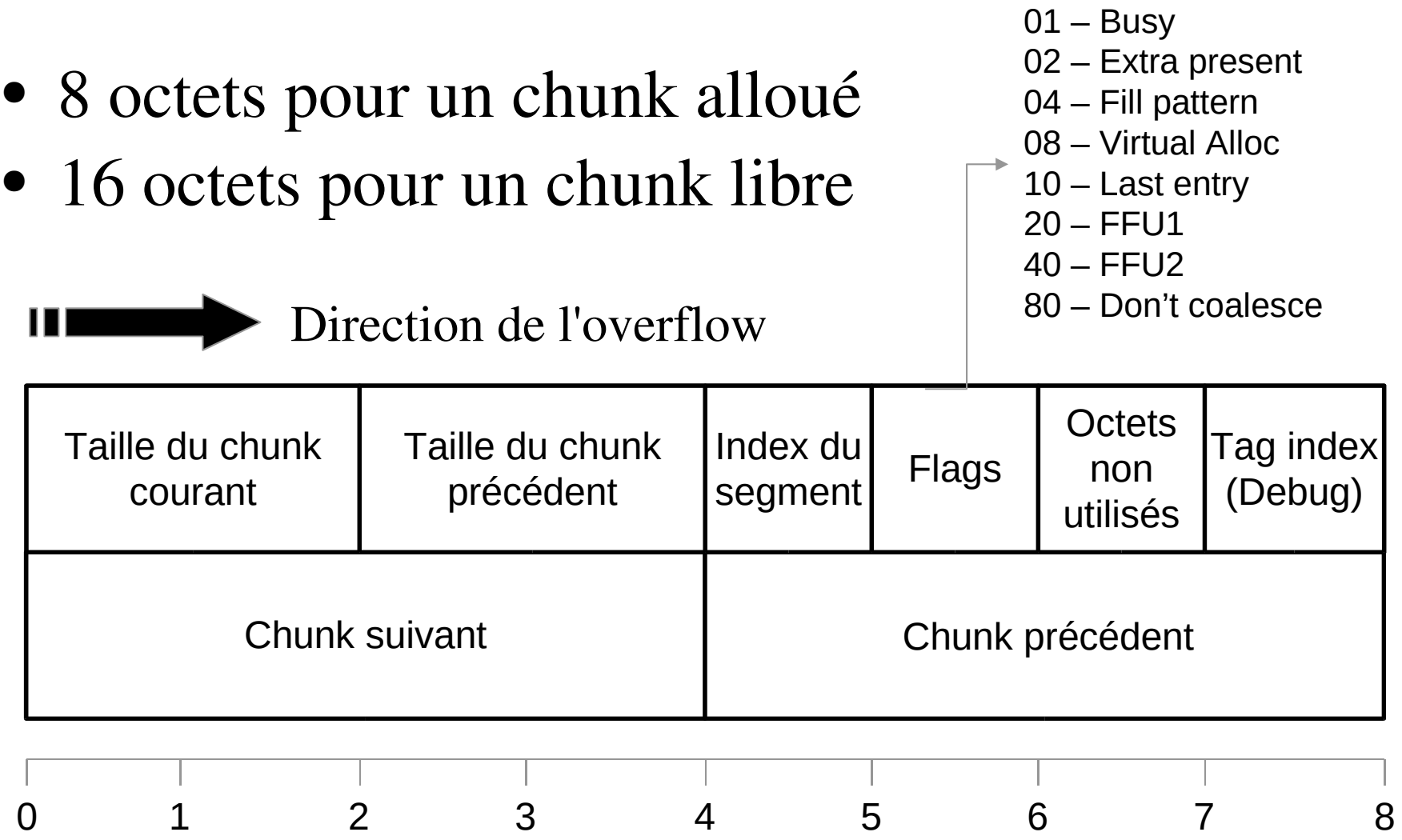
Exploitation fiable des heap overflows sous Windows

Kostya Kortchinsky
kostya.kortchinsky@renater.fr
Rump session SSTIC04

Basé sur le travail de Oded Horovitz et Matt Conover présenté à CanSecWest 2004

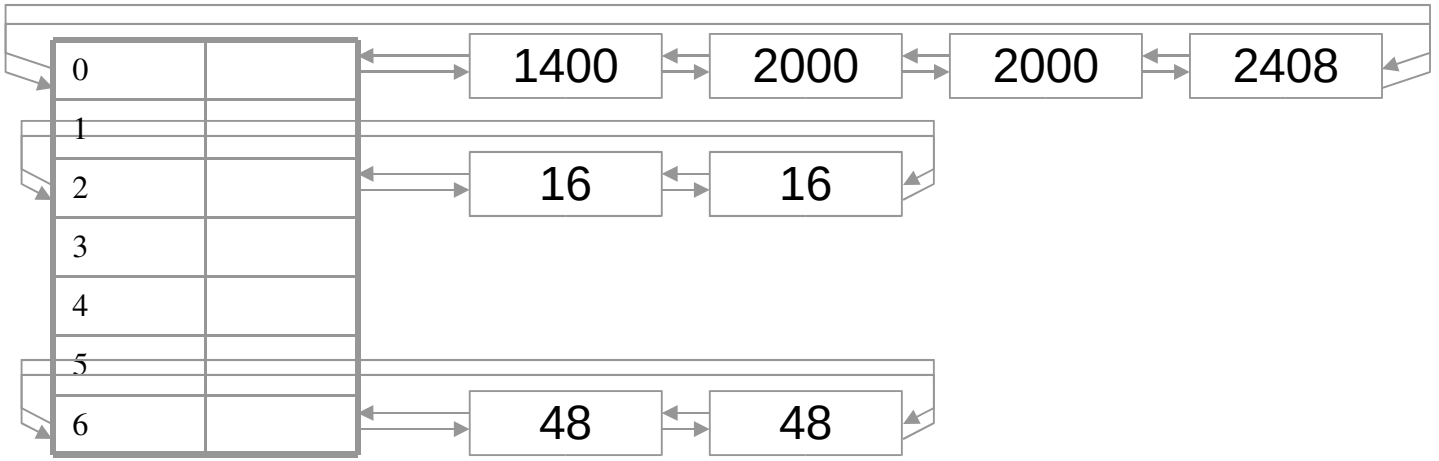
Structure d'un chunk

- 8 octets pour un chunk alloué
- 16 octets pour un chunk libre



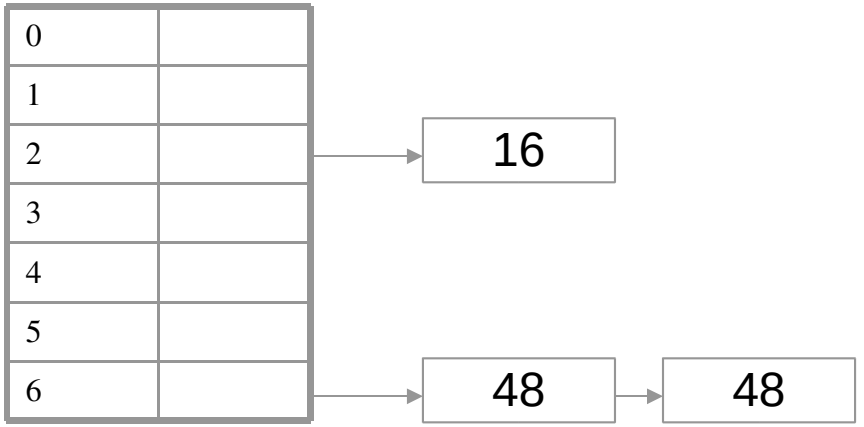
Les free lists

- 128 listes doublement chainées de chunks libres
- La taille d'un chunk correspond à l'index de la rangée * 8
- L'entrée 0 est une exception contenant les buffers dont 1024 < taille < “seuil d'allocation virtuelle”



La lookaside table

- La voie la plus rapide pour l'allocation et la libération
- Commence vide
- 128 listes simples de chunks occupés (flag busy)



Ecrasement de la mémoire

- Les couples populaires

Address A	Address B	Comments
Unhandled exception filter	Call [esi+xx] Or similar	High rate of success, but SP dependent
Vector Exception Handling	Stack location pointing to our buffer	High rate of success, but SP dependent
PEB Locks	Guessed address or application specific	Medium/ Low rate of success, SP Independent

Peut-on améliorer cela ?

Prise de contrôle de la lookaside table

- La lookaside table est la première option pouvant satisfaire une requête d'allocation ou de libération
- L'emplacement de la table est fixe relativement à la base du Heap, ainsi que la taille des entrées

En conséquence, on peut ...

- Envoyer un buffer d'une taille < 1024 : il sera libéré vers la lookaside table
- Provoquer un 4 byte overflow : l'adresse du buffer dans la lookaside table sera remplacée
- Envoyer un buffer de taille identique au premier : l'entrée dans la lookaside table sera utilisée ...