Vulnérabilité mémoire

ESIR

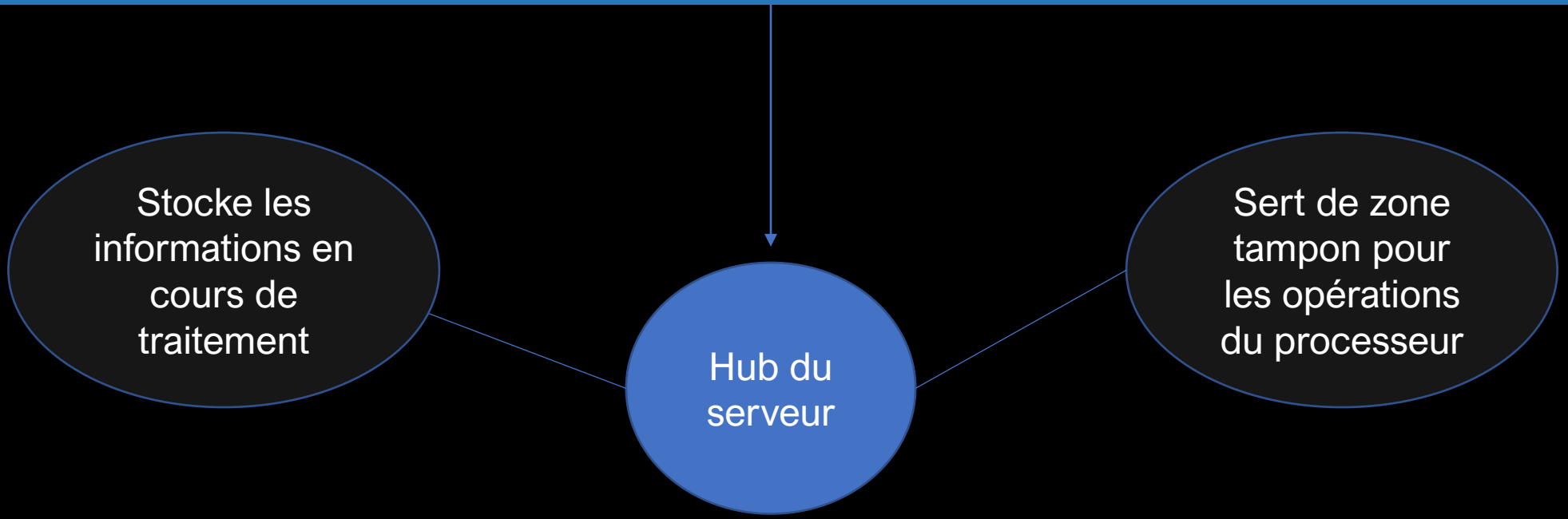
Djob Mvondo

Pourquoi la mémoire ?

D'après vous, pourquoi il est important de protéger la mémoire ?

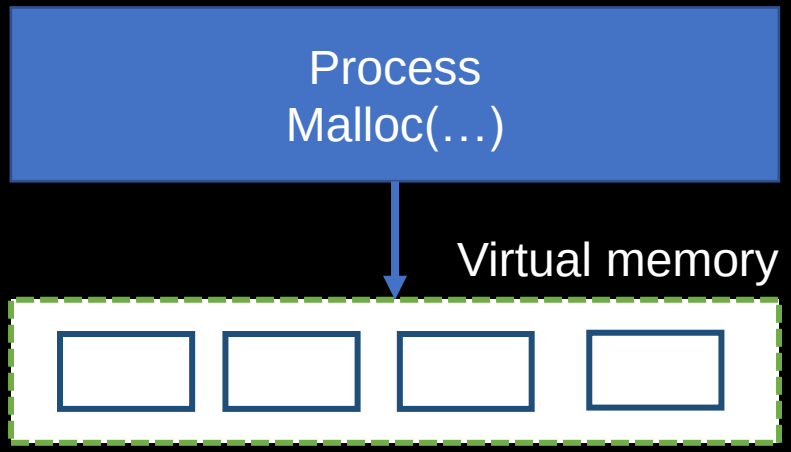
Pourquoi la mémoire ?

D'après vous, pourquoi il est important de protéger la mémoire ?



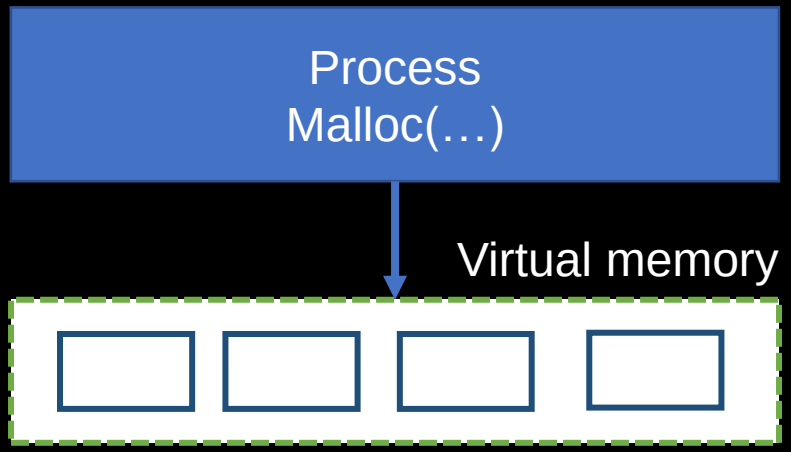
Principe d'allocation mémoire

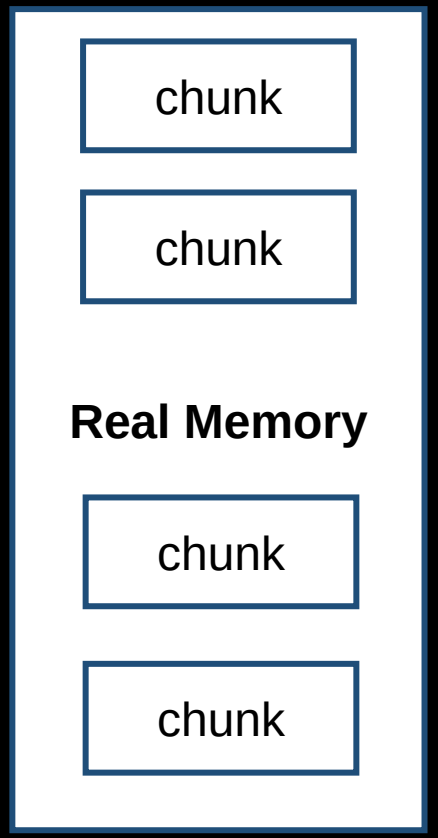
Pour comprendre les problèmes liés à la mémoire, il faut cerner le mécanisme d'allocation de la mémoire à des processus



Principe d'allocation mémoire

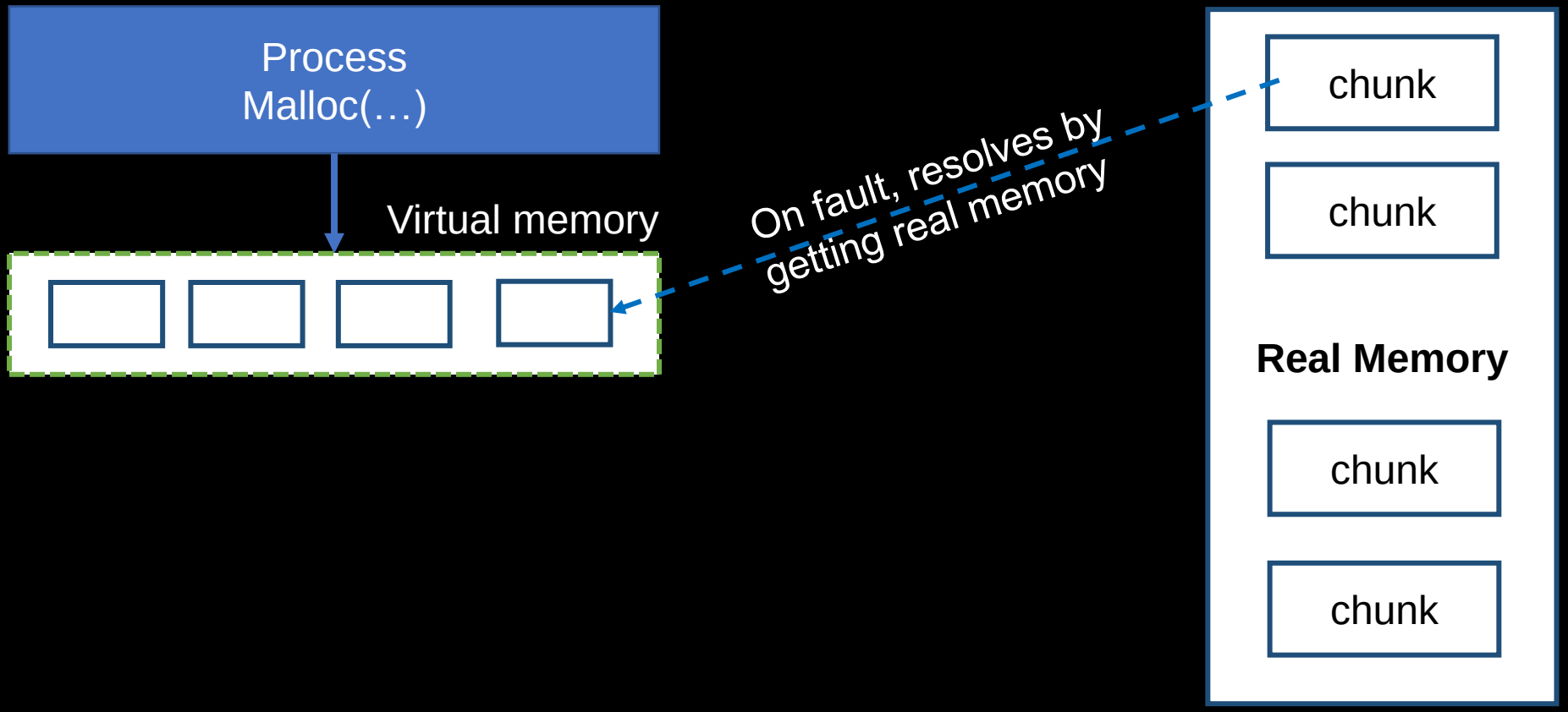
Pour comprendre les problèmes liés à la mémoire, il faut cerner le mécanisme d'allocation de la mémoire à des processus





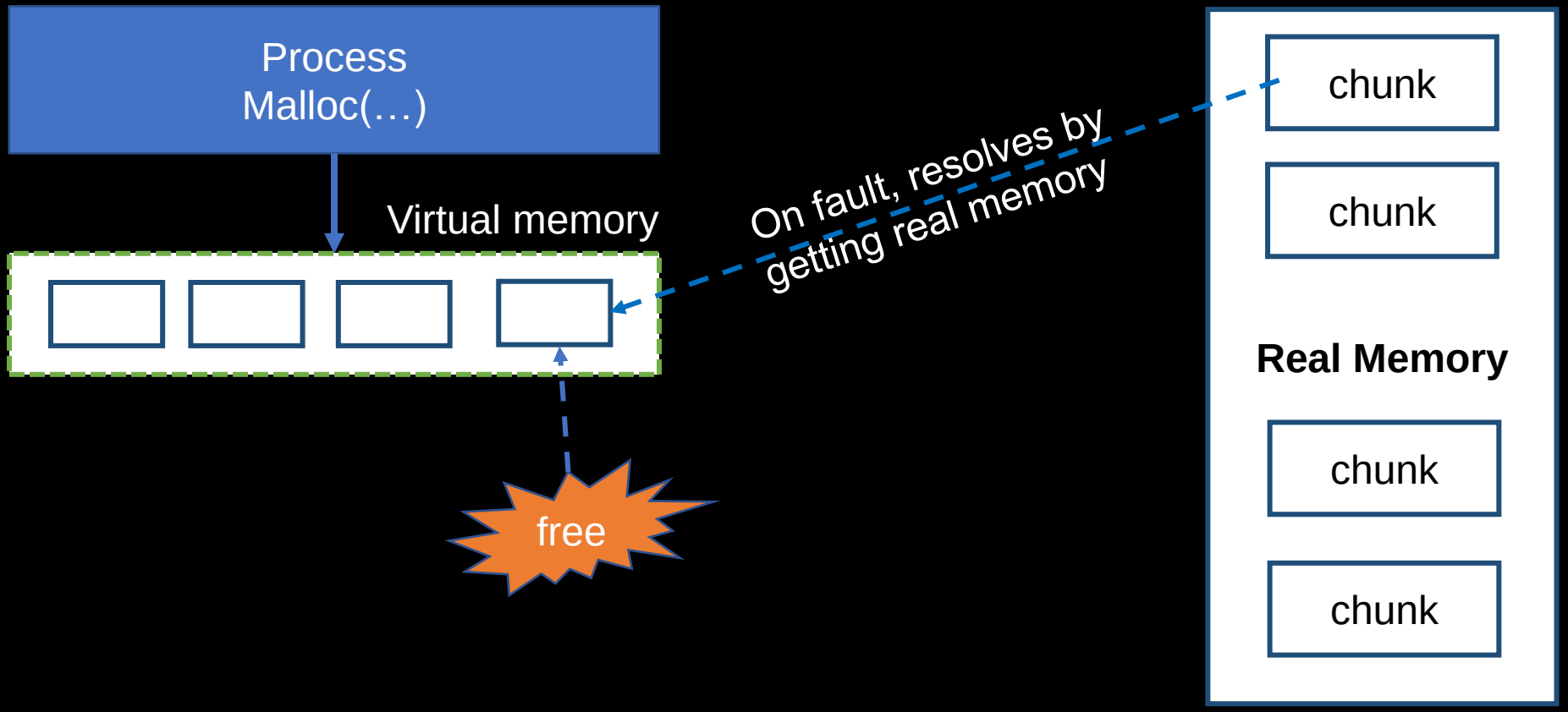
Principe d'allocation mémoire

Pour comprendre les problèmes liés à la mémoire, il faut cerner le mécanisme d'allocation de la mémoire à des processus



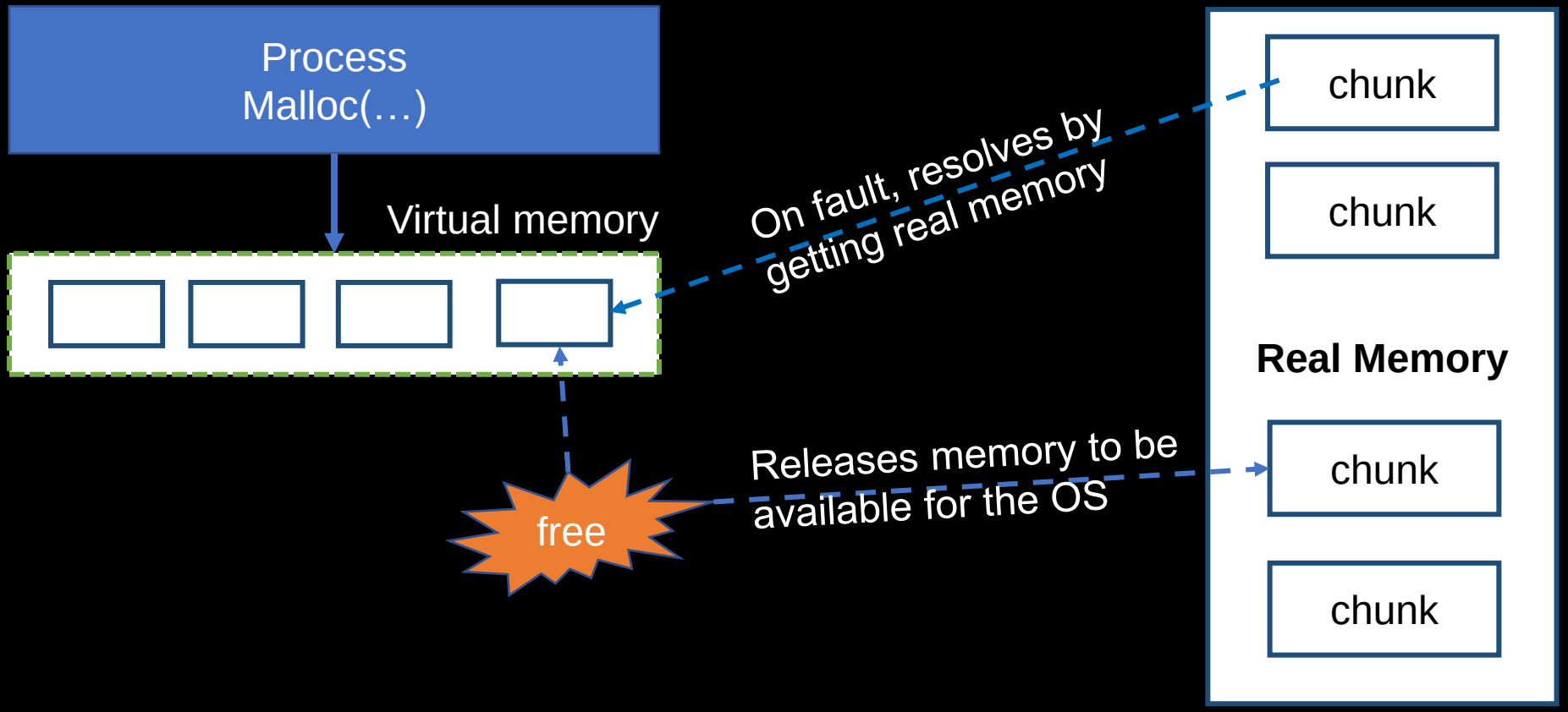
Principe d'allocation mémoire

Pour comprendre les problèmes liés à la mémoire, il faut cerner le mécanisme d'allocation de la mémoire à des processus



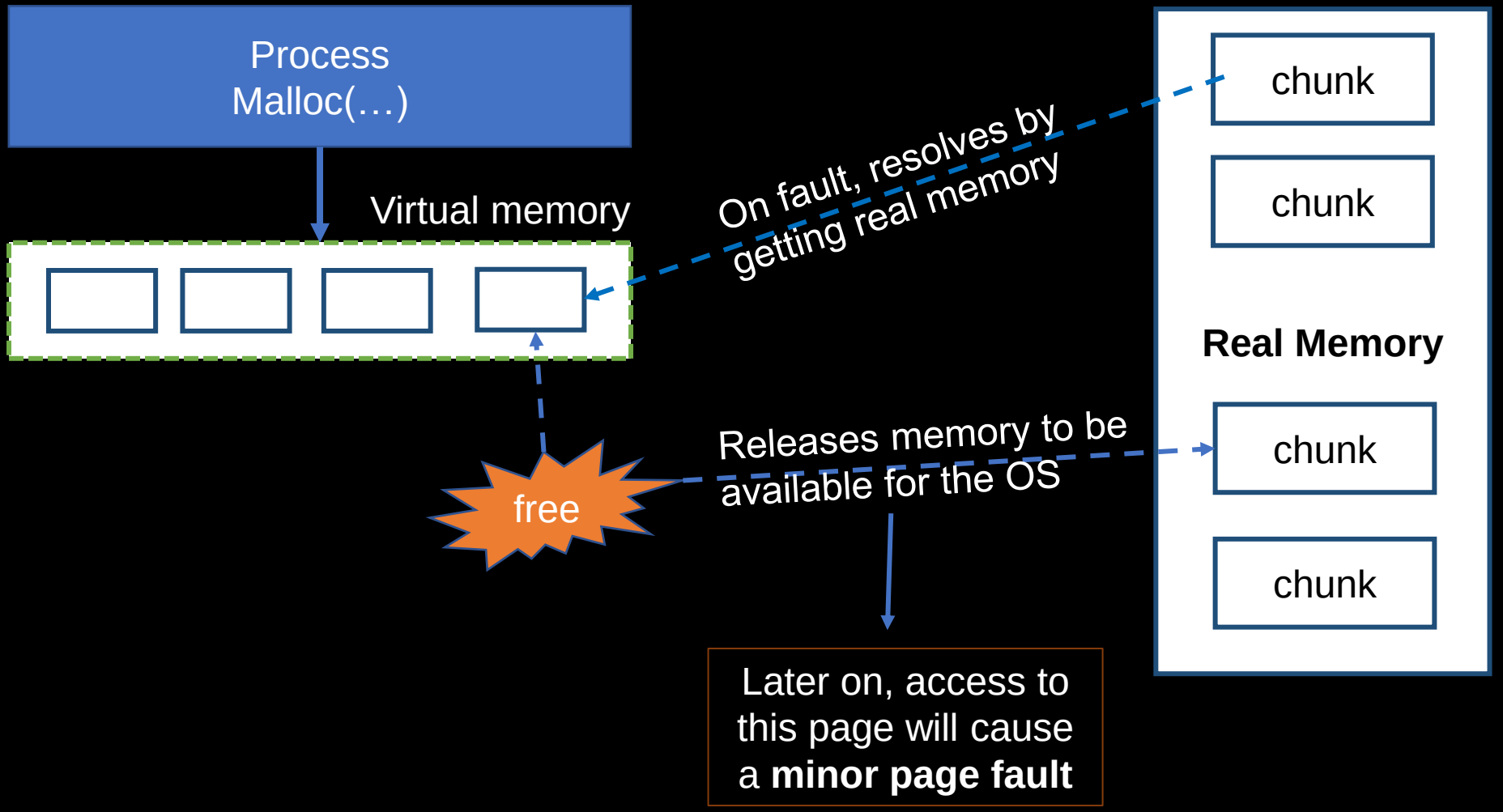
Principe d'allocation mémoire

Pour comprendre les problèmes liés à la mémoire, il faut cerner le mécanisme d'allocation de la mémoire à des processus



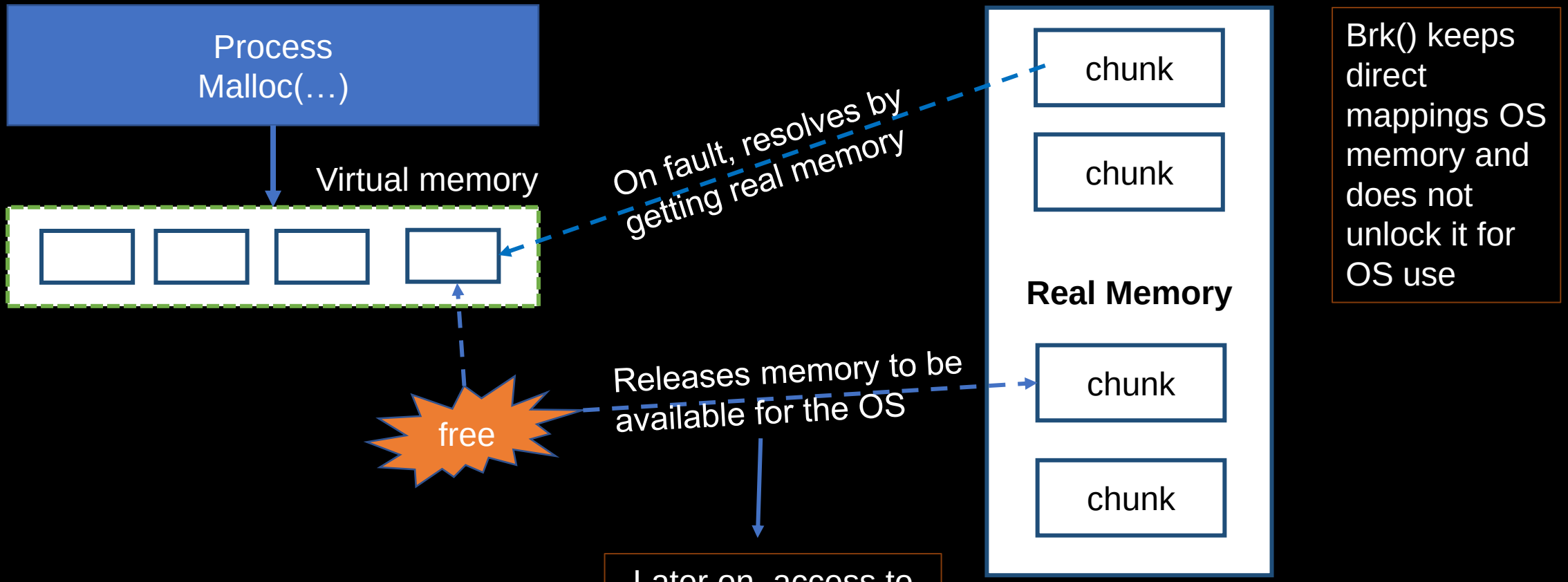
Principe d'allocation mémoire

Pour comprendre les problèmes liés à la mémoire, il faut cerner le mécanisme d'allocation de la mémoire à des processus



Principe d'allocation mémoire

Pour comprendre les problèmes liés à la mémoire, il faut cerner le mécanisme d'allocation de la mémoire à des processus



Principe d'allocation mémoire: **le tas**

L'espace d'adressage d'un processus est référencé par le tas (**heap)**, qui contient les zones mémoire stockant le code, données, and des sections des librairies tiers.

Each block has specific permissions

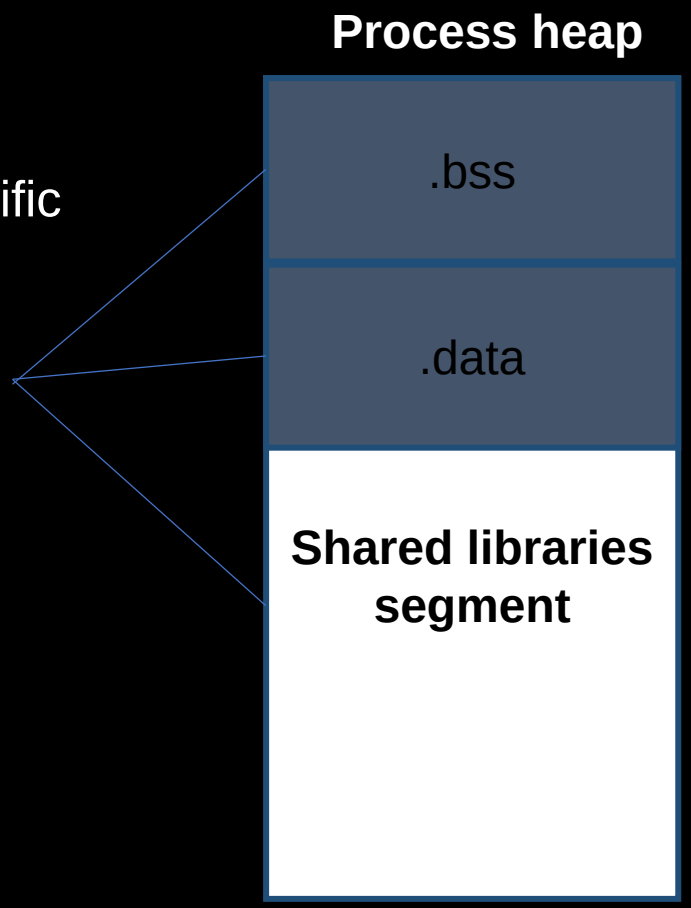
R: read

W: write

X: execution

P: protected

S: shared



Using informations in the **private heap** of a process is the **prefered attack** mode when dealing with memory issues

Espace d'adressage d'un programme

Visualisons l'espace d'adressage d'un processus

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>

int main(){

  char *addr;
  printf("Welcome to this course ::%d\n",getpid());
  printf("Enter a sentence\n");
  getchar();
  addr=(char *)malloc(1000);
  free(addr);
  printf("Finished\n");

  return 0;
}
```

demo.c

gcc -odemo.o demo.c

strace ./demo.o

Qu'observez-vous ?

Espace d'adressage d'un programme

Visualisons l'espace d'adressage d'un processus

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>

int main(){

  char *addr;
  printf("Welcome to this course ::%d\n",getpid());
  printf("Enter a sentence\n");
  getchar();
  addr=(char *)malloc(1000);
  free(addr);
  printf("Finished\n");

  return 0;
}
```

demo.c

Qu'observez-vous ?

mmap()
mprotect()

Quel est leur rôle ?

Devoir 4

Faire des recherches sur **mmap()** et **mprotect()** et détailler leur fonctionnement avec des exemples d'utilisation (code à l'appui).

Votre rapport ne doit pas faire plus de **10 pages.**

Espace d'adressage d'un programme

Plus d'information sur le tas (heap) d'un processus à travers le fichier:

/proc/<pid>/maps

```
7faa724da000-7faa72652000 r-xp 00025000 08:30 11971                      /usr/lib/x86_64-
linux-gnu/libc-2.31.so
7faa72652000-7faa7269c000 r--p 0019d000 08:30 11971                      /usr/lib/x86_64-
linux-gnu/libc-2.31.so
7faa7269c000-7faa7269d000 ---p 001e7000 08:30 11971                      /usr/lib/x86_64-
linux-gnu/libc-2.31.so
```

Donne les informations sur les zones mémoire utilisés, correspondances, droits, et le type du composant où elles sont hébergés.

Chaque ligne un format:

<address start>-<address end> <mode> <offset> <major id:minor id> <inode id> <file path>

Memory issues: Corruption - access

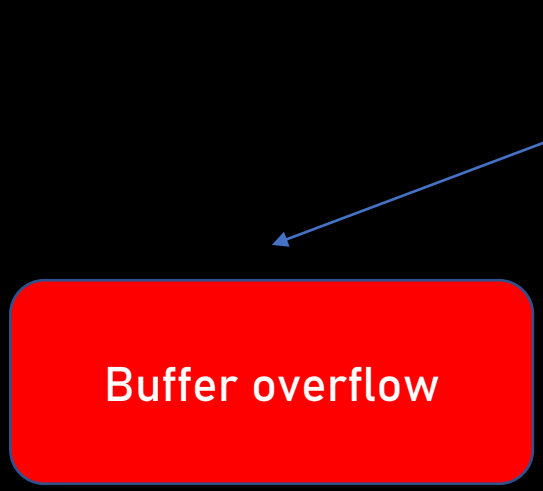
Trouver un moyen de regarder les zones de votre programme C et ceci: **cat /proc/self/maps**

Comment cette information peut être utilisé de façon malicieuse ?

Memory issues: Corruption - access

Check the mappings for your C program and also **cat /proc/self/maps**

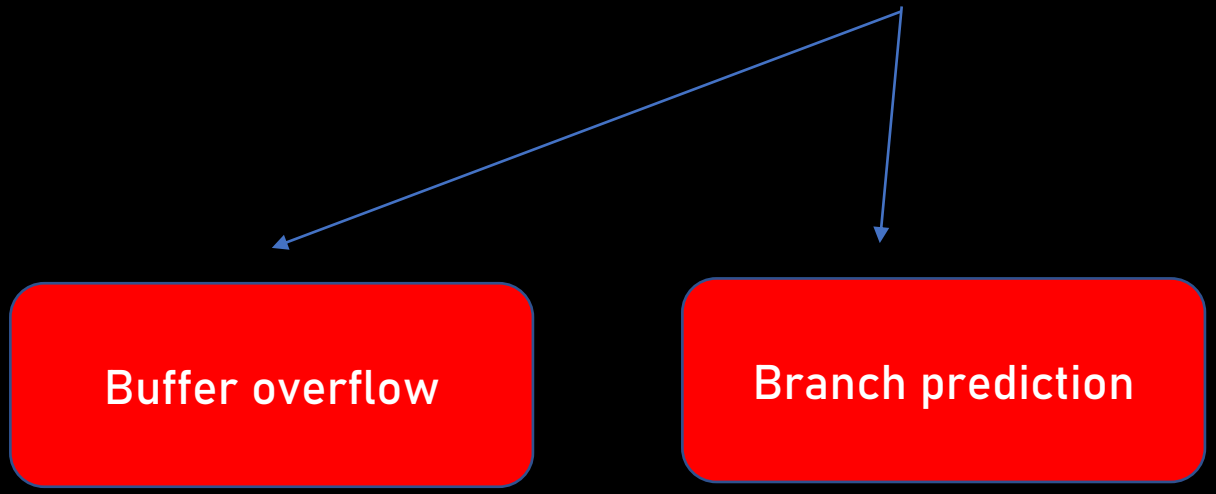
Comment cette information peut être utilisé de façon malicieuse ?



Memory issues: Corruption - access

Check the mappings for your C program and also **cat /proc/self/maps**

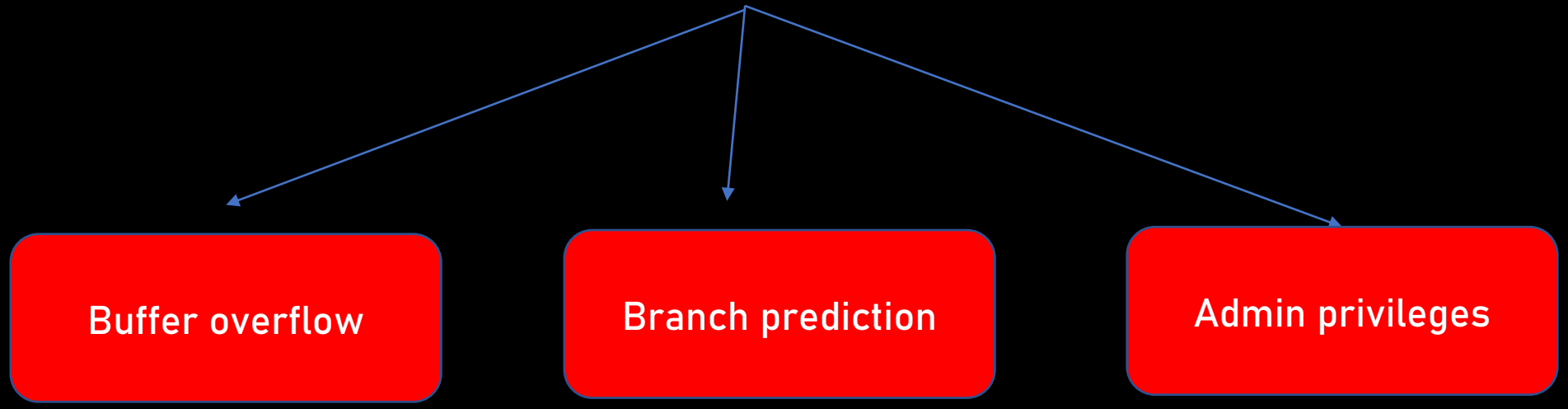
Comment cette information peut être utilisé de façon malicieuse ?



Memory issues: Corruption - access

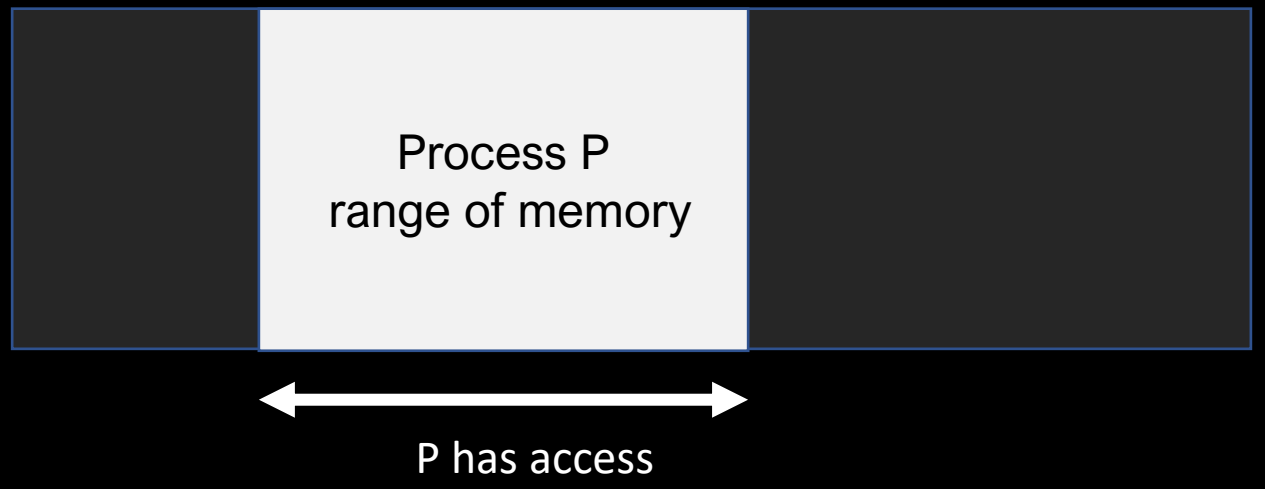
Check the mappings for your C program and also **cat /proc/self/maps**

Comment cette information peut être utilisé de façon malicieuse ?



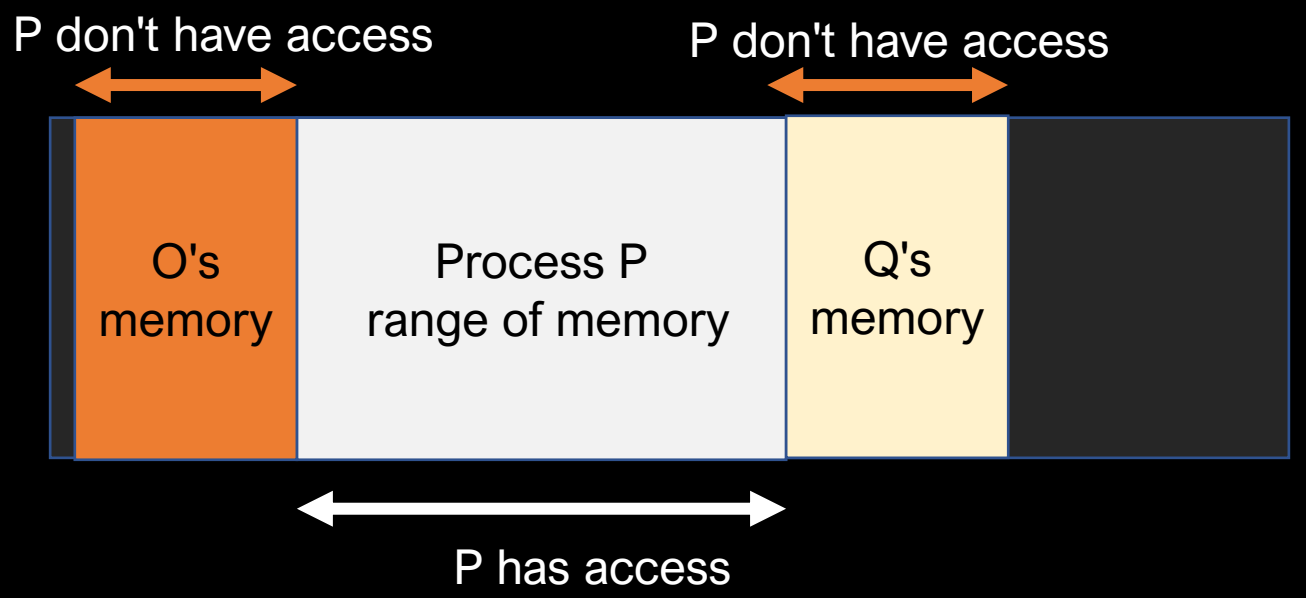
Buffer overflow

Utiliser les zones de mémoires d'un processus pour essayer de lire les zones en dehors de sa juridiction.



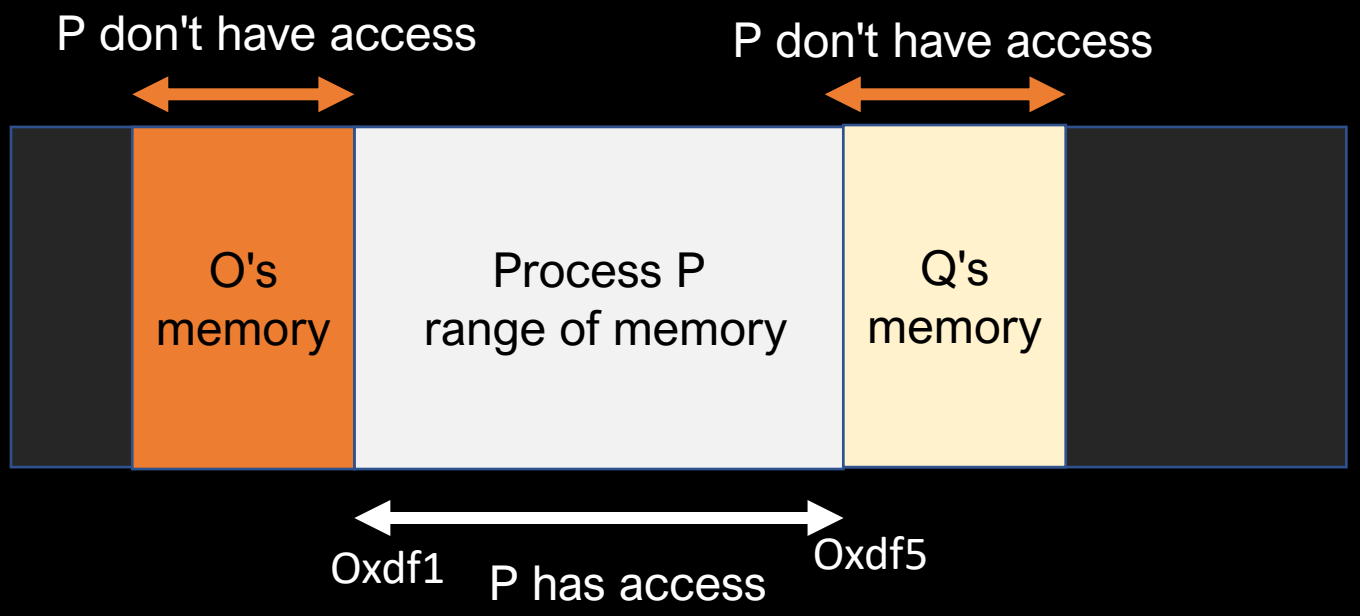
Buffer overflow

Utiliser les zones de mémoires d'un processus pour essayer de lire les zones en dehors de sa juridiction.



Buffer overflow

Utiliser les zones de mémoires d'un processus pour essayer de lire les zones en dehors de sa juridiction.



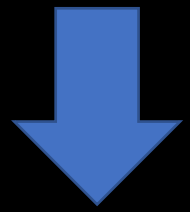
But what happens if Q's memory is not correctly protected?

Then P can go overbound (overflow). Any process can try to go overbound by manually triggering a read at a specific address.

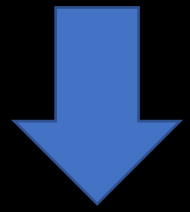
Buffer overflow

```c
#include <stdio.h>
#include <string.h>

int main(void)
{
    char buff[15];
    int pass = 0;

    printf("\n Enter the password : \n");
    gets(buff);

    if(strcmp(buff, "ndoleplantain"))
        printf ("\n Wrong Password \n");
    else
    {
        printf ("\n Correct Password \n");
        pass = 1;
    }

    if(pass)
       /* Now Give root or admin rights to user*/
        printf ("\n Root privileges given to the user \n");

    return 0;
}
```

Exemple: Code d'authentification en C



gcc -ooverflow.o overflow.c –fno-stack-protector –zexecstatck –fno-pie



./overflow.o

What do you observe ?

Buffer overflow

Morris Worm: The Morris worm of 1988 was one of the first internet-distributed computer worms, and the first to gain significant mainstream media attention. It exploited a buffer overflow vulnerability in the Unix sendmail, finger, and rsh/rexec, infecting 10% of the

SQL Slammer: SQL Slammer is a 2003 computer worm that exploited a buffer overflow bug in

Adobe Flash Player: In 2016, a buffer overflow vulnerability was found in Adobe Flash Player for Windows, macOS, Linux and Chrome OS. The vulnerability was due to an error in Adobe Flash Player while parsing a specially crafted SWF (Shockwave Flash) file. Malicious

WhatsApp VoIP: In May 2019, Facebook announced a vulnerability associated with all of its WhatsApp products. The vulnerability exploited a buffer overflow weakness in WhatsApp's VOIP stack on smartphones. This allows remote code execution via a specially-crafted

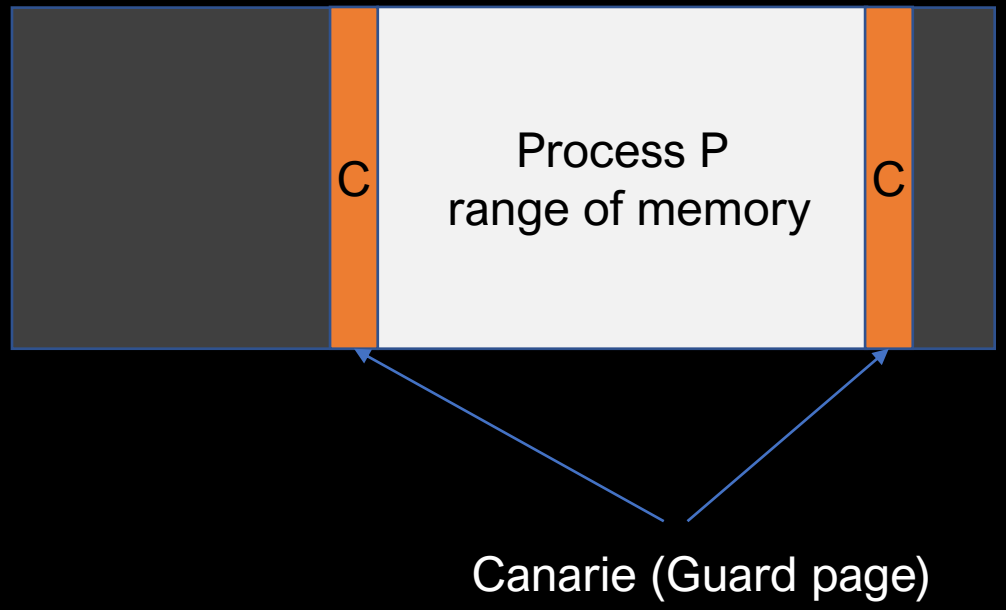
Branch prediction

Admin privileges

Se protéger contre Buffer overflow

Activer les canaries dans un système d'exploitation

Page mémoire placées à la fin d'une zone de mémoire afin de détecter des débordements



Branch prediction

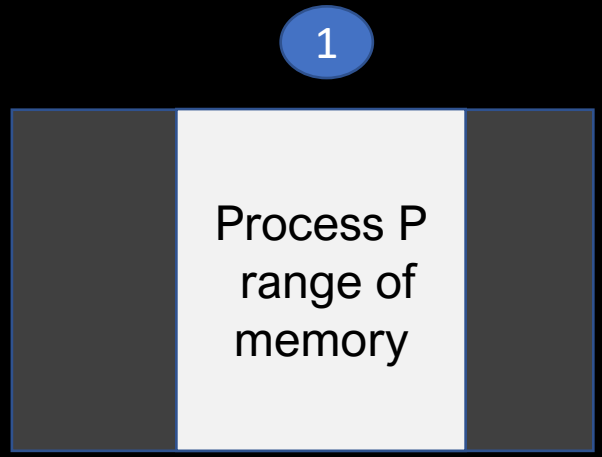
Admin privileges

Se protéger contre Buffer overflow

Le système va régulièrement changer l'emplacement des adresses de votre tas pour bloquer des attaques liés à l'ancien emplacement

Activer les canaries dans un système d'exploitation

(K)ASLR – (Kernel) Adress Space Layout Randomization



Page mémoire placées à la fin d'une zone de mémoire afin de détecter des débordements

Branch prediction

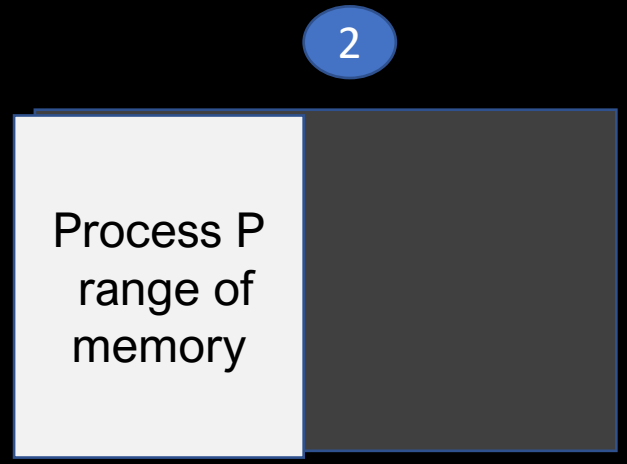
Admin privileges

Se protéger contre Buffer overflow

Le système va régulièrement changer l'emplacement des adresses de votre tas pour bloquer des attaques liés à l'ancien emplacement

Activer les canaries dans un système d'exploitation

(K)ASLR – (Kernel) Adress Space Layout Randomization



Page mémoire placées à la fin d'une zone de mémoire afin de détecter des débordements

Branch prediction

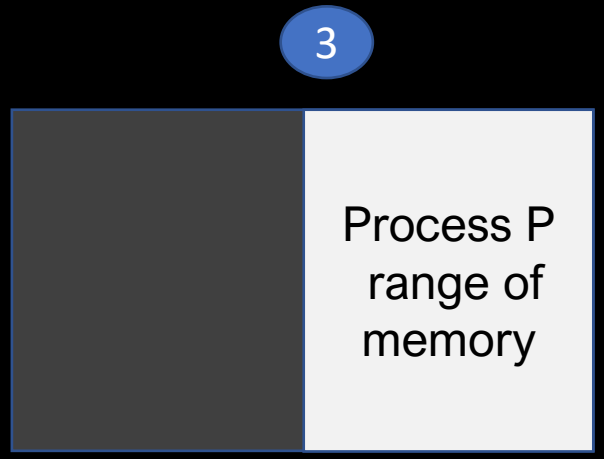
Admin privileges

Se protéger contre Buffer overflow

Le système va régulièrement changer l'emplacement des adresses de votre tas pour bloquer des attaques liés à l'ancien emplacement

Activer les canaries dans un système d'exploitation

(K)ASLR – (Kernel) Adress Space Layout Randomization



Page mémoire placées à la fin d'une zone de mémoire afin de détecter des débordements

Branch prediction

Admin privileges

Se protéger contre Buffer overflow

Le système va régulièrement changer l'emplacement des adresses de votre tas pour bloquer des attaques liés à l'ancien emplacement

Activer les canaries dans un système d'exploitation

(K)ASLR – (Kernel) Adress Space Layout Randomization

Limiter les zones de mémoire avec les droits d'exécution

Page mémoire placées à la fin d'une zone de mémoire afin de détecter des débordements

Branch prediction

Admin privileges

Exploiter les prédicteur de branchement

```
if(x > 20)
  printf("Je fais ce que je veux")
if(x < 20 && x > 10)
  printf("Courage!!! D'ici 10 ans max")
else
  printf(":(")
```

Que fais le processeur lorsqu'il arrive sur un branchement ?

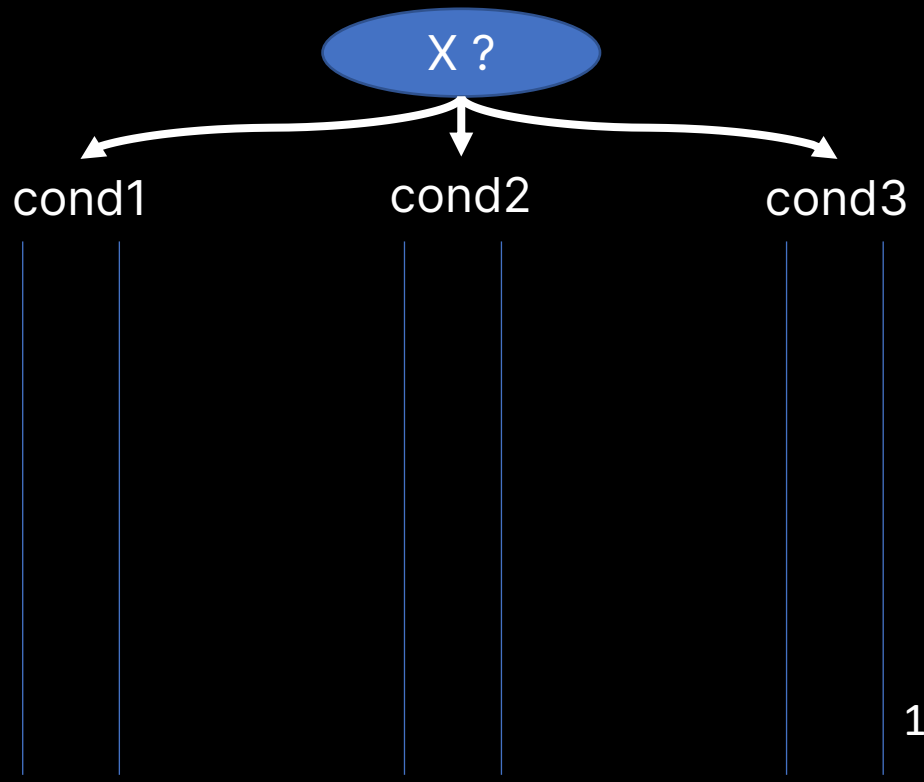
Buffer overflow

Admin privileges

Exploiter les prédicteur de branchement

```c
if(x > 20)
  printf("Je fais ce que je veux")
if(x < 20 && x > 10)
  printf("Courage!!! D'ici 10 ans max")
else
  printf(":(")
```

Que fais le processeur lorsqu'il arrive sur un branchement ?



Buffer Overflow

Admin privileges

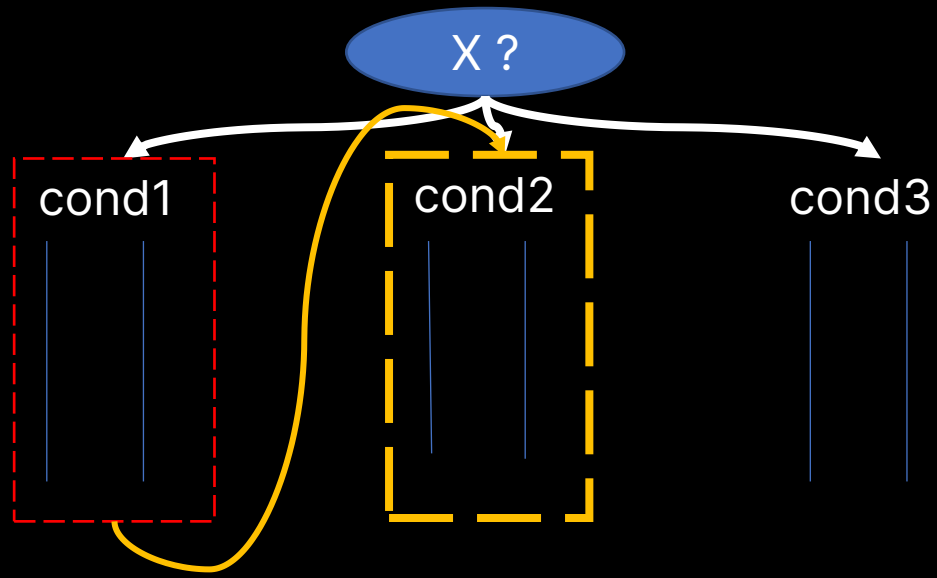
Exploiter les prédicteur de branchement

```
if(x > 20)
  printf("Je fais ce que je veux")
if(x < 20 && x > 10)
  printf("Courage!!! D'ici 10 ans max")
else
  printf(":(")
```

Le processeur n'attends pas la valeur de X et choisi une route

Si X correspond, alors le processeur continue

Si X ne correspond pas, alors le processeur vide son cache et revient en arrière



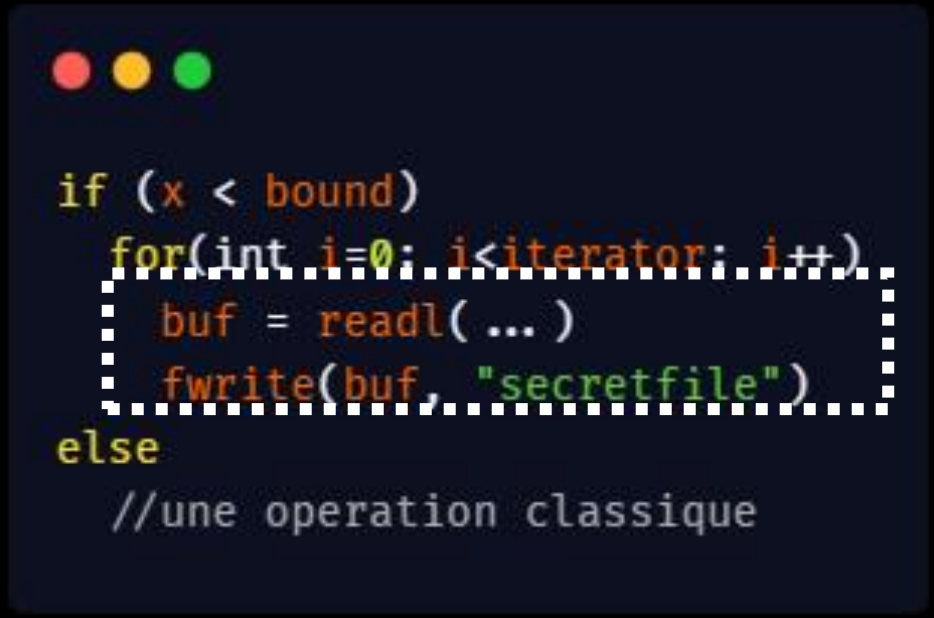
Buffer overflow

Admin privileges

Exploiter les prédicteur de branchement

Pour pouvoir actionner le rollback et le flush des caches en cas d'erreur, lors d'une prédiction de branchement, le processeur possède des superdroits durant ce lapse de temps.

Une attaque consiste donc à forcer le processeur à lire des zones mémoires sécurisée pour exfiltrer leur contenu lors d'une prédiction de branchement



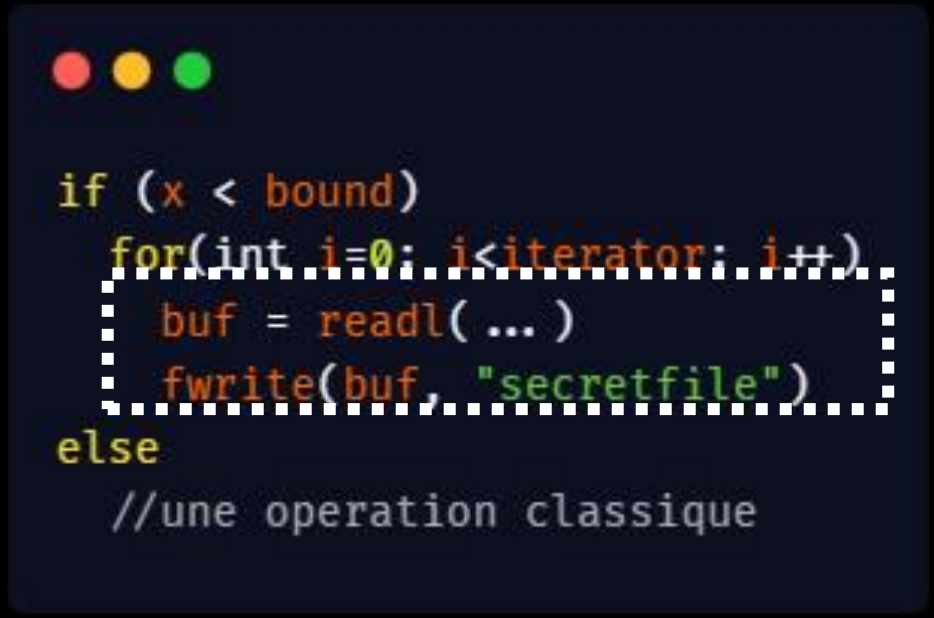
Buffer overflow

Admin privileges

Exploiter les prédicteur de branchement

Ce type d'attaques exploitant les prédictions de branchement sont dit Spectre ou Meltdown Side Channel attacks.

Une attaque consiste donc à forcer le processeur à lire des zones mémoires sécurisée pour exfiltrer leur contenu lors d'une prédiction de branchement



https://spectreattack.com/spectre.pdf

Buffer overflow

Admin privileges

Débats

Une boite se soucie des buffer overflow et ne sait pas si la meilleure solution c'est d'implémenter toutes les protections (canaries, ASLR, etc.) au risque de voir les performances diminuer ou juste former le personnel sur les meilleures pratiques de programmation. De plus, elle se demande quel système d'exploitation utilisé pour combattre au mieux les failles et évoluer dans le temps.

Débat 2: Defender l'idée que former le personnel est une meilleure approche ou l'idée qu'implémenter les mécanismes de protections serait plus adaptés.

Débat 3: Choisissez entre une distribution Linux et Windows et défendez le système d'exploitation de votre choix (par rapport à la problématique de la sécurité).