SORBONNE UNIVERSITÉ

Application d'Intel Sub-Page Permission à la sécurité de la mémoire

Master 2ème année, Systèmes et Applications Répartis

Présenté par:
Yves KONE

Encadré par:
Alain TCHANA - ENS Lyon
Pierre OLIVIER - Manchester University
Stella BITCHEBE - ENS Lyon

10 Septembre 2020

Introduction

Mémoire: ressource sensible

- 70% des bugs pour Chrome et Microsoft
- Langages de programmation (C/C++)

Introduction

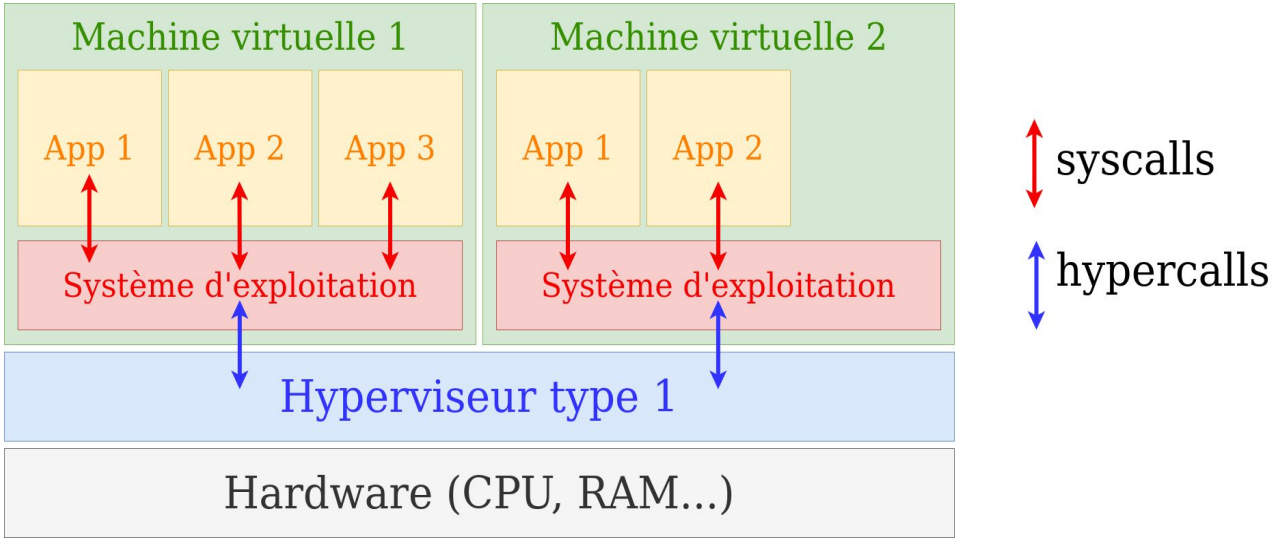
Mémoire: ressource sensible

- 70% des bugs pour Chrome et Microsoft
- Langages de programmation (C/C++)

Buffer overflow

Virtualisation

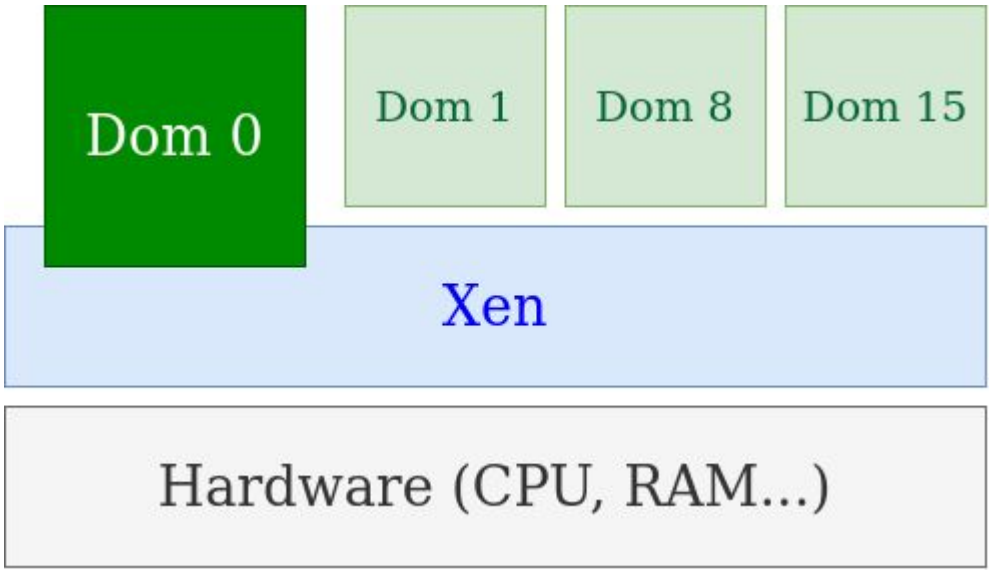
Virtualisation de type 1



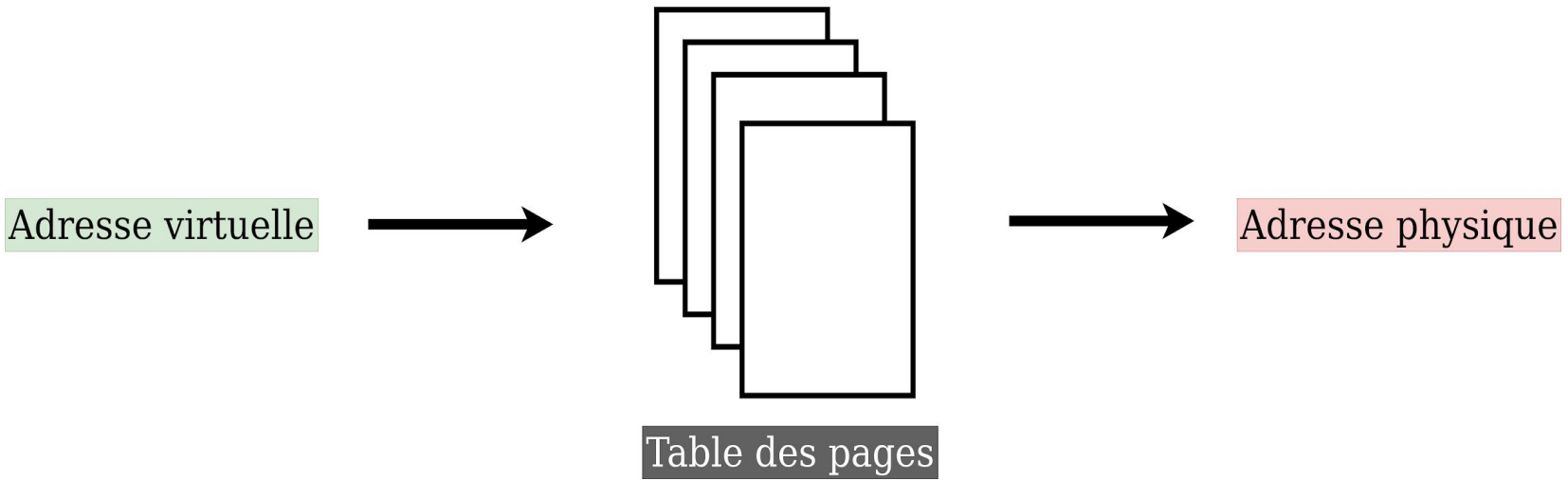
Xen

Très populaire (Amazone)

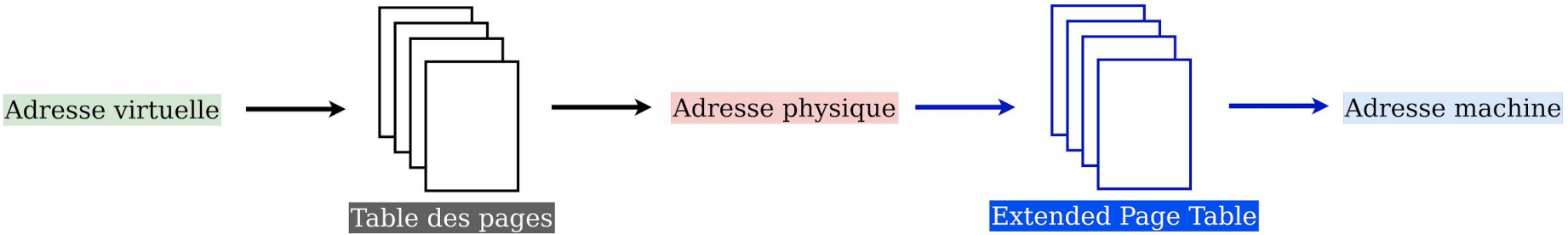
Open source



Mémoire



Mémoire

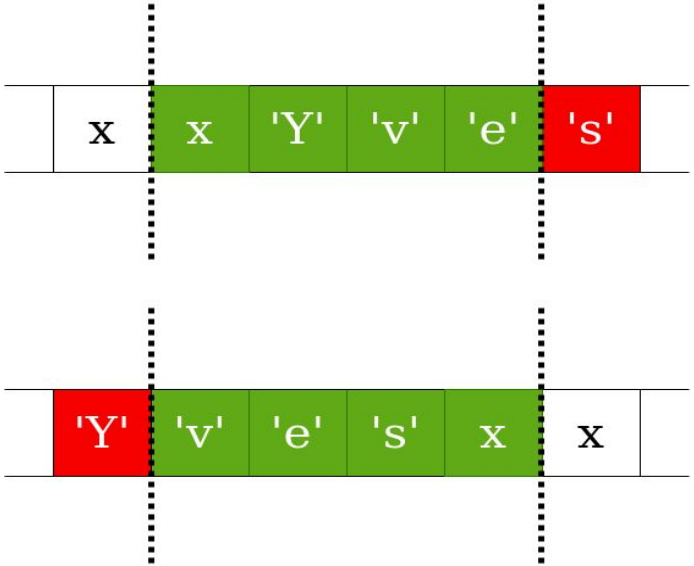

Adresse virtuelle
Table des pages
Adresse physique
Extended Page Table
Adresse machine

Buffer overflow

Buffer overflow

Accès extérieur au tampon

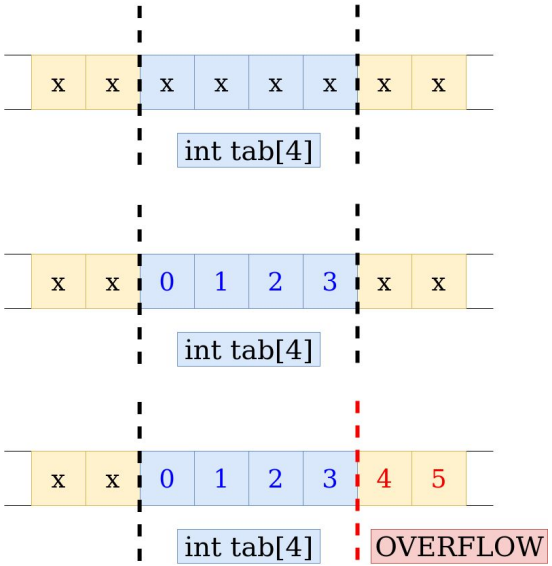
3 autres types de débordement



Débordements de tampon

```
int *tab = (int *) malloc(sizeof(int) * 4);

for (int i = 0; i < 6; i++) {
    tab[i] = i;
}
```

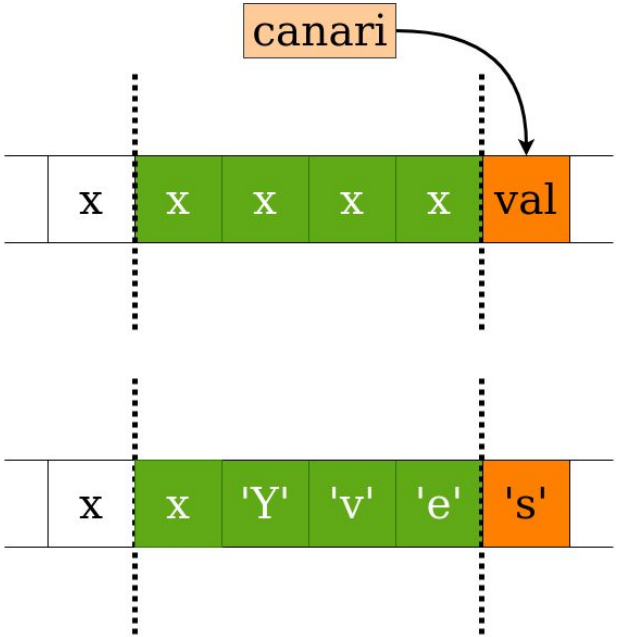


Solutions: deux approches

Canari

Barrière logicielle → 1 octet

Valeur magique recalculable



Canari

Avantage:

- surcoût mémoire (1 octet)

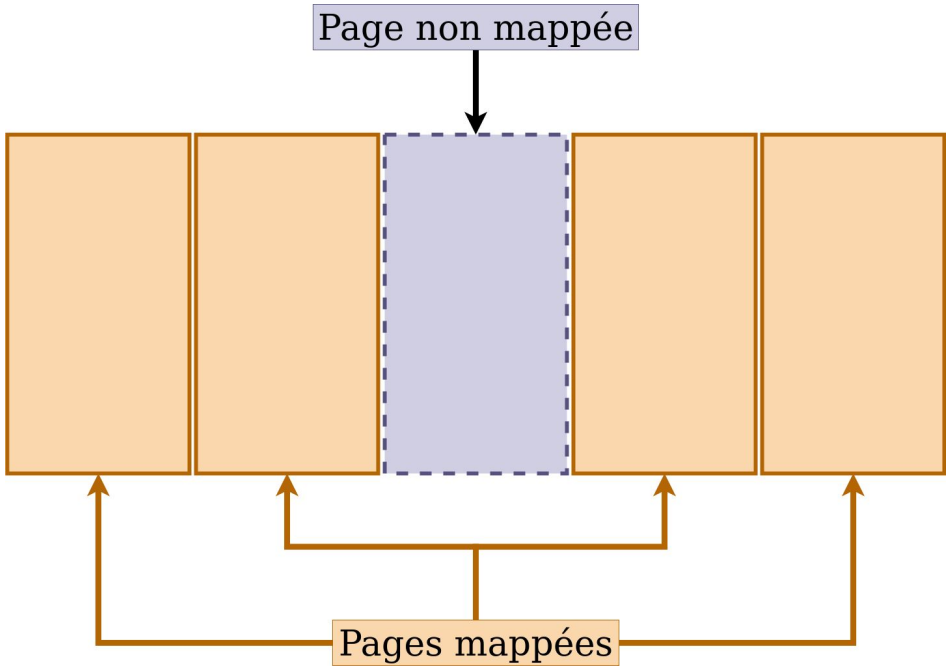
Inconvénient:

- détection asynchrone

Guardpage

Page non mappée

Ecriture → exception



Guardpage

Avantage:

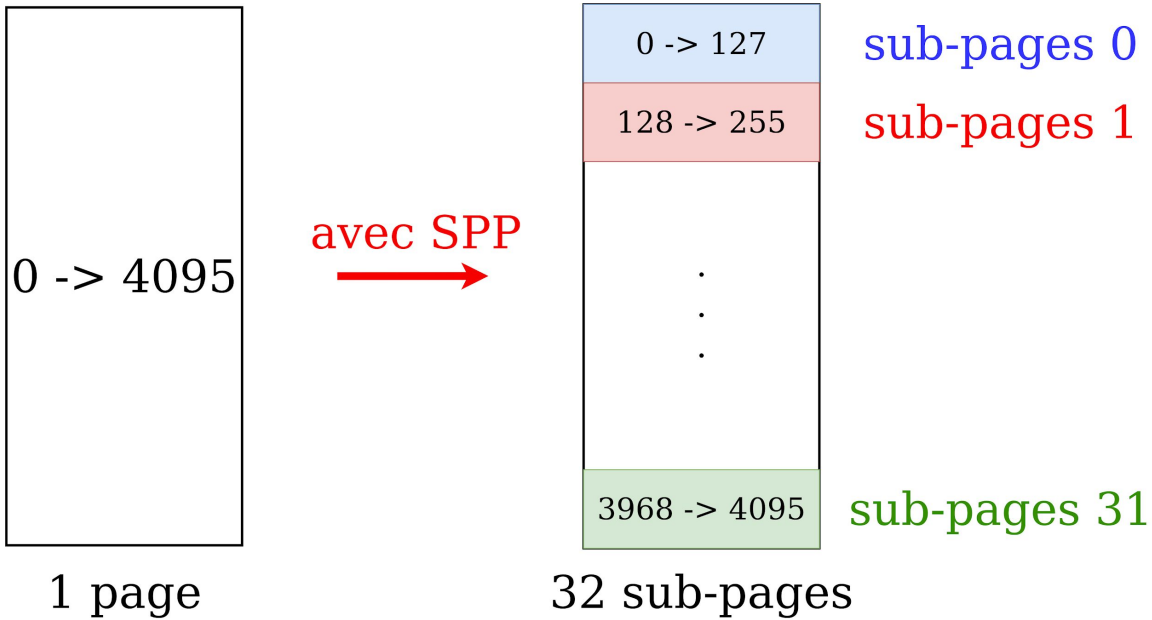
- détection synchrone

Inconvénient:

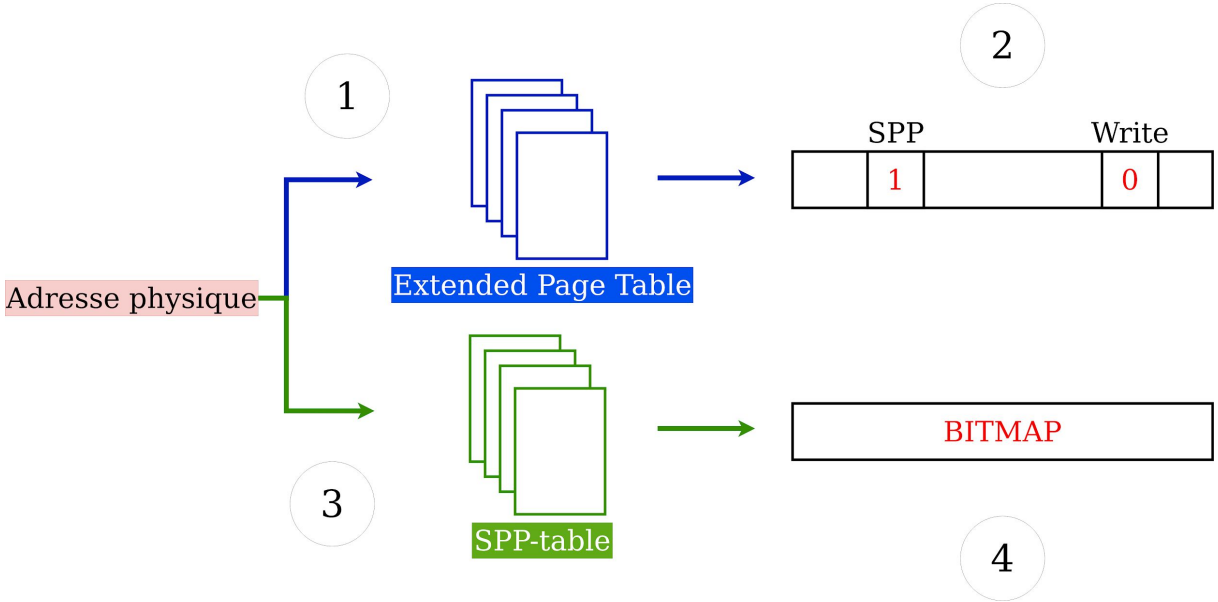
- gaspillage mémoire

Intel Sub-Page Permission

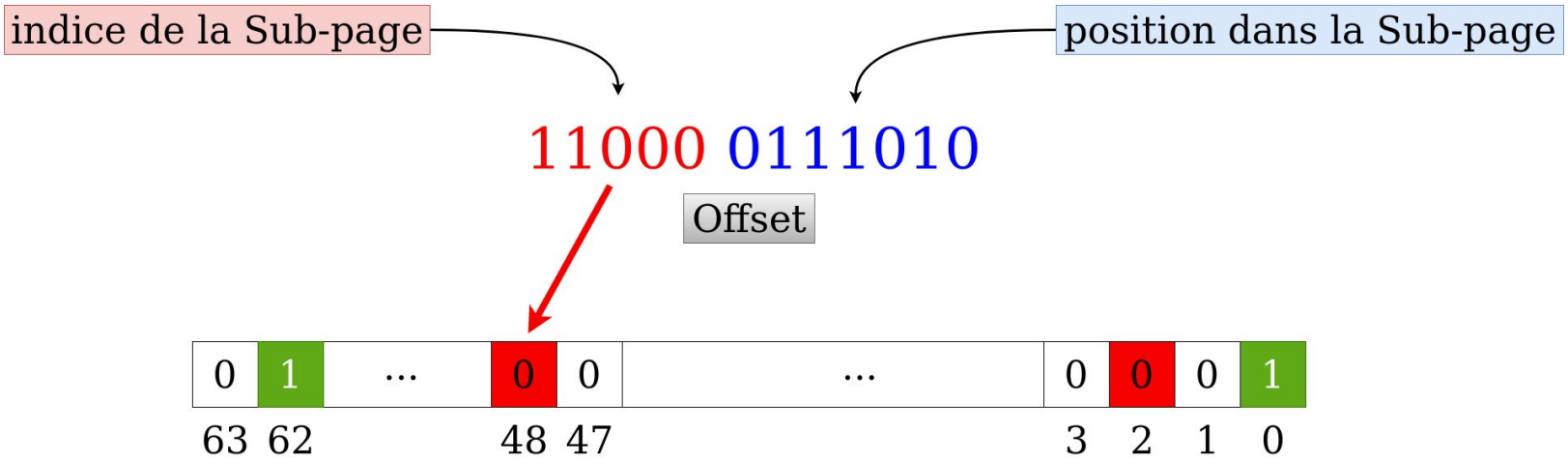
Intel Sub-Page Permission



SPP-table

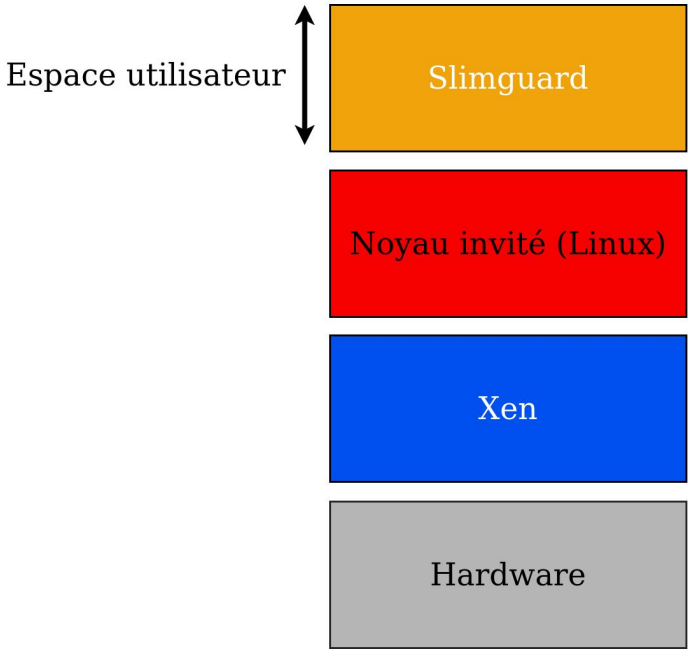

1
2
SPP
Write
1
0
Adresse physique
Extended Page Table
BITMAP
3
SPP-table
4

SPP-table


indice de la Sub-page
position dans la Sub-page
11000 0111010
Offset
0 1 … 0 0 … 0 0 0 1
63 62 48 47 3 2 1 0

Contribution

Contribution



Challenges

Protéger une sub-page

Emuler les accès pour SPP

Rendre les écritures effectives

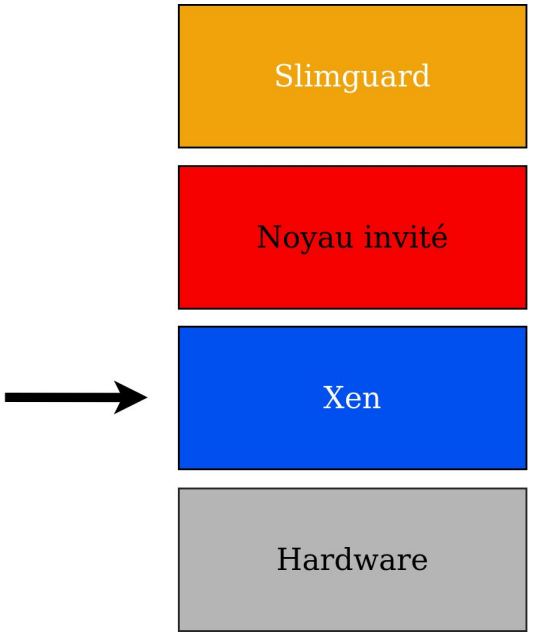
Préserver l’intégrité de la VM

Protection de sub-page

Hypercall:

1. domid
2. page physique
3. bitmap

Modification de l’EPT et SPP-table

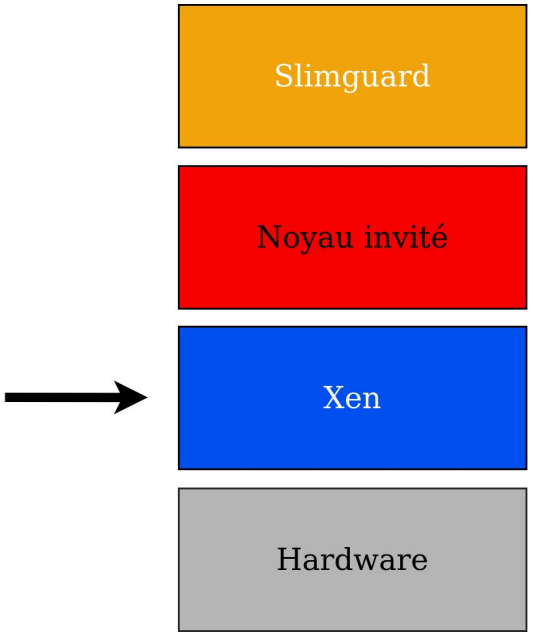


Protection de sub-page

Hypercall:

1. domid
2. page physique
3. sub-page

Modification de l'EPT et SPP-table

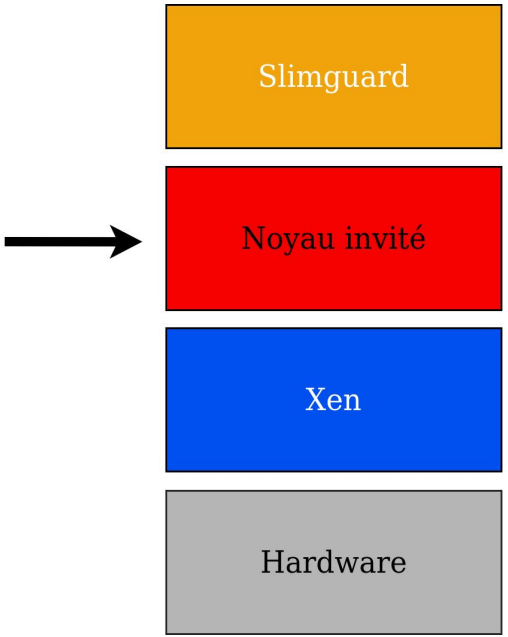


Protection de sub-page

Traduire l’adresse de la sub-page

Calculer le numéro de la sub-page

Modification de mprotect

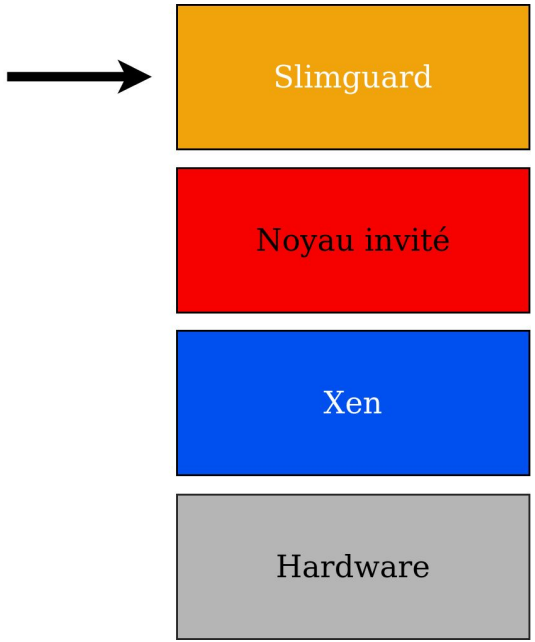


Protection de Sub-page

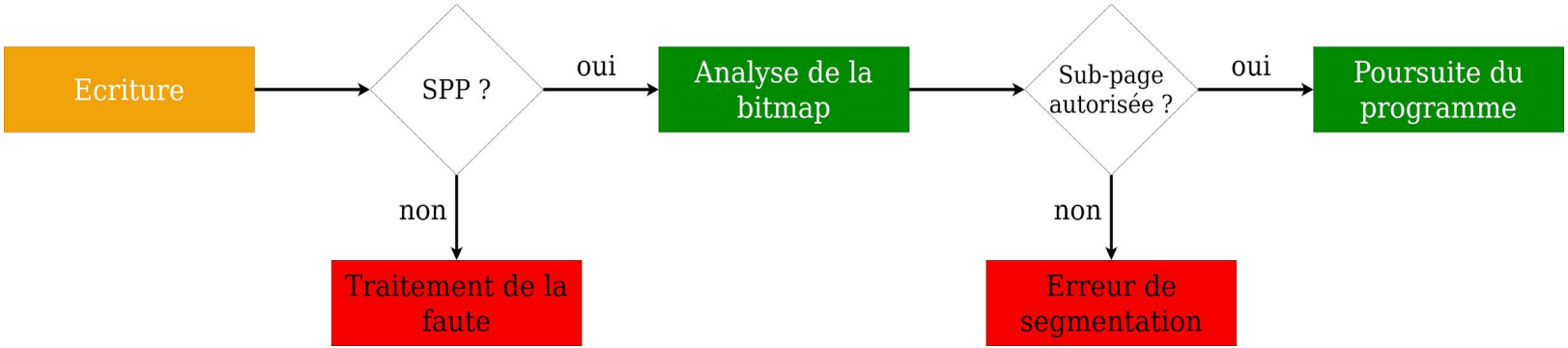
Fréquence de sub-page (SUB)

Calcul de l'adresse de la sub-page

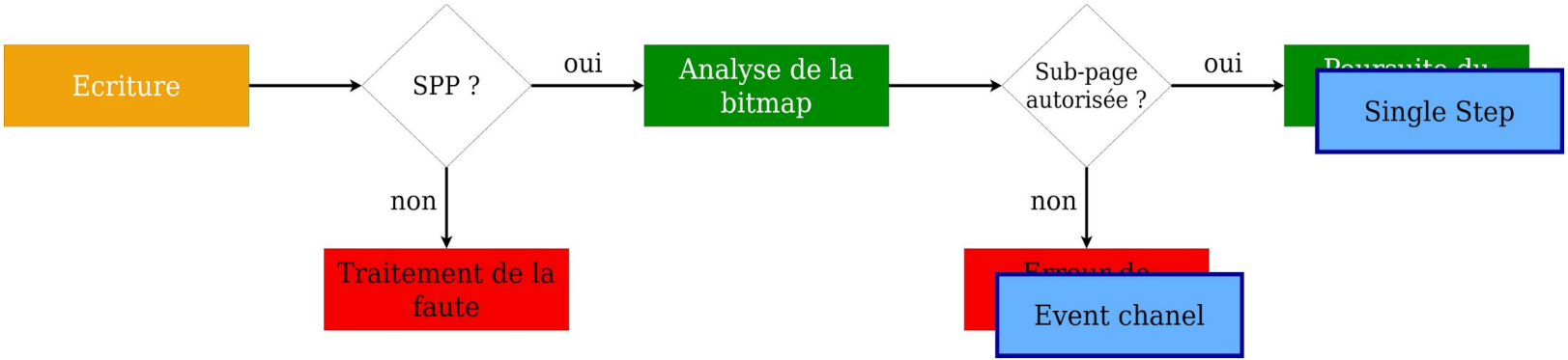
Appel à mprotect.



Emulation


Ecriture
SPP ?
oui
Analyse de la bitmap
Sub-page autorisée ?
oui
Poursuite du programme
non
Traitement de la faute
non
Erreur de segmentation

Emulation

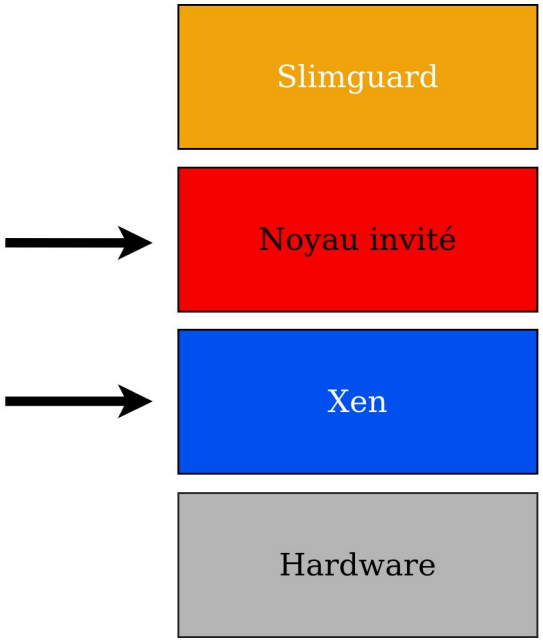

Ecriture
SPP ?
oui
Analyse de la bitmap
Sub-page autorisée ?
oui
Single Step
non
Traitement de la faute
non
Event chanel

Ecriture

Event channel:

Arrêt du programme

Événement → asynchrone



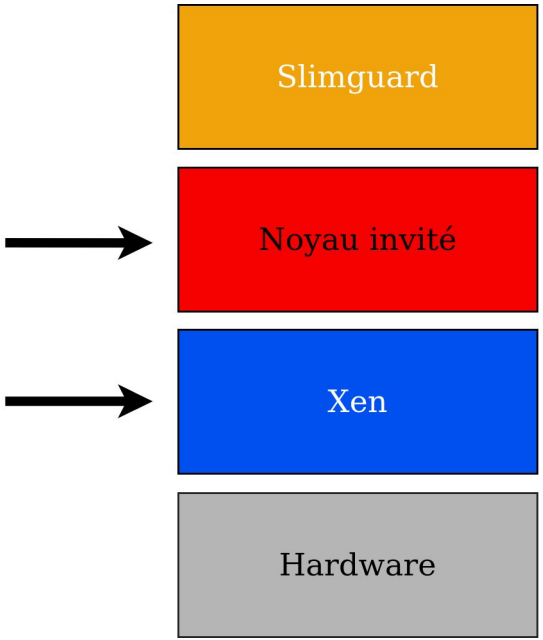
Ecriture

Single Step:

Modification de l'EPT → invalidation de la TLB

Exécution de l'écriture

Modification de l'EPT

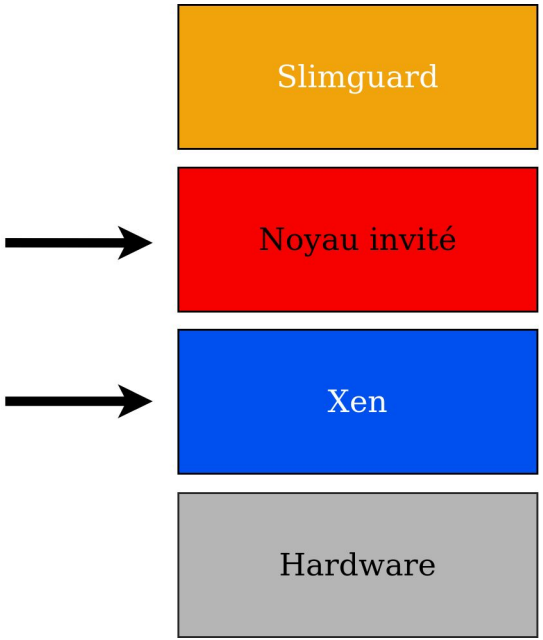


Libération des sub-pages

Modification de struct task_struct

Appel à mprotect → ajout dans la liste

Modification de l'EPT



Evaluation

Métriques

Surcoûts mémoire: /usr/time

Surcoûts CPU: nombre de syscalls et hypercalls

Variation de la fréquence de sub-page

Programmes de test

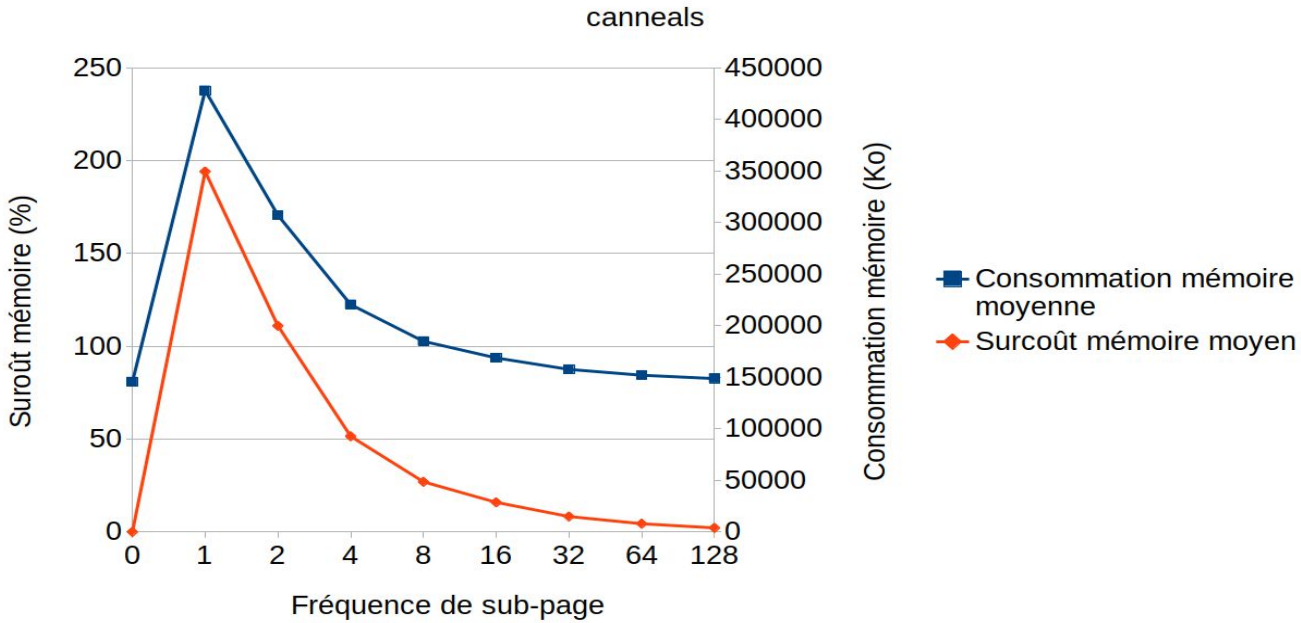
Micro-benchmarks:

10 000 allocations de tailles identiques (16, 512, 3072 octets)

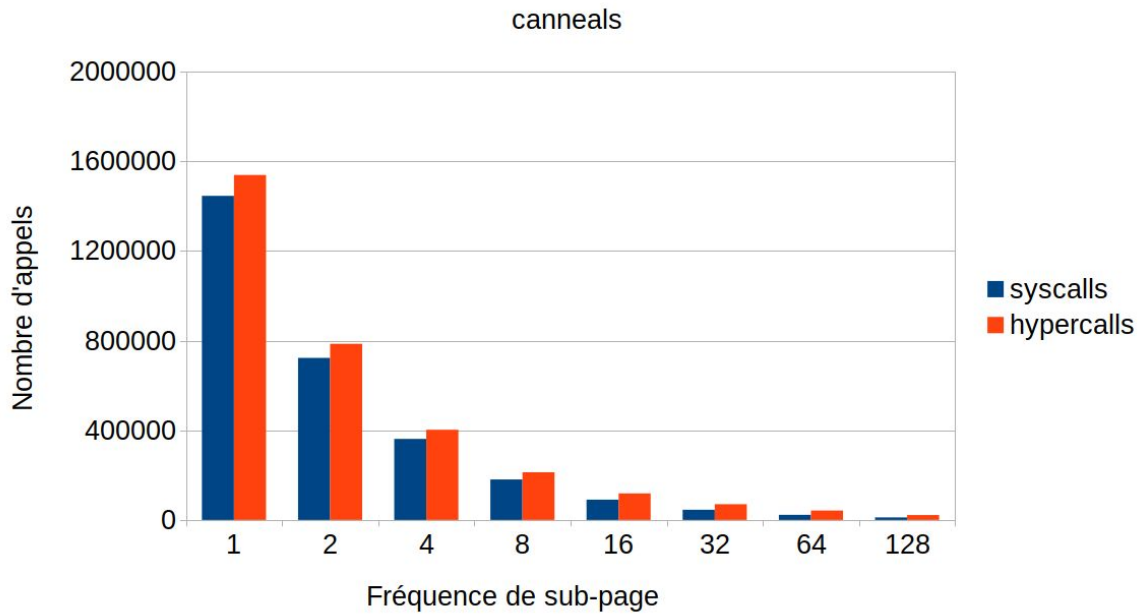
Macro-benchmarks:

parsec (Canneals, Freqmine, Blackscholes, Swaption)

Résultats


canneals
250
200
150
100
50
0
Suroût mémoire (%)
450000
400000
350000
300000
250000
200000
150000
100000
50000
0
Consommation mémoire (Ko)
0
1
2
4
8
16
32
64
128
Fréquence de sub-page
Consommation mémoire moyenne
Surcoût mémoire moyen

Résultats


canneals
2000000
1600000
1200000
800000
400000
0
Nombre d'appels
1
2
4
8
16
32
64
128
Fréquence de sub-page
syscalls
hypercalls

Conclusion

Perspectives

Optimisation:

- partager la SPP-table

Extension:

- accès en lecture

Références

Microsoft: 70 percent of all security bugs are memory safety issues, de Catalin Cimpanu, dans Zero Day, le 11 Février 2019
https://www.zdnet.com/article/microsoft-70-percent-of-all-security-bugs-are-memory-safety-issues/#:~:text=Around%2070%20percent%20of%20all,week%20at%20a%20security%20conference.&text=Users%20who%20often%20read%20vulnerability,terms%20over%20and%20over%20again.

70% of security bugs are memory safety problems: Chrome, de Hi-Tech, dans News, le 26 Mai 2020
https://tech.hindustantimes.com/tech/news/70-of-security-bugs-are-memory-safety-problems-chrome-71590483623525.html#:~:text=Nearly%2070%25%20of%20the%20high,management%20and%20safety%20related%20bugs.

SlimGuard: A Secure and Memory-Efficient Heap Allocator, Beichen Liu, Pierre Olivier, and Binoy Ravindran. Middleware '19:Middleware '19: 20th International Middleware Conference, December 8–13, 2019, Davis, CA, USA.
https://doi.org/10.1145/3361525.3361532

Références

Intel EPT-Based Sub-page Write Protection Support, Zhang Yi
https://lists.xenproject.org/archives/html/xen-devel/2017-10/msg02215.html

Agile Paging: Exceeding the Best of Nested and Shadow Paging, Jayneel Gandhi, Mark D.Hill, Michael M.Swift. ACM SIGARCH Computer Architecture News, Juin 2016.
https://doi.org/10.1145/3007787.3001212

Xen hypervisor,
https://xenproject.org/

Références

FreeGuard: A Faster Secure Heap Allocator, Sam Silvestro, Hongyu Liu, Corey Crosser, Zhiqiang Lin, Tongping Liu. Conference on Computer and Communications Security (CCS'17). October 2017. Pages 2389–2403.
https://doi.org/10.1145/3133956.3133957

Guarder: A Tunable Secure Allocator, Sam Silvestro, Hongyu Liu, Tianyi Liu, Zhiqiang Lin, Tongping Liu. This paper is included in the Proceedings of the 27th USENIX Security Symposium.August 15–17, 2018. Baltimore, MD, USA.

NMV: Virtualisation système, Gauthier Voron.
https://www.gauthiervoron.net/teaching/upmc-nmv-cmvirt.pdf