leak de l'information mais on ignore ce que c'est... Puisque le LFB contient du bruit, l'attaquant doit **stimuler** la victime puis **filtrer** le résultat.

- **Stimulation par Appels Système (Syscalls) :**
  L'attaquant force le noyau à manipuler le secret via des fonctions standard (ex: `passwd` pour charger `/etc/shadow`, ou `ssh` pour les clés).
- **Entraînement du Prédicteur (Style Spectre) :**
  L'attaquant entraîne le prédicteur de branchement pour forcer le CPU à charger spéculativement l'adresse cible dans le LFB (ex: via `copy_from_user`).
- **Répétition Statistique :**
  L'attaque est répétée des milliers de fois et statistiquement on retire le bruit de fond.

Comment obtenir précisément ce qu'on veut ?

On leak de l'information mais on ignore ce que c'est... Puisque le LFB contient du bruit, l'attaquant doit **stimuler** la victime puis **filtrer** le résultat.

- **Stimulation par Appels Système (Syscalls) :**
  L'attaquant force le noyau à manipuler le secret via des fonctions standard (ex: `passwd` pour charger `/etc/shadow`, ou `ssh` pour les clés).
- **Entraînement du Prédicteur (Style Spectre) :**
  L'attaquant entraîne le prédicteur de branchement pour forcer le CPU à charger spéculativement l'adresse cible dans le LFB (ex: via `copy_from_user`).
- **Répétition Statistique :**
  L'attaque est répétée des milliers de fois et statistiquement on retire le bruit de fond.
- **Filtrage "Mask-Subtract-Rotate" :**
  Injection de calculs arithmétiques dans la spéculation pour aligner les données connues (ex: `/etc/shadow` commence par le préfixe "root:") et éliminer le bruit avant la mesure.

Une Surface d'Attaque Étendue

Possibles attaques :

- **Cross-Process & Kernel :** Un processus utilisateur non privilégié peut lire la mémoire du noyau (ex: `/etc/shadow`) ou d'autres processus.

Une Surface d'Attaque Étendue

Possibles attaques :

- **Cross-Process & Kernel :** Un processus utilisateur non privilégié peut lire la mémoire du noyau (ex: `/etc/shadow`) ou d'autres processus.
- **Cross-VM (Cloud) :** Fuite de données entre machines virtuelles co-localisées sur le même cœur physique (testé sur KVM et Hyper-V).

Une Surface d'Attaque Étendue

Possibles attaques :

- **Cross-Process & Kernel :** Un processus utilisateur non privilégié peut lire la mémoire du noyau (ex: `/etc/shadow`) ou d'autres processus.
- **Cross-VM (Cloud) :** Fuite de données entre machines virtuelles co-localisées sur le même cœur physique (testé sur KVM et Hyper-V).
- **JavaScript / Navigateur :** Exploitation possible depuis une page web (via WebAssembly) pour fuiter des données du navigateur ou du système.

Une Surface d'Attaque Étendue

Possibles attaques :

- **Cross-Process & Kernel :** Un processus utilisateur non privilégié peut lire la mémoire du noyau (ex: `/etc/shadow`) ou d'autres processus.
- **Cross-VM (Cloud) :** Fuite de données entre machines virtuelles co-localisées sur le même cœur physique (testé sur KVM et Hyper-V).
- **JavaScript / Navigateur :** Exploitation possible depuis une page web (via WebAssembly) pour fuiter des données du navigateur ou du système.
- **Page Table Disclosure :** Fuite des adresses physiques utilisées par le MMU, permettant de contourner les protections ASLR.

Mitigations Recommandées

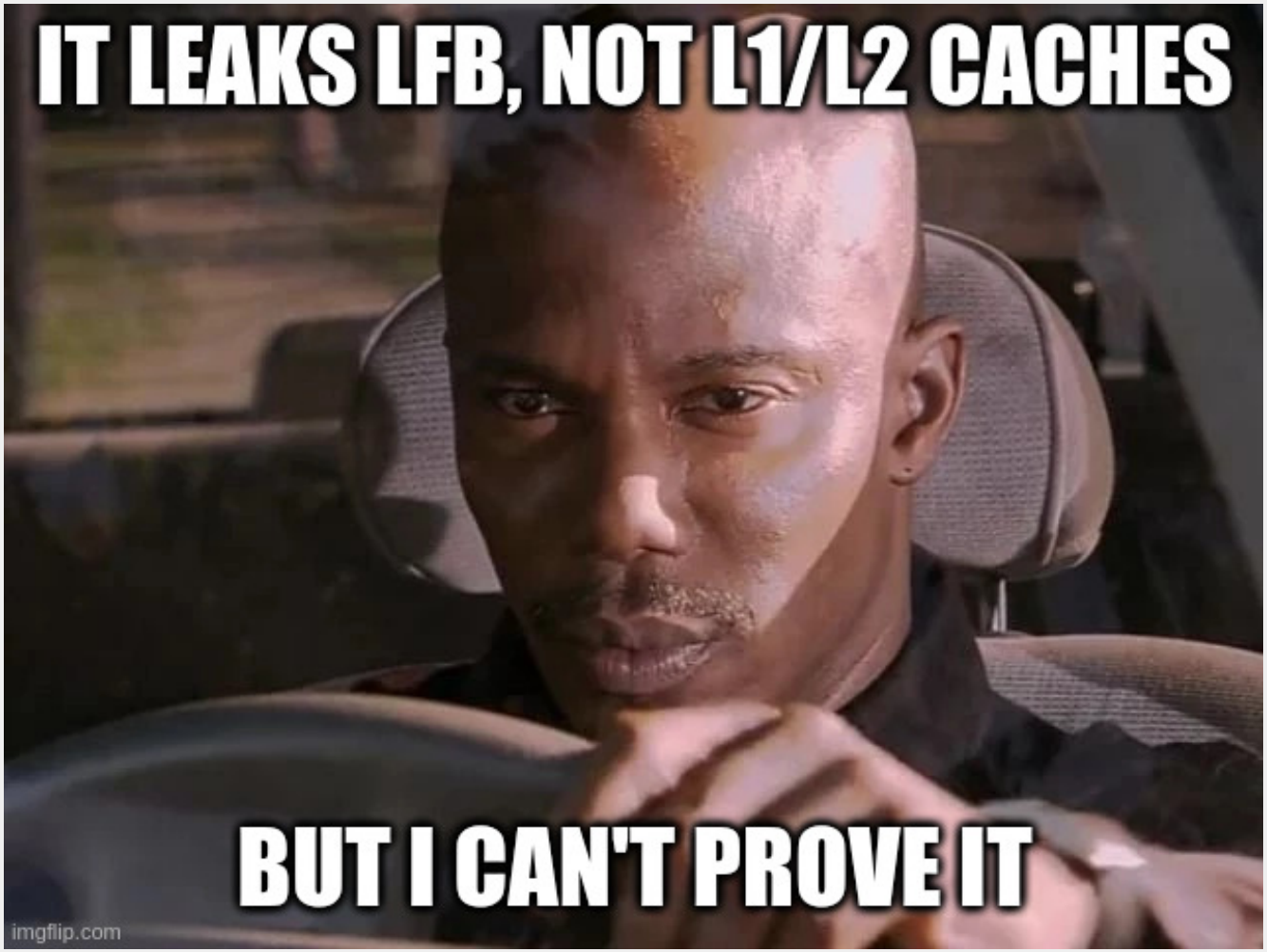
- **Désactiver SMT (Hyper-Threading)** : C'est la seule manière fiable d'empêcher le partage des buffers entre threads.
- **Flush des Buffers (Instruction VERW)** : Le CPU doit vider ses buffers internes à chaque changement de contexte (coûteux en performances).

Démo

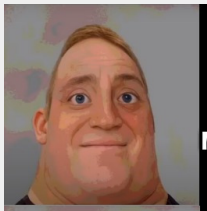

On va faire une approche speculative
On va stocker la donnée dans le LFB comme ça l'attaquant ne peut pas leak le cache
L'attaquant leak le LFB
L'attaquant leak le LFB
imgflip.com

References

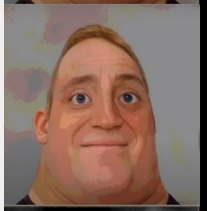
Stephan Van Schaik, Alyssa Milburn, Sebastian Osterlund, Pietro Frigo, Giorgi Maisuradze, Kaveh Razavi, Herbert Bos, and Cristiano Giuffrida.
RIDL: Rogue in-flight data load.
In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 88–105. IEEE, 2019.


IT LEAKS LFB, NOT L1/L2 CACHES
BUT I CAN'T PROVE IT
imgflip.com


Me leaking
the latest
book from
Nicolas Sarkozy
Intel
optimising by
putting data in LFB
imgflip.com



LA VICTIME CHARGE UNE DONNÉE DE MANIÈRE SPECULATIVE



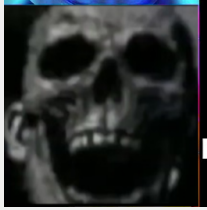
LA DONNÉE PASSE PAR LES LFB

UN ATTAQUANT LEAK LES LFB

LE PROCESSEUR FLUSH LES LFB

L'ATTAQUANT AVAIT FAIT UN DEPENDENT LOAD ET LA DONNÉE EST DANS LES CACHES

LE PROCESSEUR FLUSH LES CACHES



L'ATTAQUANT FAIT UNE ATTAQUE FLUSH AND RELOAD


CACHE
LEAK
OPTIMISATION
TO STORE
DATA LFB TO
AVOID CACHE LEAK
LFB
LEAK
imgflip.com