Vulnerabilidades e Ataques em Sistemas

Jorge Correia

Segurança Computacional
Universidade Federal do Paraná

2024

Condição de Corrida

"Condição em que o resultado do processo é dependente da sequência ou sincronia de outros eventos"

Condição de Corrida

"Condição em que o resultado do processo é dependente da sequência ou sincronia de outros eventos"

- A execução de operações em ordem não prevista pode levar o sistema a um estado inconsistente

Condição de Corrida

```
def producer():
    while 1:
        if buffer_size == FULL:
            sleep
        ...
        buffer_size++
        wake_consumers()

def consumer():
    while 1:
        if buffer_size == EMPTY:
            sleep
        ...
        buffer_size--
        wake_producers()
```

Time-of-check time-of-use (TOCTOU)

- “Um bug causado por uma condição de corrida”.

Time-of-check time-of-use (TOCTOU)

- “Um bug causado por uma condição de corrida”.
- Problemas que variam de computações incorretas a acessos indevidos

Time-of-check time-of-use (TOCTOU)

```
if (access("file", W_OK) != 0) {
    exit(1);
}

fd = open("file", O_WRONLY);
write(fd, buffer, sizeof(buffer));
```

Dirty CoW

https://dirtycow.ninja/

- Condição de corrida no subsistema de Copy-On-Write (CoW)

Dirty CoW

https://dirtycow.ninja/

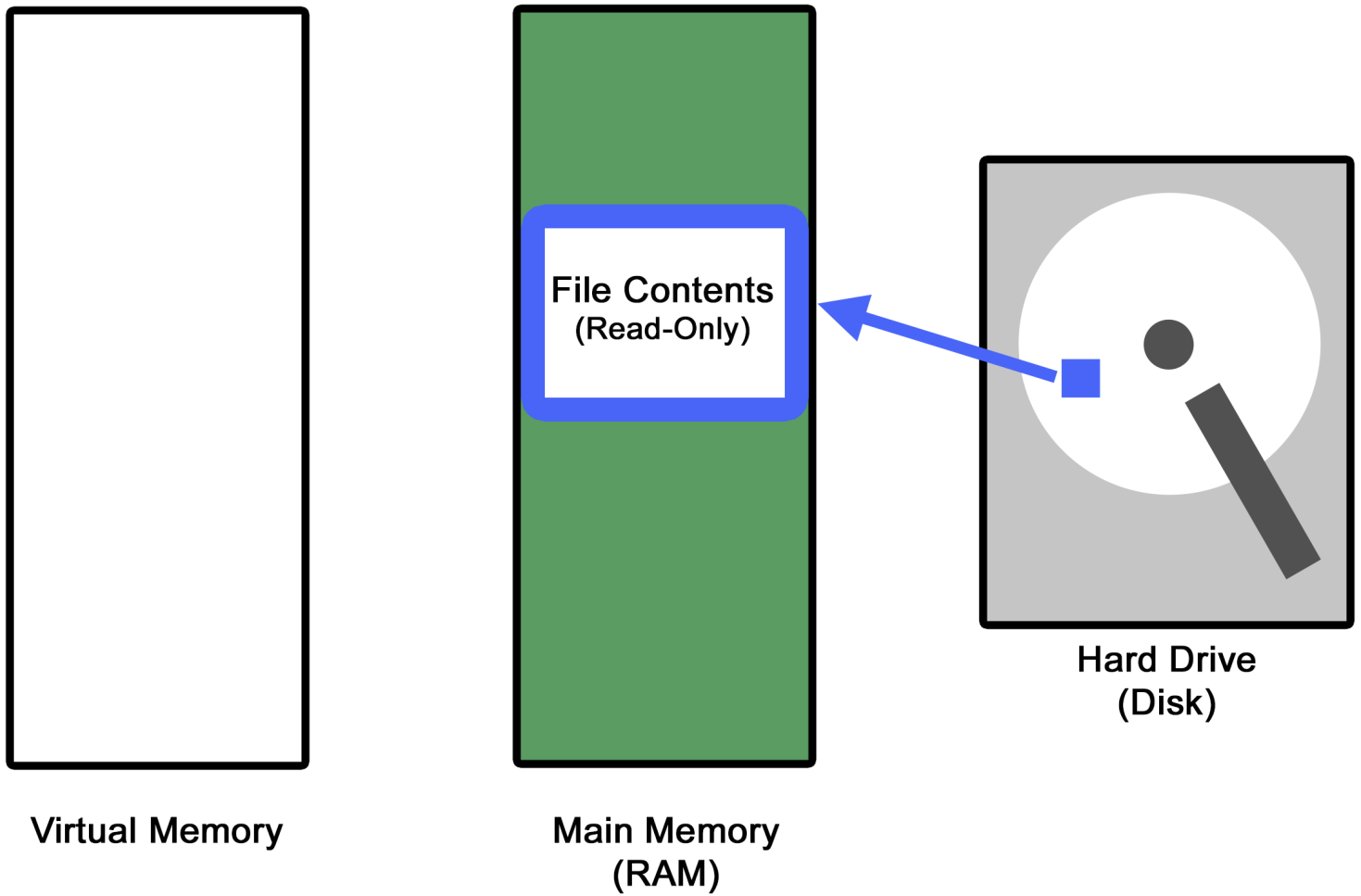
- Condição de corrida no subsistema de Copy-On-Write (CoW)
- Introduzida no kernel em 2007 (2.6.22)

Dirty CoW

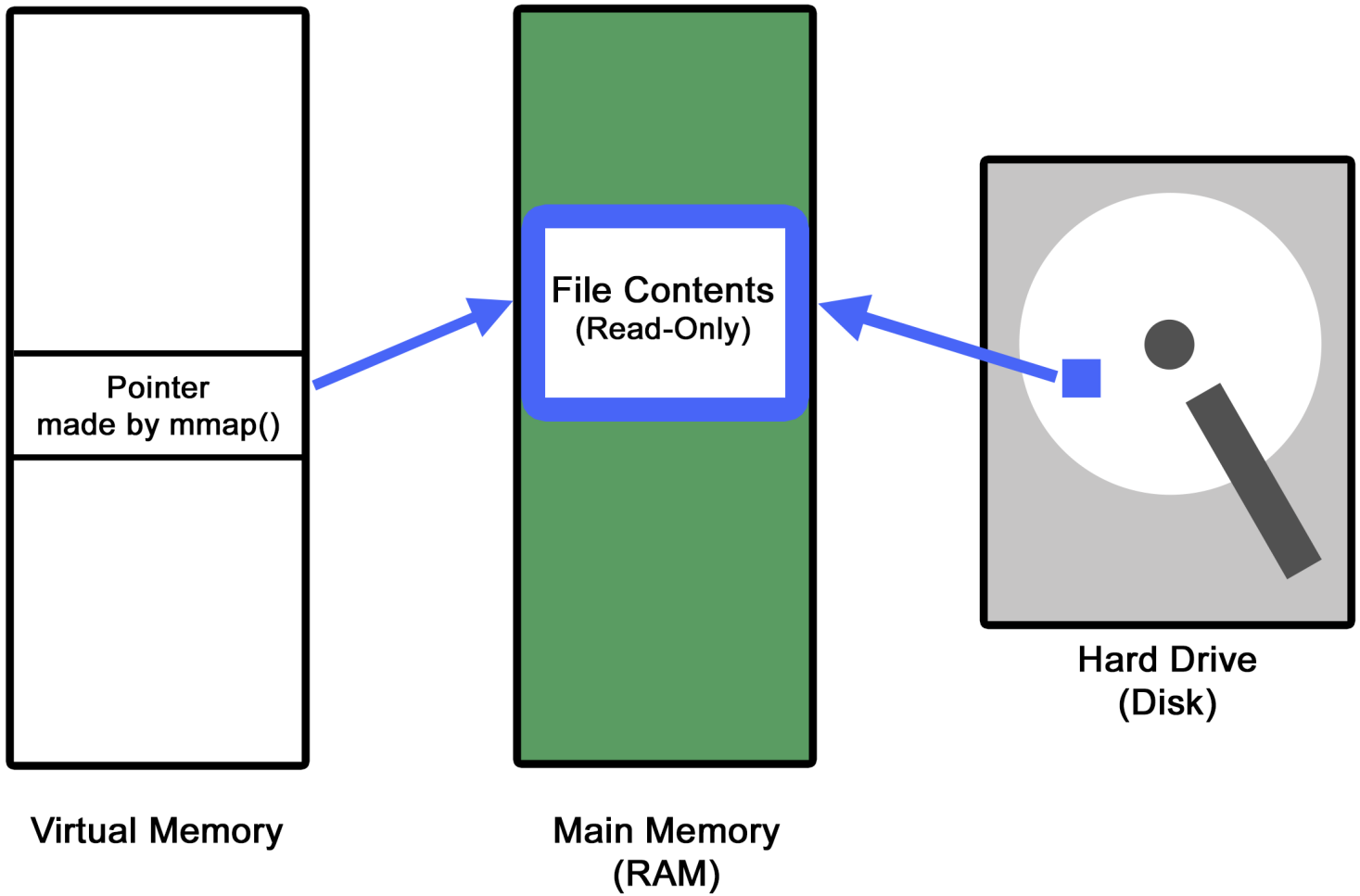
https://dirtycow.ninja/

- Condição de corrida no subsistema de Copy-On-Write (CoW)
- Introduzida no kernel em 2007 (2.6.22)
- Patch de correção total em 2017 (4.8.3, 4.7.9, 4.4.26)

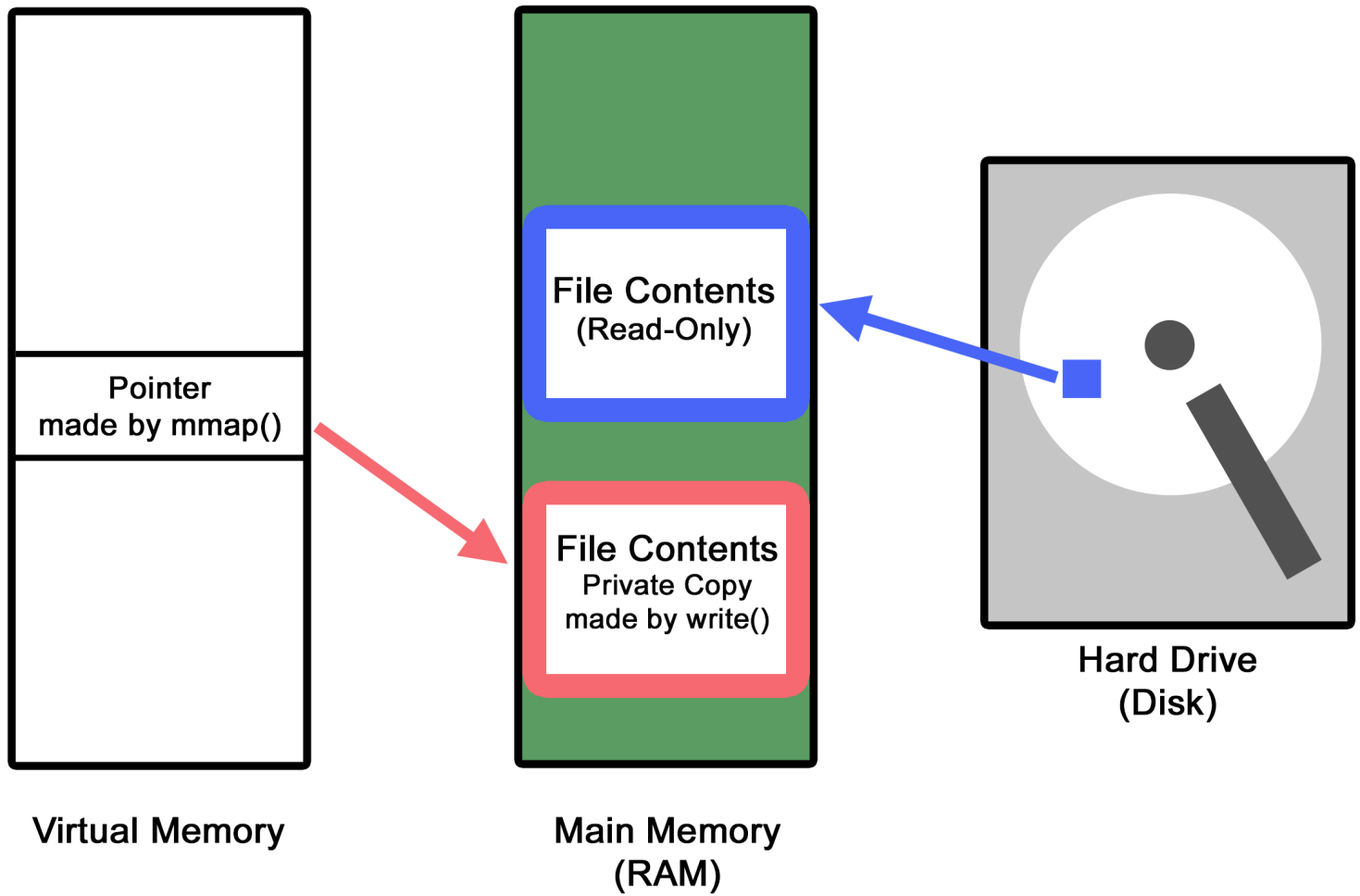
Dirty CoW


File Contents
(Read-Only)
Hard Drive
(Disk)
Virtual Memory
Main Memory
(RAM)

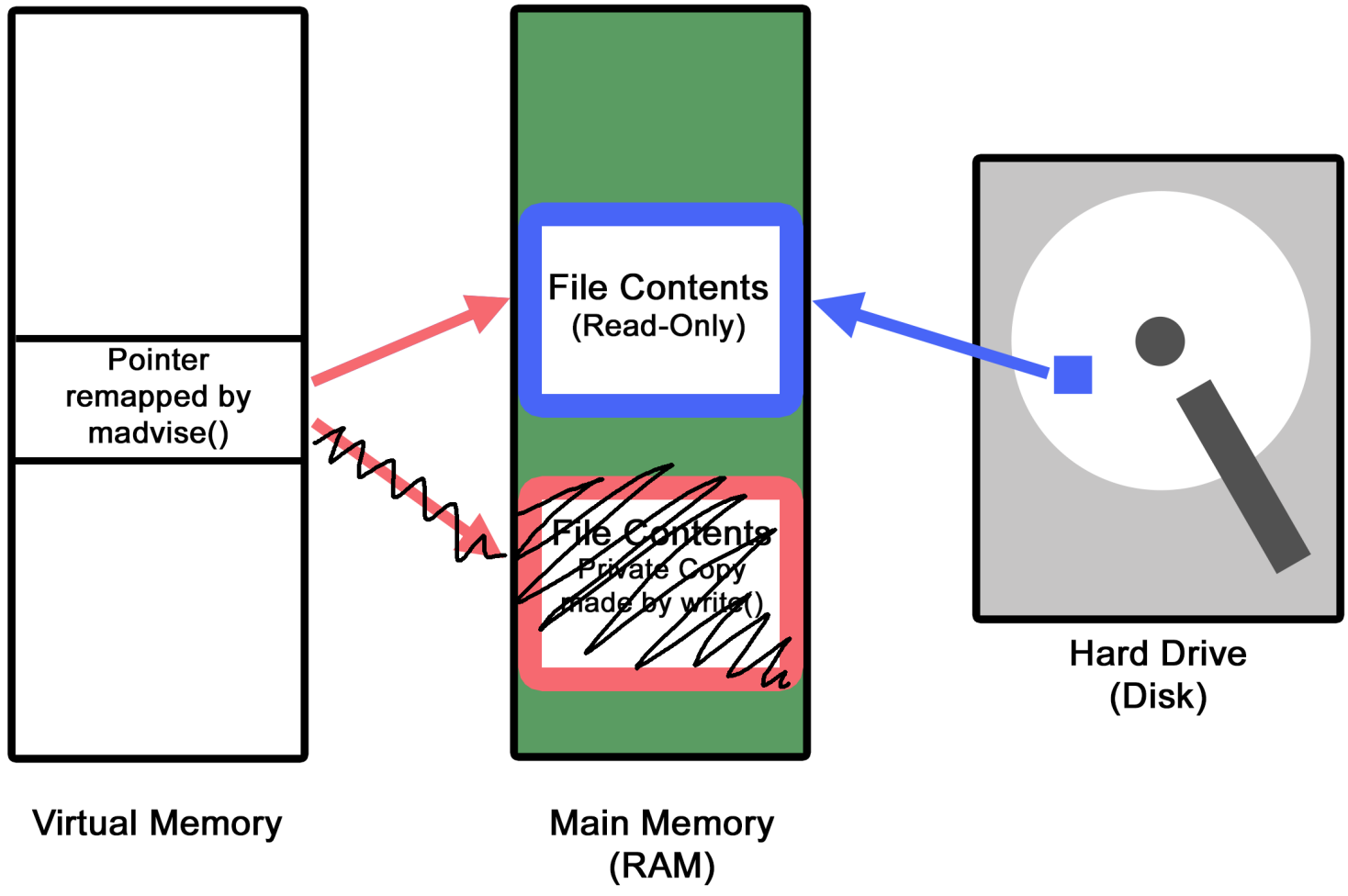
Dirty CoW


Pointer
made by mmap()
File Contents
(Read-Only)
Hard Drive
(Disk)
Virtual Memory
Main Memory
(RAM)

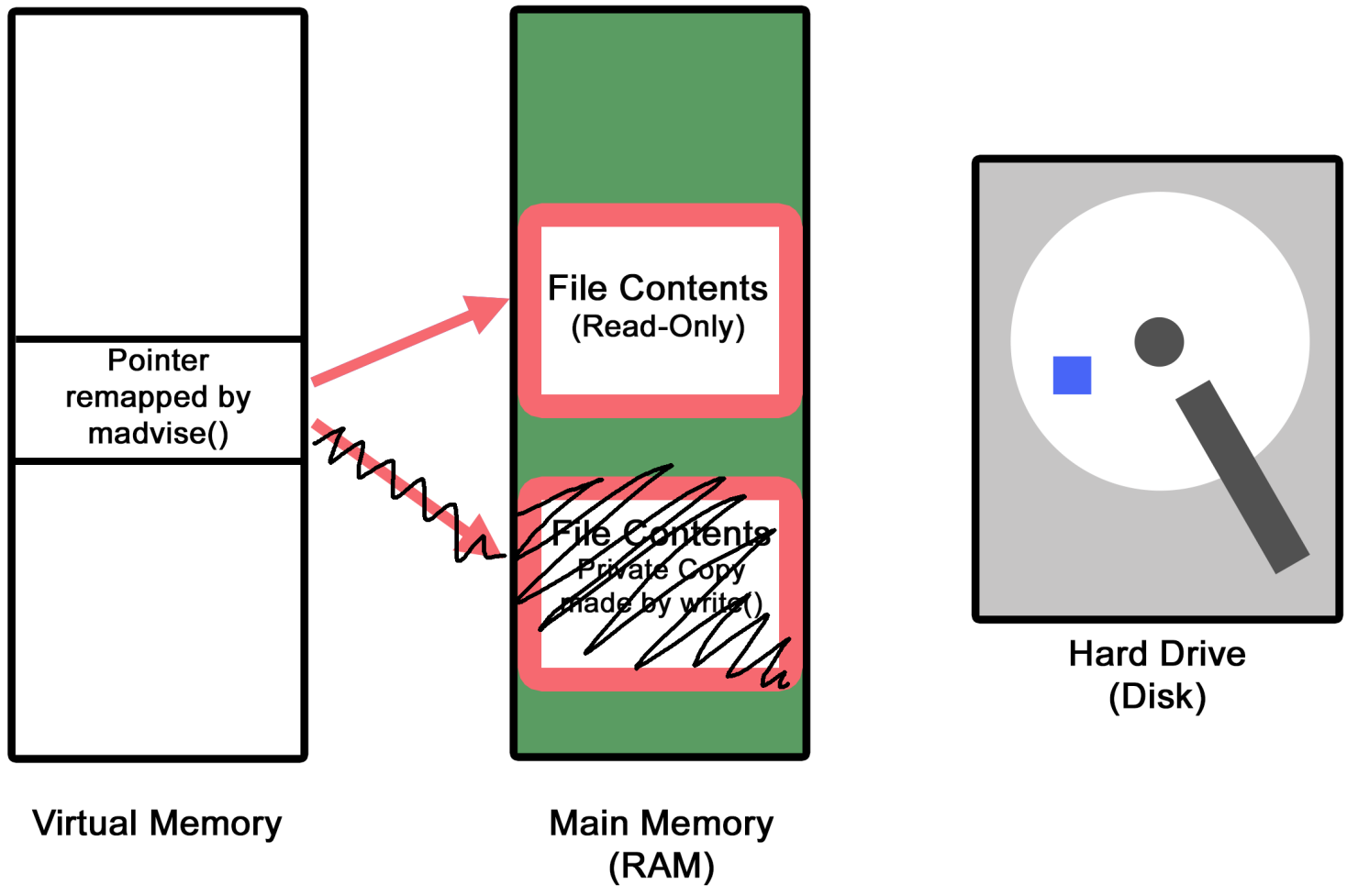
Dirty CoW


Pointer
made by mmap()
File Contents
(Read-Only)
File Contents
Private Copy
made by write()
Hard Drive
(Disk)
Virtual Memory
Main Memory
(RAM)

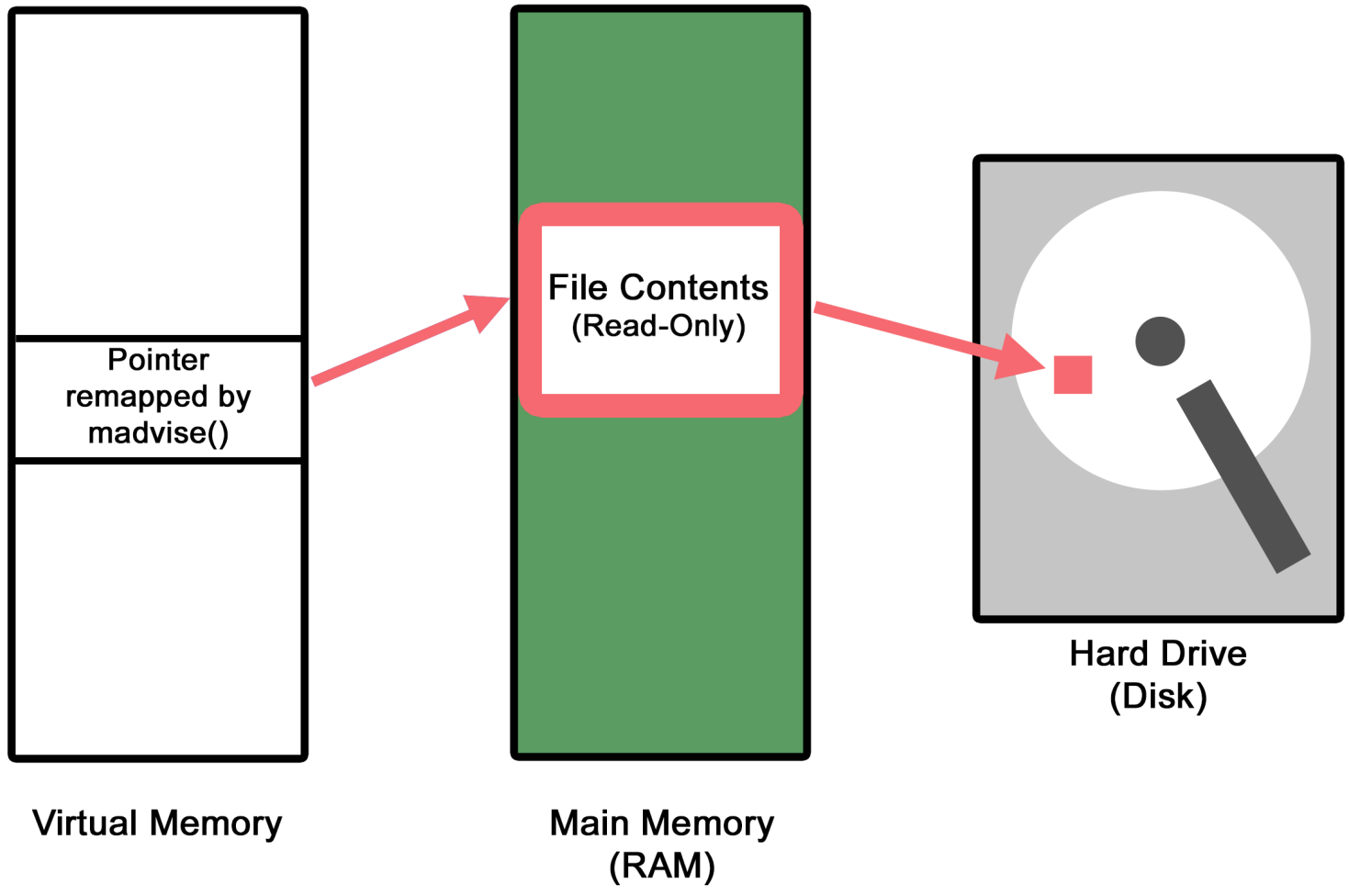
Dirty CoW


Pointer
remapped by
madvise()
File Contents
(Read-Only)
File Contents
Private Copy
made by write()
Hard Drive
(Disk)
Virtual Memory
Main Memory
(RAM)

Dirty CoW


Pointer
remapped by
madvise()
File Contents
(Read-Only)
File Contents
Private Copy
made by write()
Hard Drive
(Disk)
Virtual Memory
Main Memory
(RAM)

Dirty CoW

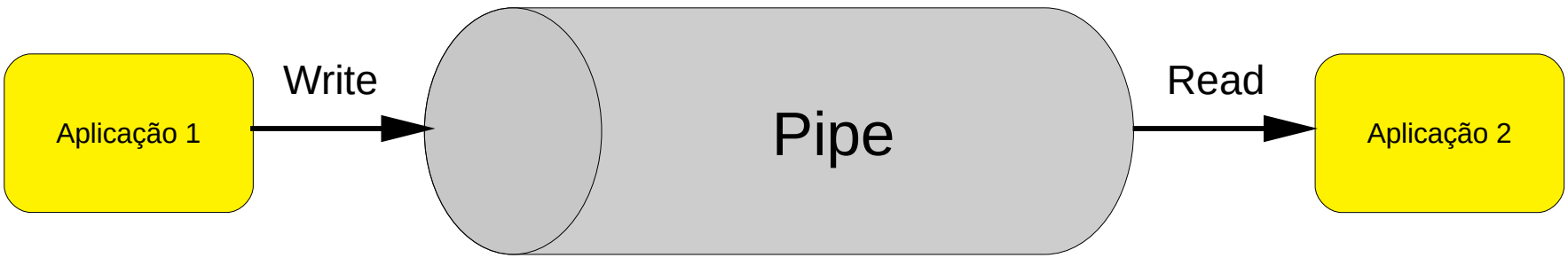

Pointer
remapped by
madvise()
File Contents
(Read-Only)
Hard Drive
(Disk)
Virtual Memory
Main Memory
(RAM)

Dirty Pipe

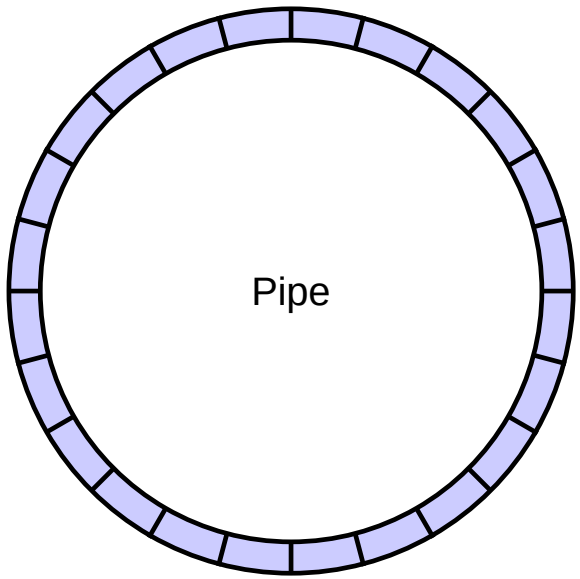
https://dirtypipe.cm4all.com/

- Introduzida na versão 5.8
- Corrigida nas versões 5.16.11, 5.15.25 e 5.10.102

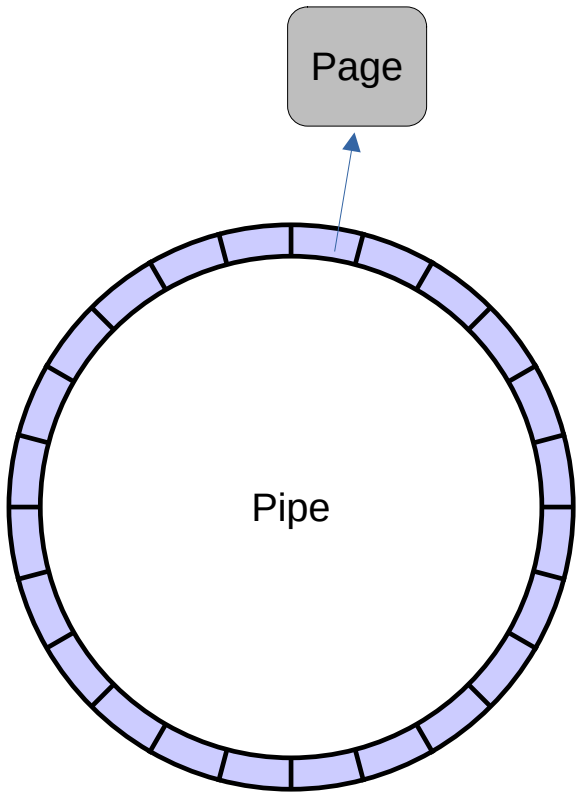
Pipes



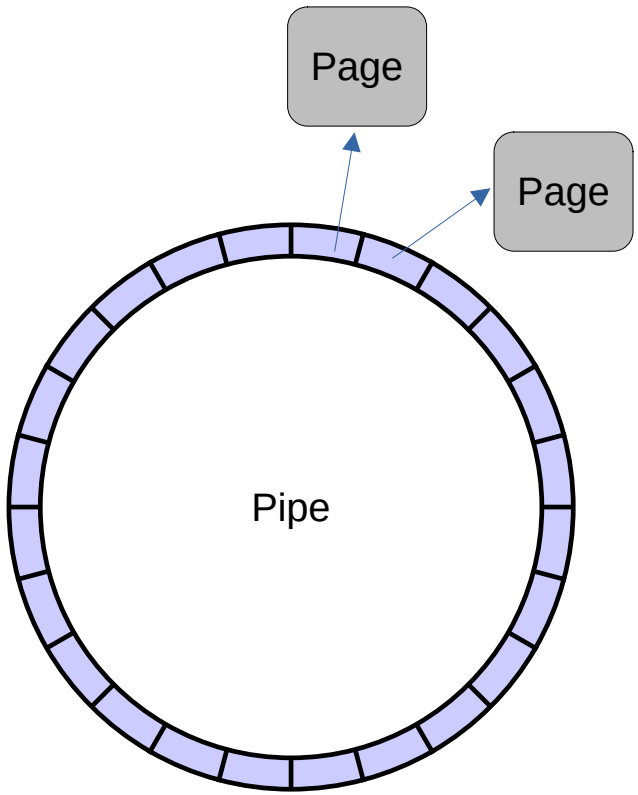
Pipes



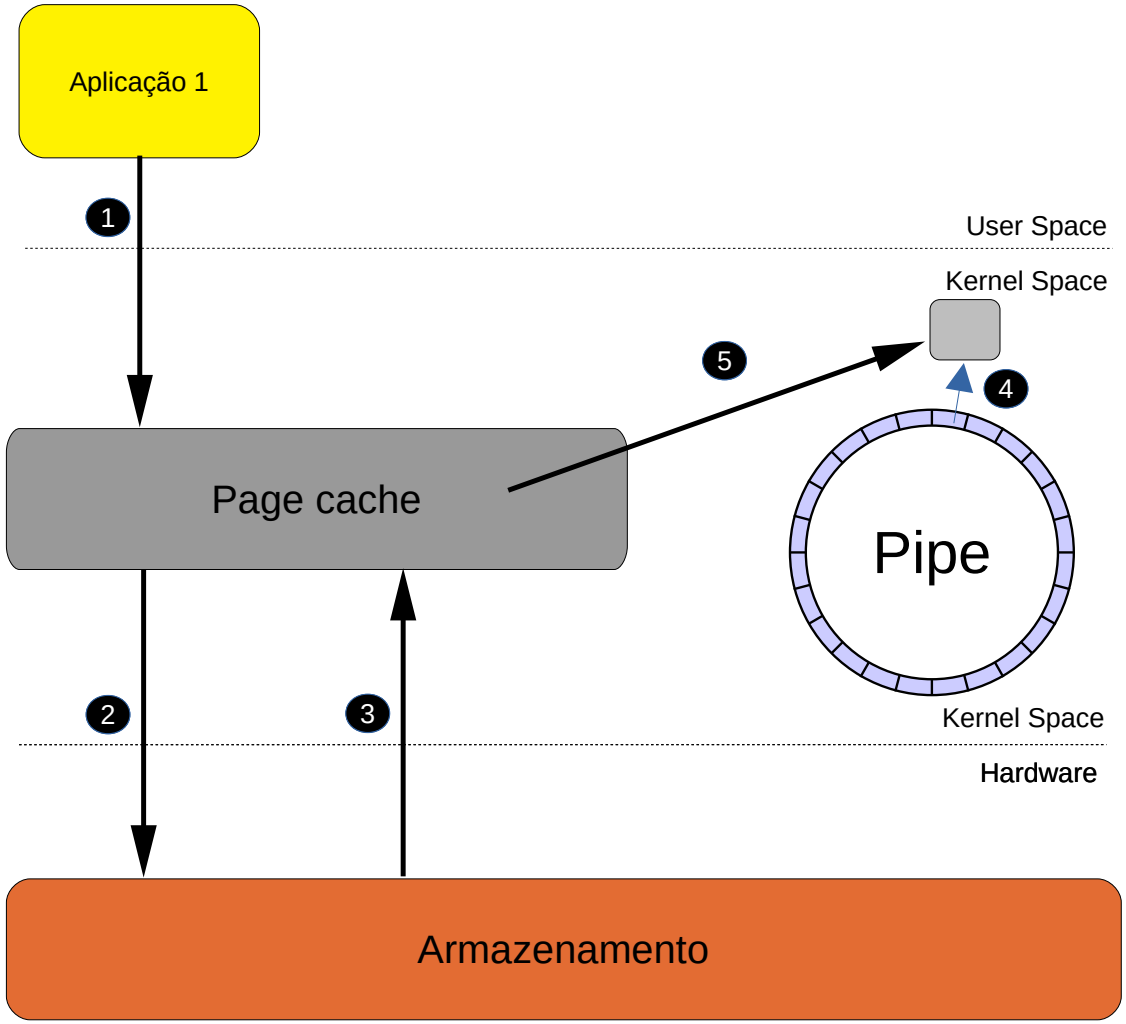
Pipes



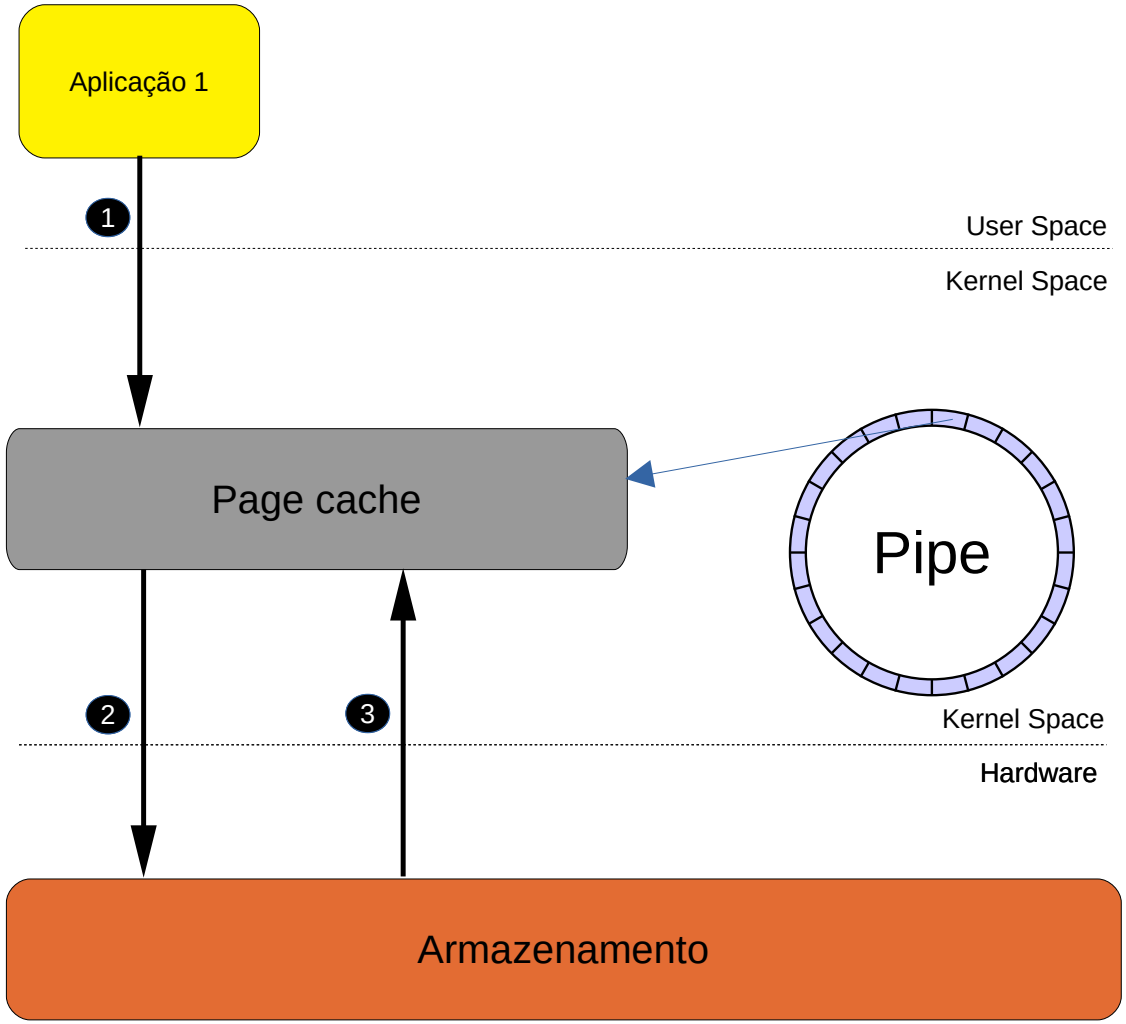
Pipes



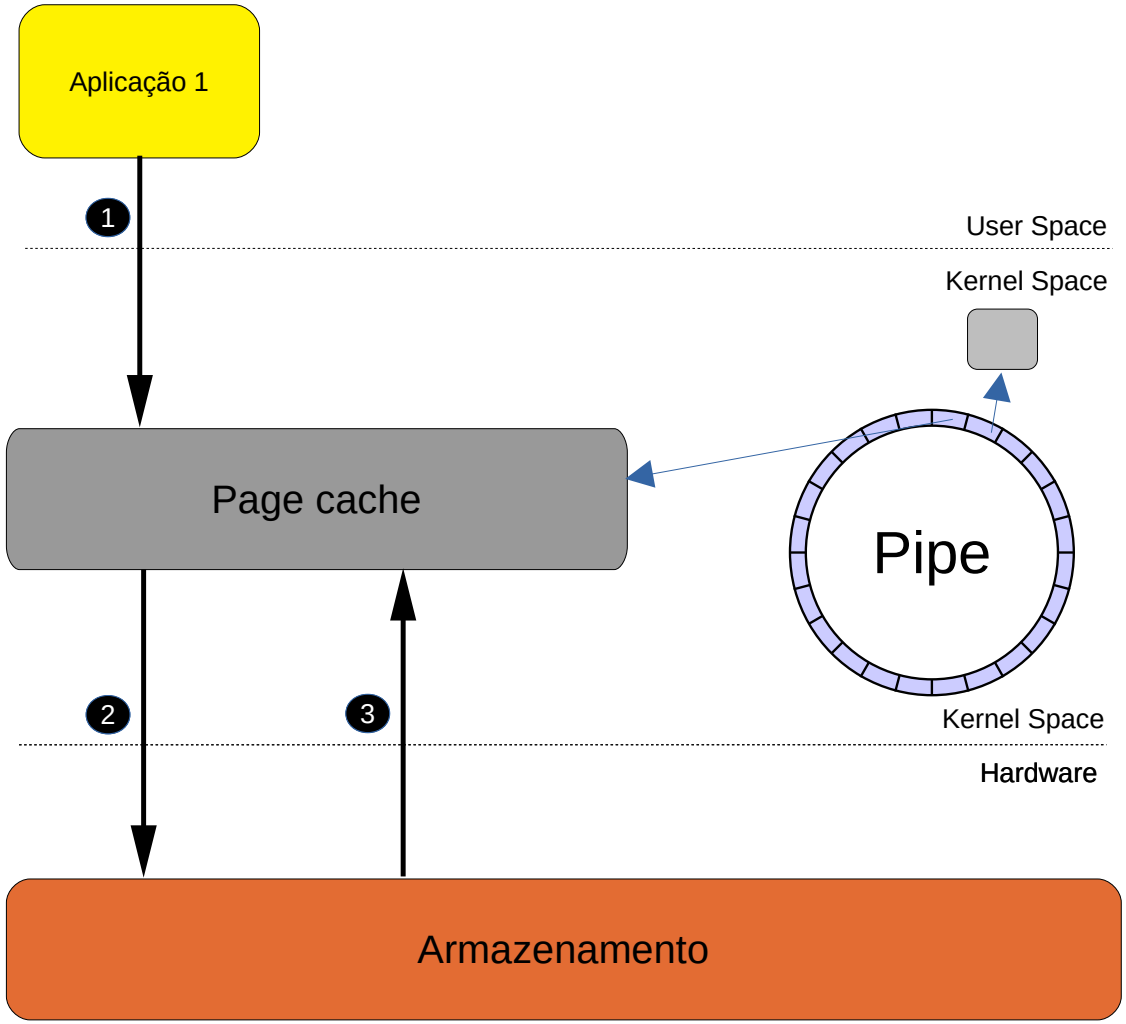
Syscall Splice



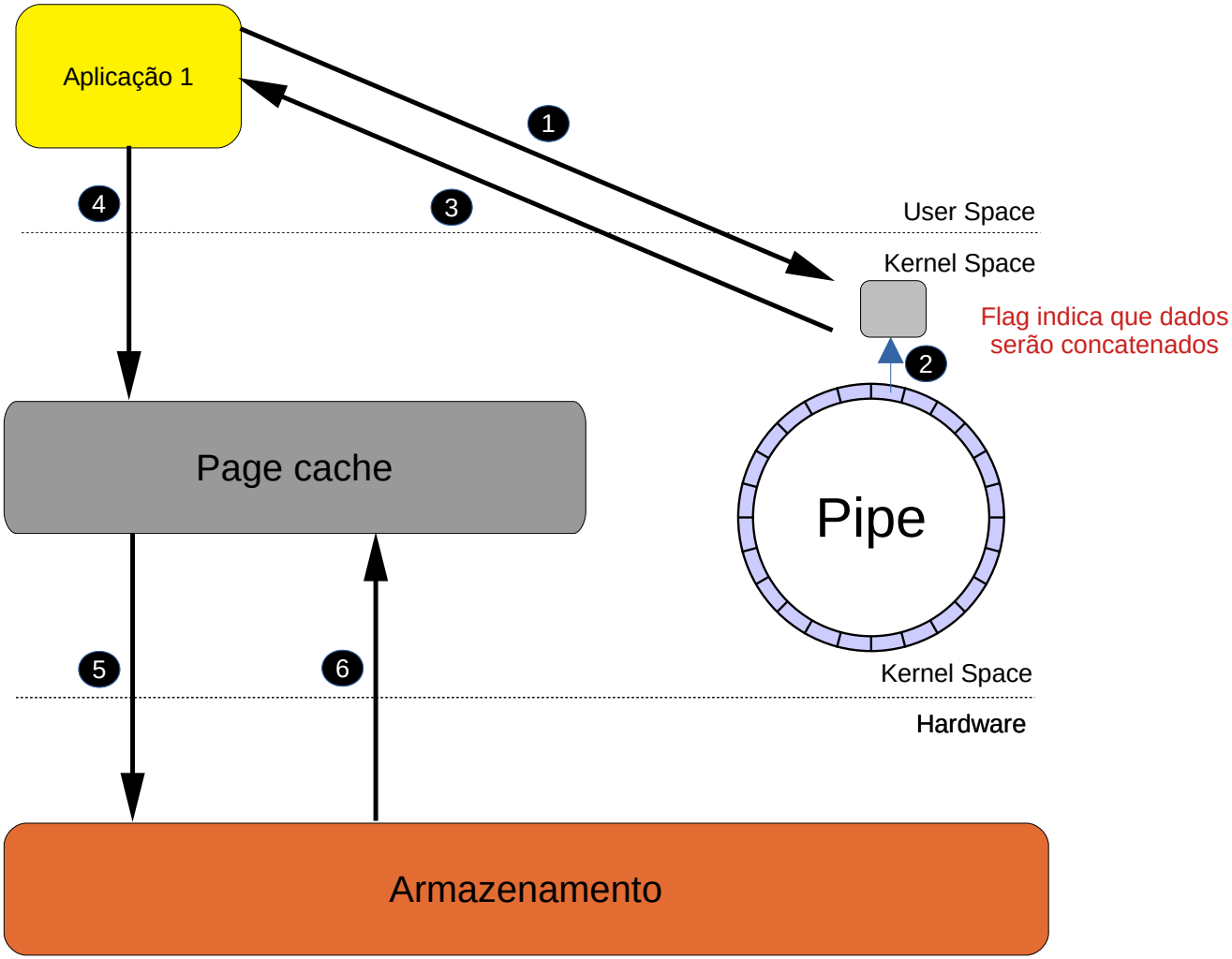
Syscall Splice



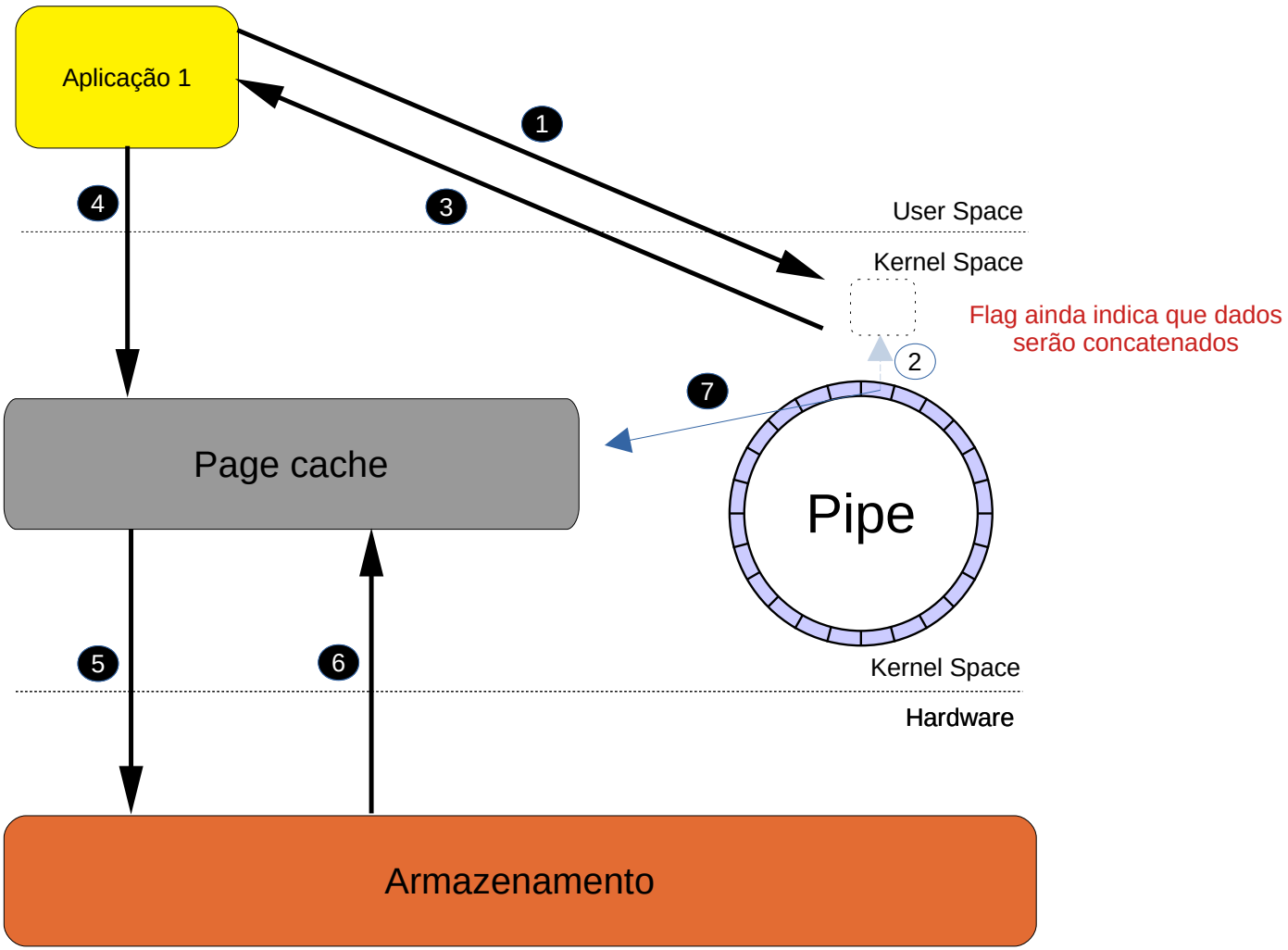
Syscall Splice



Exploração da Vulnerabilidade



Exploração da Vulnerabilidade



- O patch de correção consiste apenas na linha de código "flags = 0"

- O patch de correção consiste apenas na linha de código "flags = 0"
- A struct pipe_buffer é bastante visada em exploits de kernel por ter informações sensíveis

- O patch de correção consiste apenas na linha de código "flags = 0"
- A struct pipe_buffer é bastante visada em exploits de kernel por ter informações sensíveis
- Percebem que o conhecimento para a exploração da vulnerabilidade consiste no conhecimento do sistema a ser atacado?

Buffer Overflow (BO)

- Vulnerabilidade onde um programa permite a escrita de dados além dos limites de um buffer

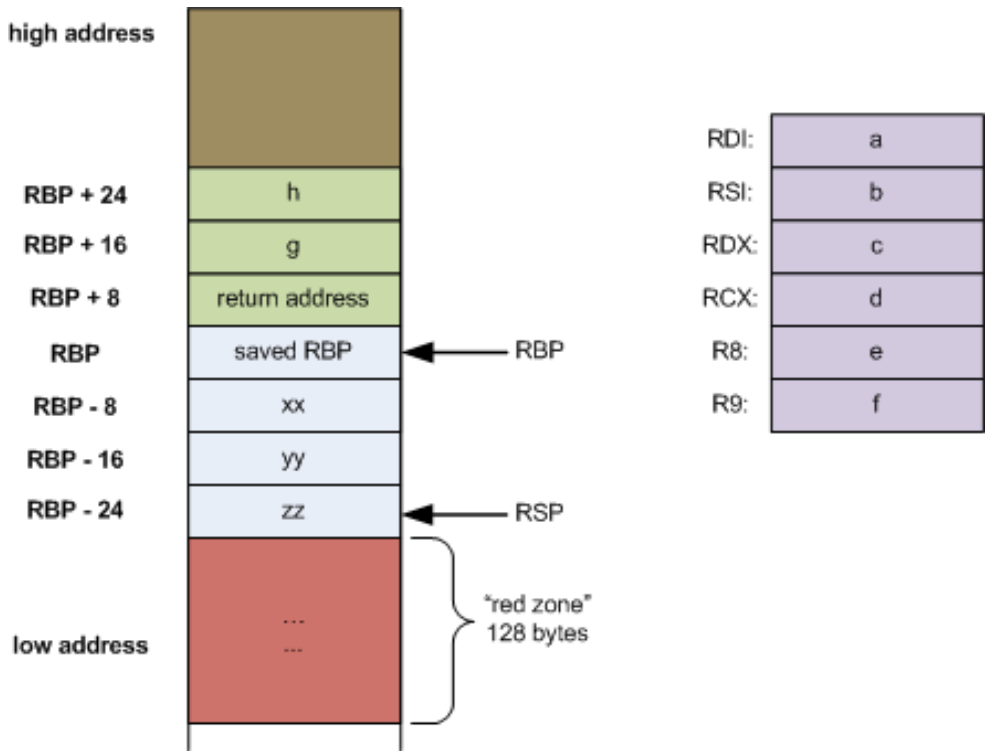
Buffer Overflow (BO)

- Vulnerabilidade onde um programa permite a escrita de dados além dos limites de um buffer
- Comum em buffers da stack

Buffer Overflow (BO)

Exemplo

Relembrando a Stack



Ferramentas para auxiliar no exploração de binários: pwntools, ghidra, r2, ...

- E se não existir uma função específica que é útil para um atacante?

- E se não existir uma função específica que é útil para um atacante?
- Shellcode!

http://shell-storm.org/shellcode/index.html

- É possível sobrescrever o buffer com o shellcode

- É possível sobrescrever o buffer com o shellcode
- Sobrescrever o endereço de retorno com o começo do shellcode

- Normalmente a pilha não tem permissão de execução. E agora?

Return-oriented programming (ROP)

- Normalmente a pilha não tem permissão de execução. E agora?
- ROP!

Return-oriented programming (ROP)

- A ideia do ROP é utilizar códigos já presentes no espaço de endereçamento do processo e que tem permissão de execução

Return-oriented programming (ROP)

- A ideia do ROP é utilizar códigos já presentes no espaço de endereçamento do processo e que tem permissão de execução
- Código do próprio programa ou em códigos de bibliotecas

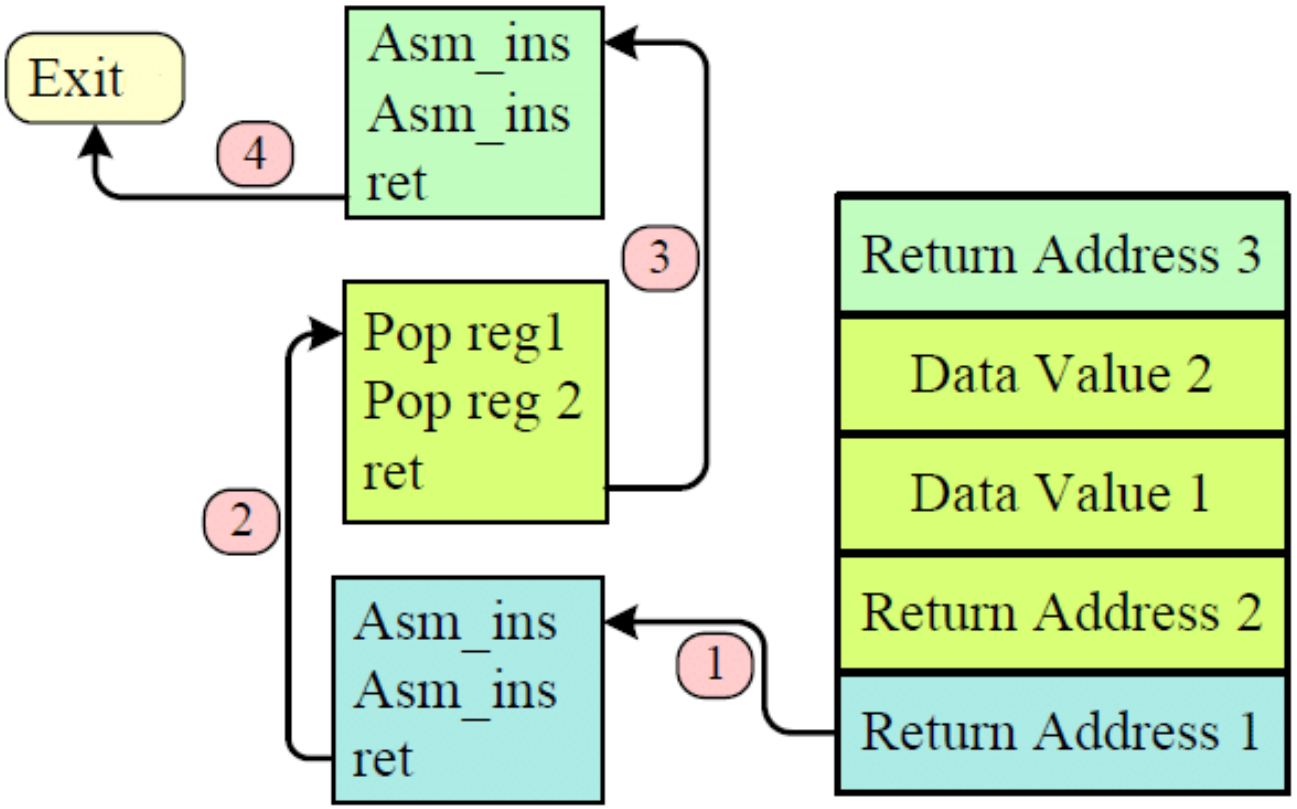
Return-oriented programming (ROP)

- A ideia do ROP é utilizar códigos já presentes no espaço de endereçamento do processo e que tem permissão de execução
- Código do próprio programa ou em códigos de bibliotecas
- Isso é realizado através da escolha de pedaços de códigos chamados "gadgets"

Return-oriented programming (ROP)

- A ideia do ROP é utilizar códigos já presentes no espaço de endereçamento do processo e que tem permissão de execução
- Código do próprio programa ou em códigos de bibliotecas
- Isso é realizado através da escolha de pedaços de códigos chamados "gadgets"
- Um getget é uma sequencia de instruções que são finalizadas com a instrução "ret"

Return-oriented programming (ROP)



- Como achar os getgets? Realmente é possível executar código útil?

- Como achar os getgets? Realmente é possível executar código útil?
- Pode prover funcionalidades turing-completas para o atacante

Return-oriented programming (ROP)

- Como achar os getgets? Realmente é possível executar código útil?
- Pode prover funcionalidades turing-completas para o atacante
- Arquitetura x86 (CISC) possui um ISA desalinhado e denso

Return-oriented programming (ROP)

- Como achar os getgets? Realmente é possível executar código útil?
- Pode prover funcionalidades turing-completas para o atacante
- Arquitetura x86 (CISC) possui um ISA desalinhado e denso
- Uma sequência de bytes que acabe com 0xc3

- Na prática não procuramos gatgets na mão

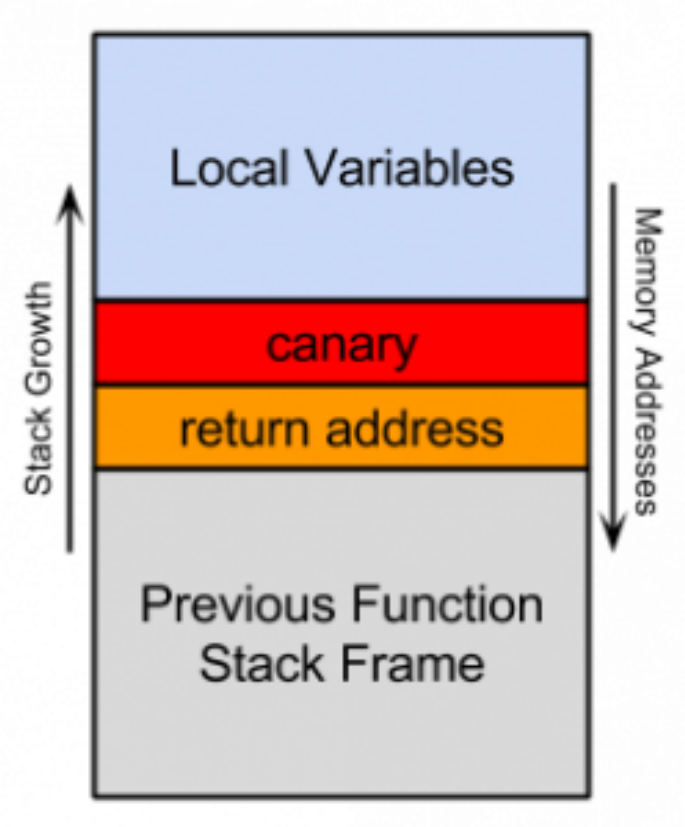
Return-oriented programming (ROP)

- Na prática não procuramos gatgets na mão
- https://github.com/JonathanSalwan/ROPgadget

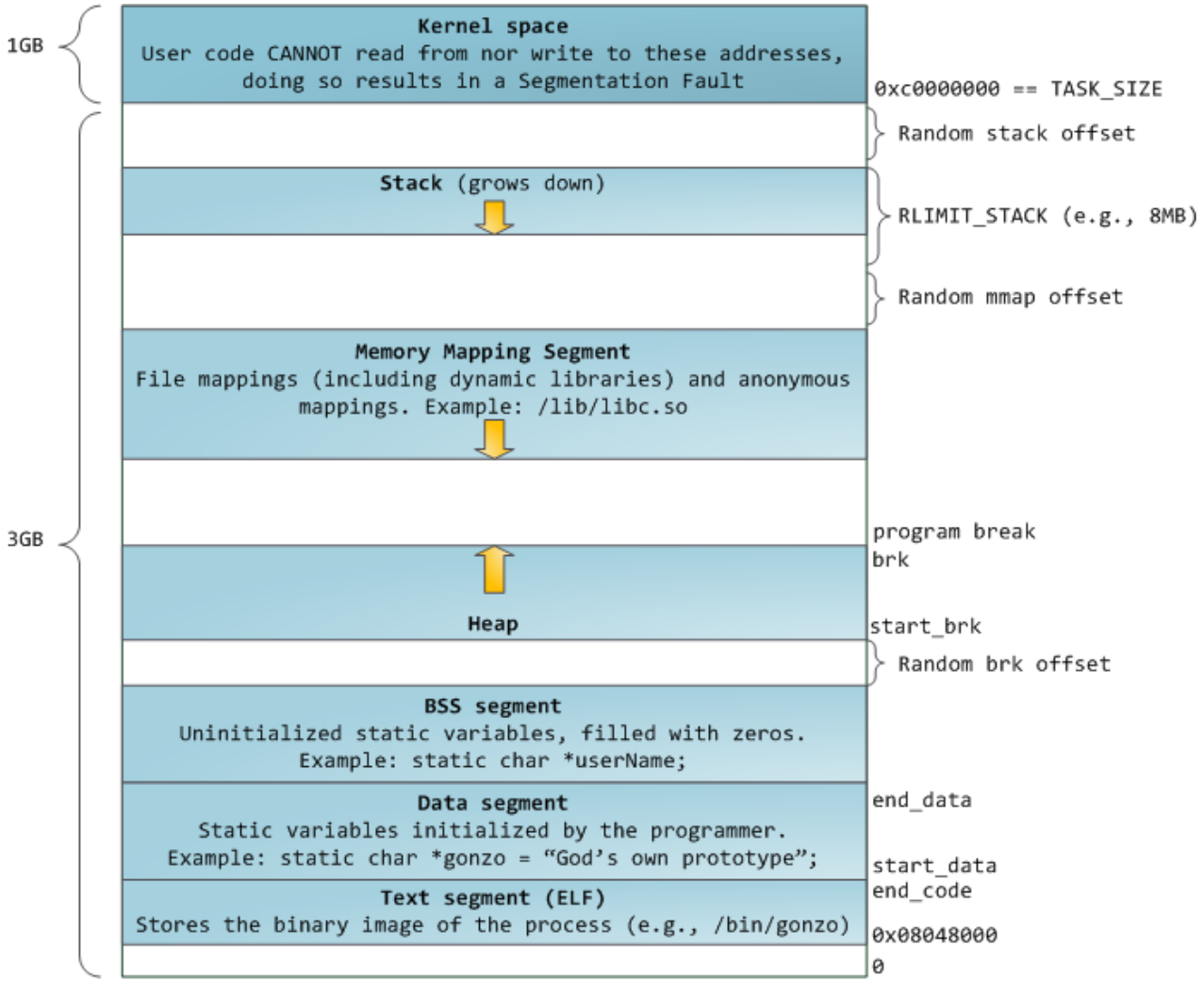
Defesas

- Canary
- NX
- PIE
- ASLR
- CET — Control-Flow Enforcement Technology
  - "Instrumentação de programas binários legados para compatibilização com Intel CET"

Canary


Local Variables
canary
return address
Previous Function
Stack Frame
Stack Growth
Memory Addresses

ASLR


1GB
3GB
Kernel space
User code CANNOT read from nor write to these addresses, doing so results in a Segmentation Fault
0xc0000000 == TASK_SIZE
Random stack offset
Stack (grows down)
RLIMIT_STACK (e.g., 8MB)
Random mmap offset
Memory Mapping Segment
File mappings (including dynamic libraries) and anonymous mappings. Example: /lib/libc.so
program break
brk
Heap
start_brk
Random brk offset
BSS segment
Uninitialized static variables, filled with zeros.
Example: static char *userName;
Data segment
Static variables initialized by the programmer.
Example: static char *gonzo = "God's own prototype";
end_data
start_data
Text segment (ELF)
Stores the binary image of the process (e.g., /bin/gonzo)
end_code
0x08048000
0

Indirect Branch Tracking