# TP6 - ASR2 - ENS Lyon - L3IF - 2019-2020
# Secure heap memory manager

**Intervenants:**

**Stella Bitchebe (celestine-stella.ndonga-bitchebe@ens-lyon.fr)**
**Lavoisier Wapet (patrick-lavoisier.wapet@ens-lyon.fr)**
**Alain Tchana (alain.tchana@ens-lyon.fr)**
(Note: ce sujet a été rédigé par Stella et Lavoisier)

## Objectif général

L'objectif de ce TP est d'enrichir votre heap manager (HM) réalisé lors du TP précédent. Faire le TP en deux étapes:

1. Lorsqu'un programme demande de la mémoire en passant en paramètre une taille supérieure à une certaine limite (que vous appellerez MMAP_THRESHOLD), utiliser plutôt la fonction mmap pour allouer de la mémoire auprès de l'OS.
2. Implanter une protection au buffer overflow en développant une version protégée de malloc que nous appellerons pmalloc.

## Description

**mmap:** Lors de l'utilisation du la fonction mmap, penser à activer le bit MAP_ANONYMOUS de la bitmap faisant office du paramètre flag. Penser aussi à donner la valeur -1 au descripteur de fichier (paramètre fd). Référez vous aux explications séance tenante et à la documentation du mmap pour plus d'informations.

**buffer overflow:** Actuellement, un programme peut aller lire au delà d'un buffer obtenu par malloc sans causer d'erreur. Donc, si un programme utilisant votre HM subit une attaque de type buffer overflow, ce dernier ne pourra pas protéger les buffers critiques (les méta-données ou les autres blocs mémoires par exemple).

Pour protéger des buffers critiques, nous allons utiliser l'approche "page guard". Elle consiste à protéger chaque buffer par une page mémoire de garde qui est soit unmapped, soit PROT_NONE avec mprotect. AInsi, Lorsqu'il y'aura un buffer overflow, cela se fera sur la "page guard", ce qui va causer un segfault. Implanter dans un premier temps l'approche en utilisant mprotect, puis unmapped en utilisant la fonction **munmap**(2) sur la page de garde.

**NB :** Bien sûr, lorsque le buffer est freed, il faut rendre les pages guards réutilisables.

Implanter aussi un handler pour le segfault (SIGSEGV) afin de détecter une tentative de buffer overflow et de l'afficher à l'écran.

Enfin, faire un programme de test (on pourra aussi enrichir le programme du TP 5 en implantant essayant de violer la protection "page guard").

Quelques indications:

- man mmap, munmap, mprotect
- int pagesize=sysconf(_SC_PAGE_SIZE); Donne la taille d'une page mémoire
- Etant donné que le buffer à protéger doit se faire par une page de garde, il faut toujours allouer le buffer à la fin d'une page, pas au milieu de la page.