# Flush and reload

Research Project

Lomig Piette
lomig.piette@etudiant.univ-rennes1.fr

8 février 2023

---

## Table des matières

# 1 Principe du cache et sa faille

## 1.1 Fonctionnement

Durant l'execution d'un processus, ce dernier a parfois besoin d'accéder à des données stockées en mémoire, comme des secrets, ou bien même des fonctions. Le problème est que cet accès en mémoire est très couteux et assez long. Cela ralentit considérablement le programme.

Pour pallier ce problème, on mit au jour une petite mémoire directement intégré dans le processeur que l'on appelle le cache. Le cache permet de stocker aussi des données, pour qu'un processus puisse y accéder rapidement et donc améliorer la rapidité d'exécution.

Son fonctionnement est assez simple dans l'ensemble. Quand un programme cherche à appeler une ligne mémoire, le processeur va stocker ces informations dans le cache. Ainsi, si le programme a besoin de nouveau y accéder, ces données auront été mis de cotés, et l'accès sera rapide.

Cependant, il est possible que le cache soit parfois inconsistant, c'est à dire que la mémoire qu'il contient est inutile puisque non réutilisé par nos processus actifs. Il est donc possible de vider (flush) le cache d'une ligne mémoire afin de libérer de l'espace pour un prochain chargement.

## 1.2 Faille

Les OS ont normalement le devoir de séparer les processus et éviter que ces derniers puissent s'influencer les uns des autres. Le problème est que le cache est partagé par plusieurs processus. Dans notre cas, voici le principe de la faille :

Si le cache permet de garder en mémoire des données appelées par un programme, cela veut dire que nous pouvons vérifier si ces données sont dans le cache ou non, en mesurant le temps d'accès.

Or, si nous pouvons vider le cache d'une mémoire, nous pourrions vider le cache d'une fonction, et vérifier ensuite si elle a été chargée par un autre processus.

# 2 Flush and reload

Flush and reload se base sur cette faille, afin de pouvoir deviner les appels de fonctions effectués par un processus tiers qu'un pirate voudrait attaquer.

1. Tout d'abord, on va viser une fonction que l'on veut tracer. Pour ça, on va d'abord vidé la mémoire cache de cette fonction.
2. Ensuite, on attend durant un temps arbitraire.
3. Après un certains temps, nous rappelons la fonction en mesurant le temps d'accès.
4. Si le temps est rapide, alors la fonction a étée rechargée et donc appelée par le processus tiers. Si non, alors la fonction n'a pas été appelée depuis.
5. On recommence autant que l'on veut pour connaitre la séquence d'appels de cette fonction.

De cette manière, nous pouvons connaître toute la séquence d'appels d'une fonction et déterminé un certains fonctionnement du programme visé.

# 3 Cas pratique sur RSA

Pour un chiffrement RSA, l'algorithme utilise des opérations sur de grands nombre afin d'obtenir une décomposition en facteur premier, etc. Cela veut dire que le programme utilise des fonctions pour ses operations modulaires, de multiplications, de puissance.

Or, si nous pouvons déterminé la séquence de ces operations, nous pouvons remonter jusqu'à la clé et ainsi lire les secrets.

Pour cela, on utilise flush and reload sur chacune de ses operations dans le but de les tracer et d'obtenir la séquence des appels.

Il faut prendre en compte que lors d'un flush and reload, il est tout à fait possible et même probable, que l'on reload trop tôt, c'est à dire avant que le processus visé n'ait eu le temps de faire son appel. Ainsi, on nous pourra savoir quand réellement la fonction a été rappelée.