Introduction au débordement de tampon

Buffer OverFlow

Université Montpellier II, 2008

Sommaire

Historique non exhaustif

- 1972, *Computer Security Technology Planning Study* sort en page 61 : "This can be used to inject code into the monitor that will permit the user to seize control of the machine"
- 1988, *Inet Worm*[1] se répand et paralysa 10% d'Internet via l'exploitation d'un **Buffer OverFlow** dans le daemon "fingerd"
- 1996, *Smashing the Stack for Fun and Profit*[2] publié dans *Phrack 49* devient célèbre pour son introduction pas-à-pas de l'exploitation des débordements de tampons

Aujourd'hui, la plupart des attaques et tentatives d'attaque passent par l'exploitation de la mémoire

[1] Robert J.Morris
[2] Elias Levy dit *Aleph One*

Redéfinition d'un Buffer OverFlow (BOF)

"segmentation fault" ou "bus error"

- Survient lorsqu'un programme tente d'allouer en mémoire plus de données que l'espace reservé
- Ainsi le risque d'écraser une zone mémoire non dédiée est fort

Légitimité de la compréhension

- Primordial de **comprendre** les méchanismes liés à l'exploitation des BOF
  - **Au moins** pour les détecter
  - **Mieux** pour les éviter
- Les aspects légaux complexes et en (d)évolution sont **très importants**

- Pourquoi ?
  - mauvaises habitudes de programmation (i.e : *strcpy* ou *gets)*
  - pas de contrôle de la taille de chaine à copier
- Pour quoi ?
  - protéger et cacher un système de protection avancé
  - fins malveillantes (i.e : *priviledge escalation*)

Légitimité de la compréhension

- Primordial de **comprendre** les méchanismes liés à l'exploitation des BOF
  - **Au moins** pour les détecter
  - **Mieux** pour les éviter
- Les aspects légaux complexes et en (d)évolution sont **très importants**

- **Pourquoi ?**
  - mauvaises habitudes de programmation (i.e : *strcpy* ou *gets)*
  - pas de contrôle de la taille de chaine à copier
- Pour quoi ?
  - protéger et cacher un système de protection avancé
  - fins malveillantes (i.e : *priviledge escalation*)

Légitimité de la compréhension

- Primordial de **comprendre** les méchanismes liés à l'exploitation des BOF
  - **Au moins** pour les détecter
  - **Mieux** pour les éviter
- Les aspects légaux complexes et en (d)évolution sont **très importants**

- **Pourquoi ?**
  - mauvaises habitudes de programmation (i.e : *strcpy* ou *gets)*
  - pas de contrôle de la taille de chaine à copier
- **Pour quoi ?**
  - protéger et cacher un système de protection avancé
  - fins malveillantes (i.e : *priviledge escalation*)

Le format Executable Linking Format (ELF)

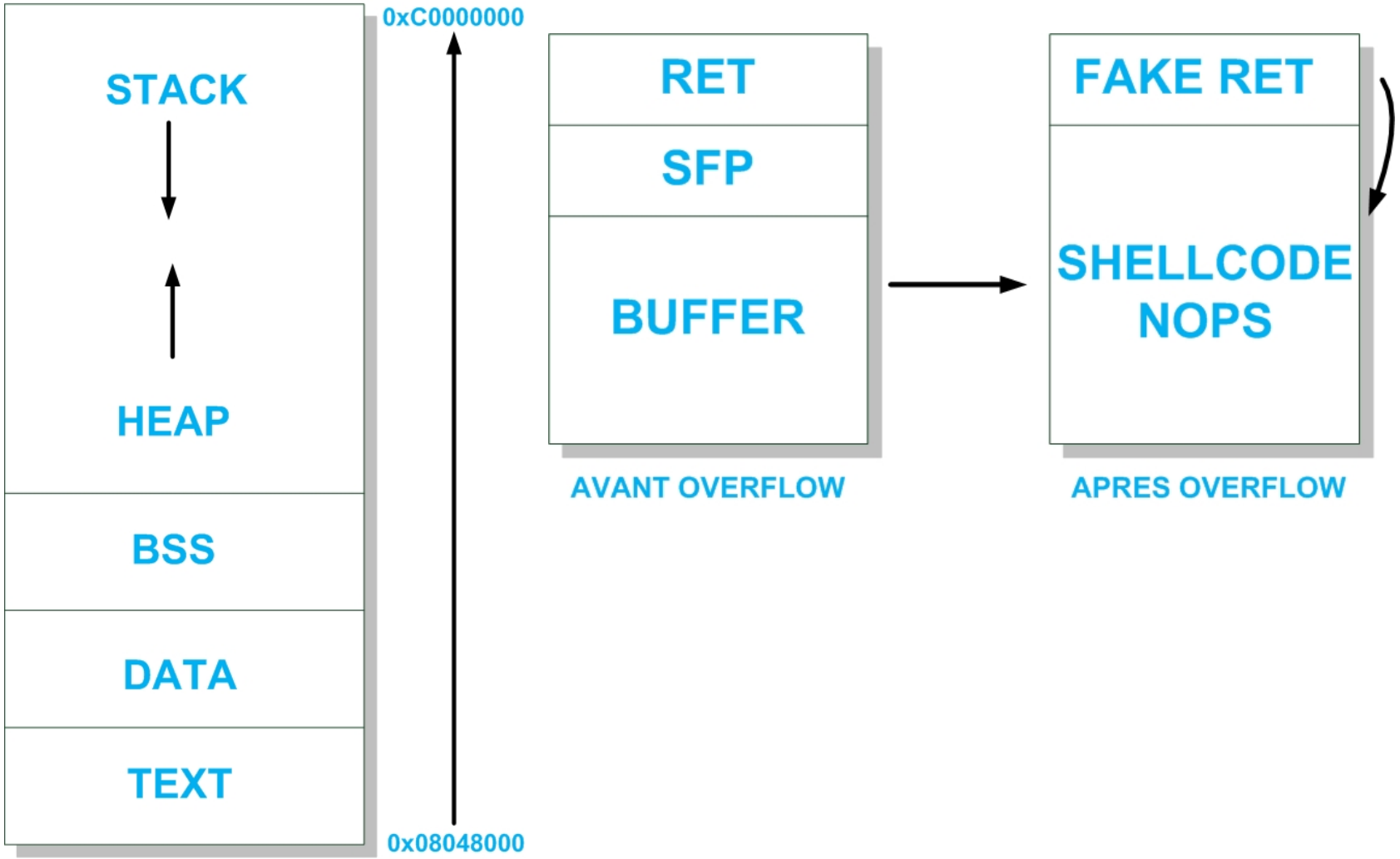
La plupart des systèmes d'exploitation partage la mémoire en deux espaces mémoires disjoints :

- L'espace utilisateur[3] s'étend de *0x00000000* à *0xBFFFFFFF*
- L'espace noyau[4] allant de *0xC0000000* à *0xFFFFFFFF*

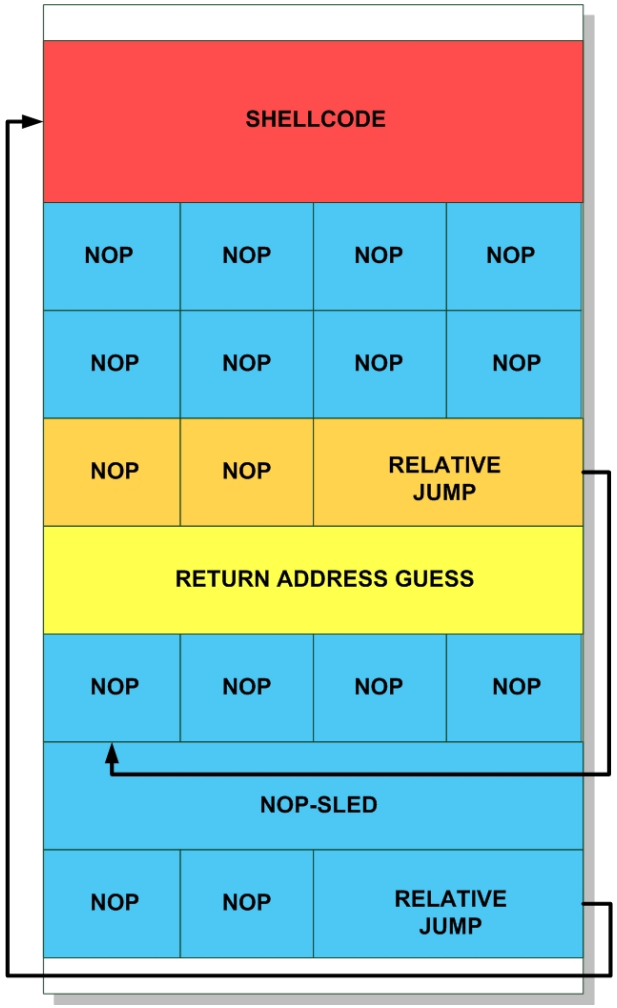
Le *Program Loader* (PL) crée une image processus à partir du fichier ELF executable

[3] User Land
[4] Kernel Land

Avant et après exploitation



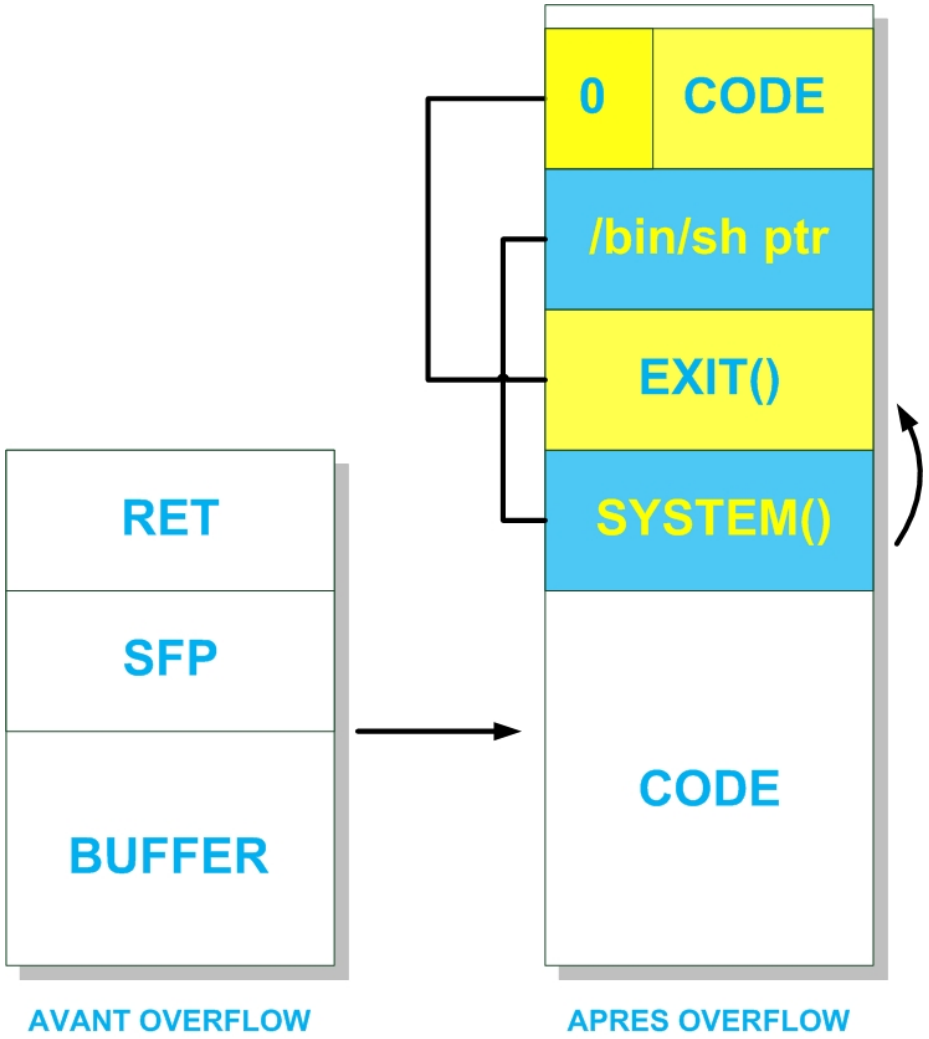
Mécanismes et utilisation du shellcode




```
u_char execshell[] =
   "\xeb\x24\x5e\x8d\x1e\x89\x5e\x0b\x33\xd2
                    \x89\x56\x07\x89\x56\x0f"
   "\xb8\x1b\x56\x34\x12\x35\x10\x56\x34\x12
                    \x8d\x4e\x0b\x8b\xd1\xcd"
 "\x80\x33\xc0\x40\xcd\x80\xe8\xd7\xff\xff\xff/bin/sh";
```

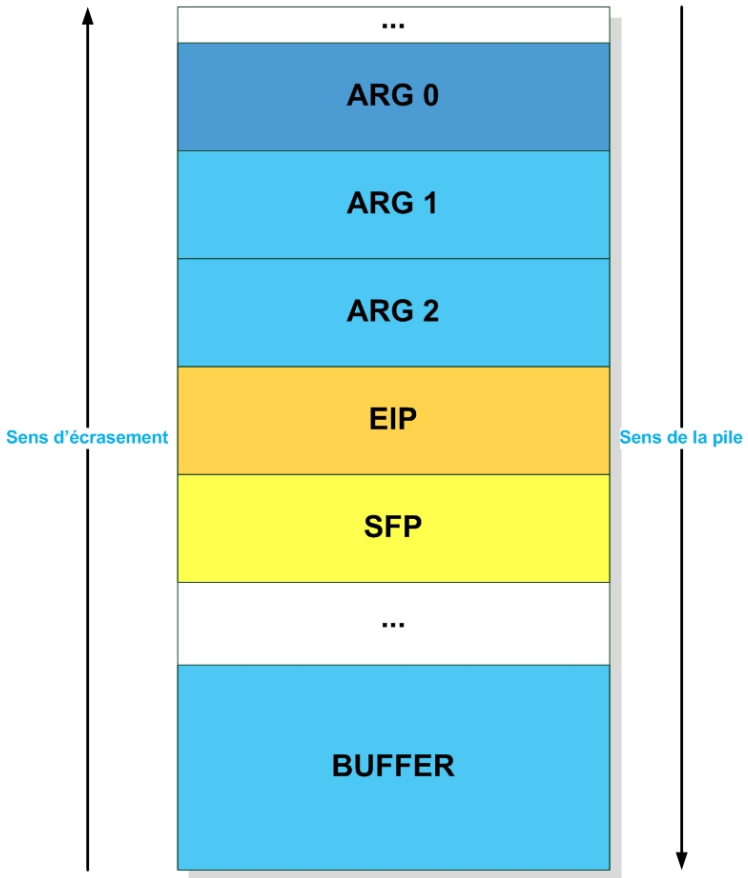
- pièce maîtresse
- dépend
  - de l'architecture
  - du système d'exploitation visé
  - du processus cible
- est constitué d'*opcode*

Méthode pseudo-générique



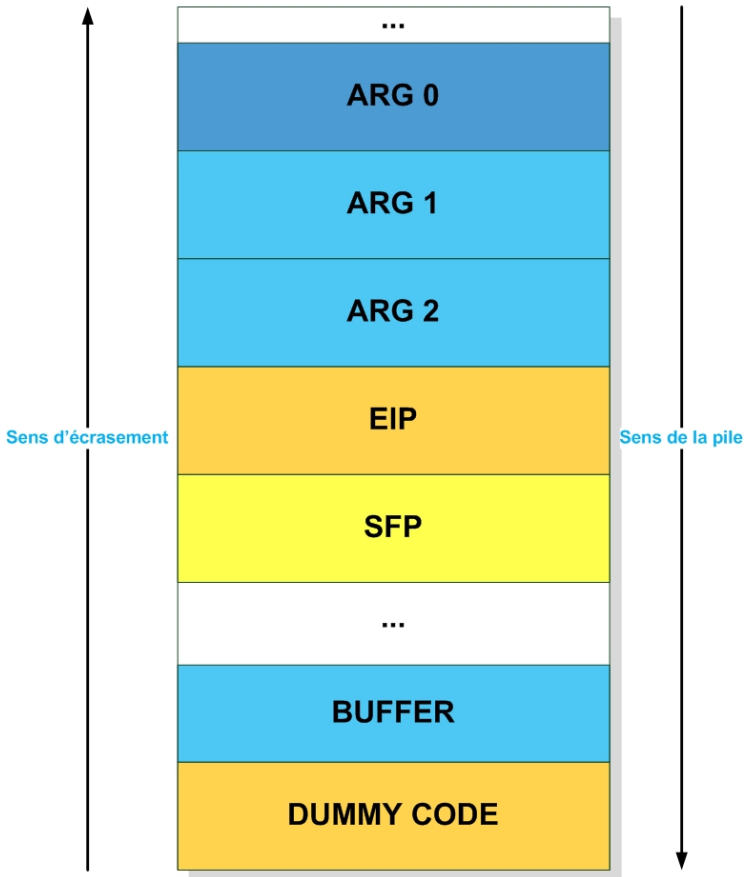
- S'**affranchir** de la protection offerte par la pile non éxecutable
- Au lieu de retourner le code situé dans la pile, la fonction vulnérable doit retourner dans une zone mémoire occupée par une librairie dynamique

Etat initial

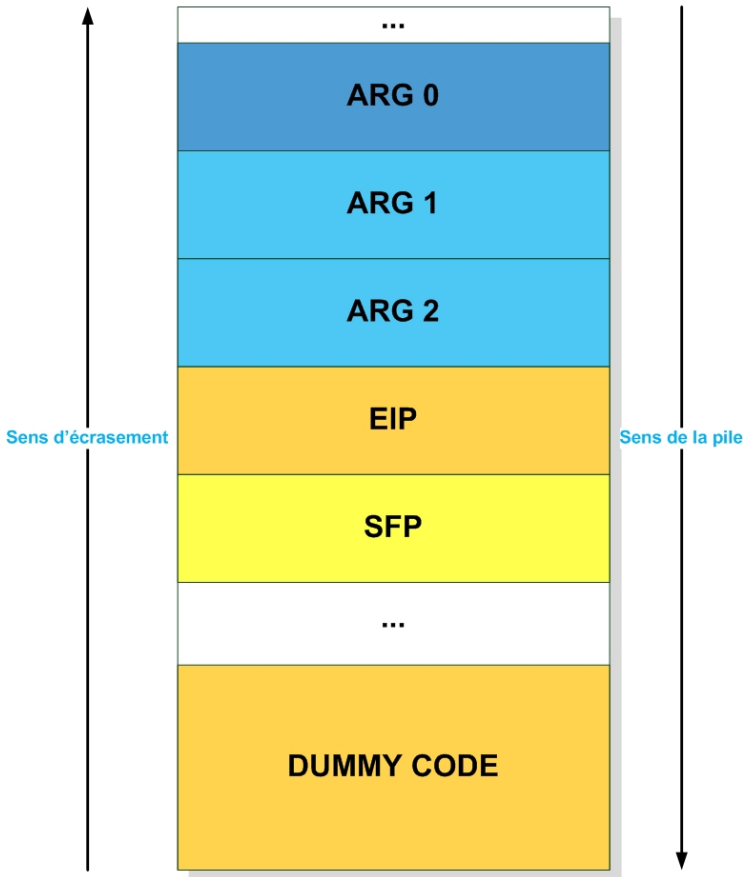


Le buffer est vide

Début de l'injection

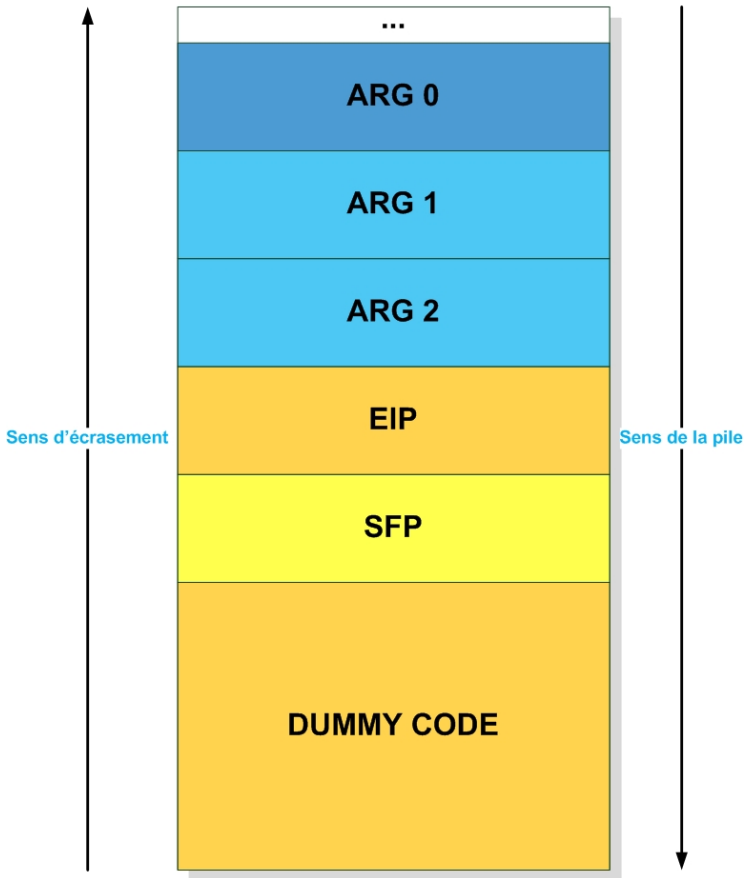


Le buffer n'est plus vide

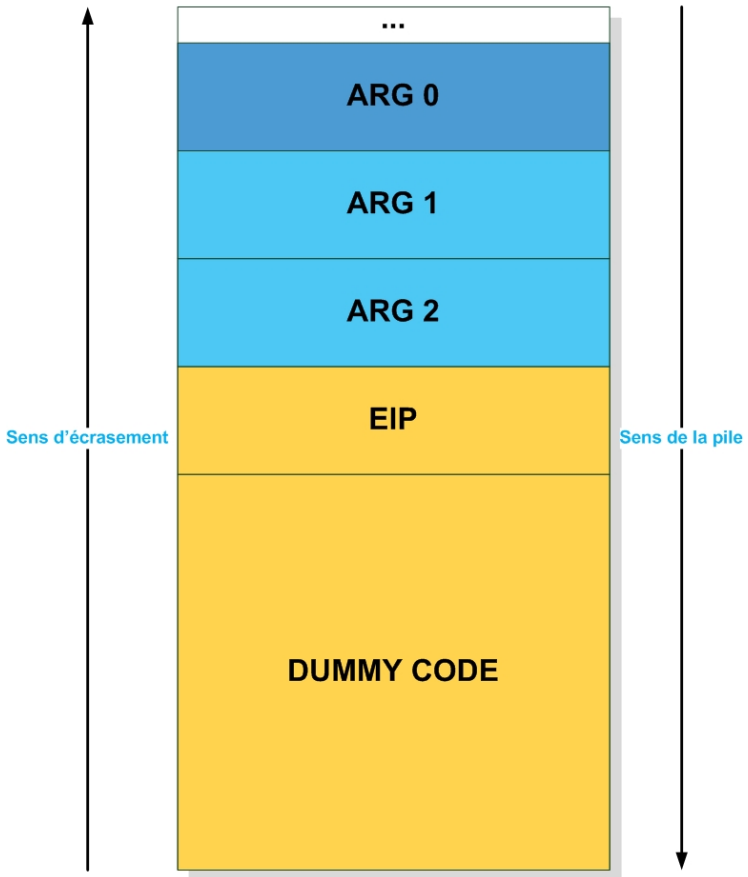


Le buffer est rempli

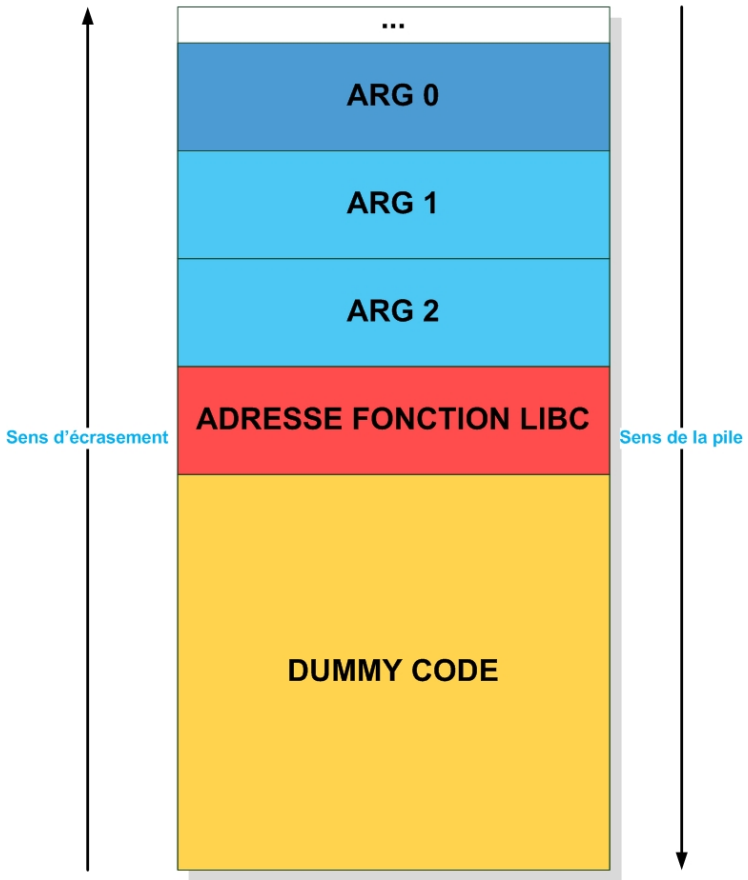
Buffer OverFlow



Le buffer déborde

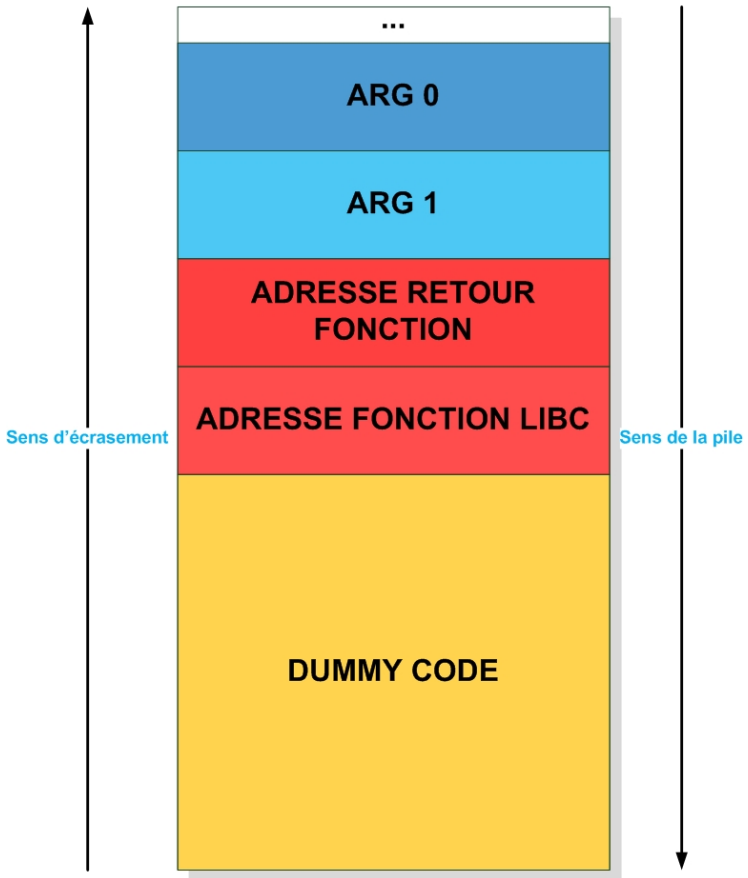


Ecrasement du **sfp**



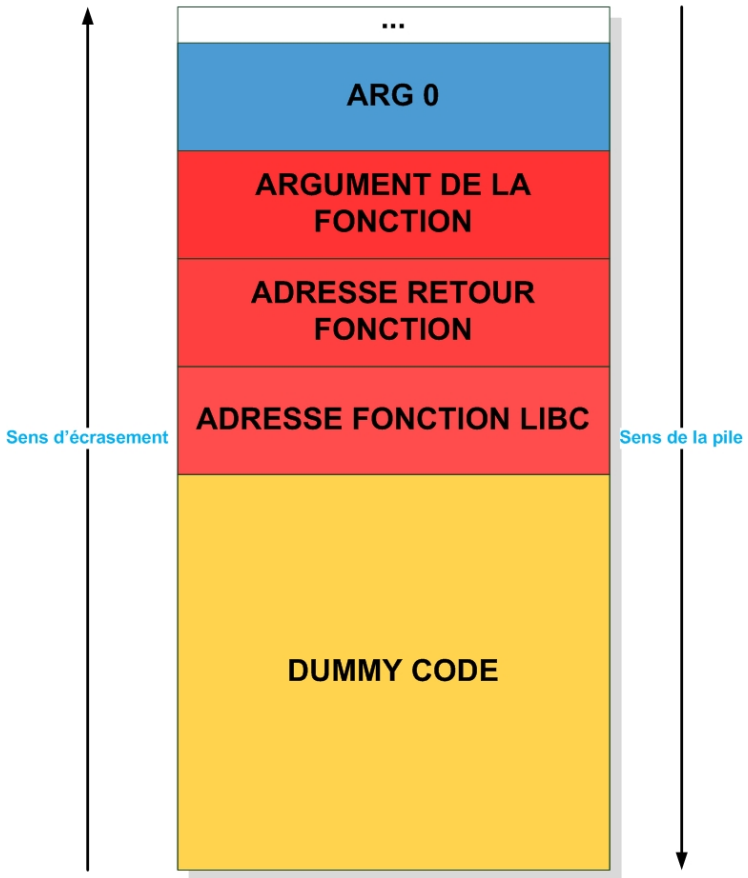
Ecrasement d'**eip**[a]

[a] i.e : par l'adresse de *system()*



Ecrasement de **ARG 2** par l'adresse de retour de la fonction[a]

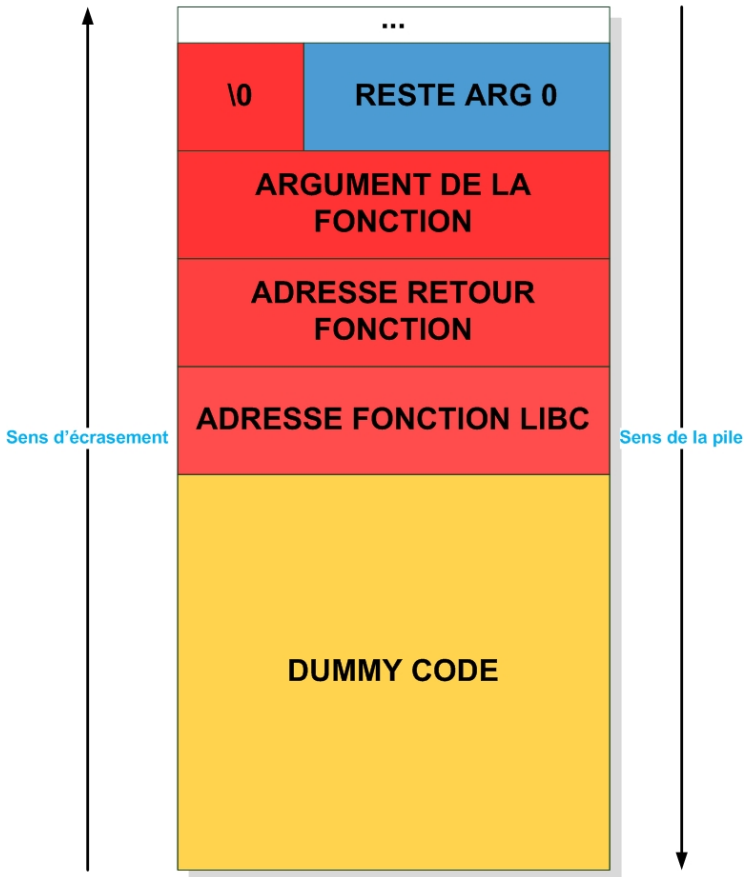
[a] i.e : l'adresse de *exit()*



Ecrasement de **ARG 1** par l'argument de la fonction[a]

[a] i.e : la chaine */bin/sh*

Etat final



Terminaison de chaine

Protections et Améliorations

Protections

randomization des adresses mémoires appliquée au chargement de la libc
injection d'un octet à 0 dans les adresses des fonctions de la libc

Améliorations

utilisation de variables globales
ret-into-libc chaînées

Jouer au chat et à la souris ?

- S'affranchir des contraintes d'une architecture
  - *little endian* vs *big endian*
  - l'option *stack non-exec* présente sur les architectures 64bits implique presque systématiquement une approche de type "ret-into-libc" ou "heap overflow"
  - Bruteforce pour la recherche des adresses en mémoire
  - utiliser le GPU ou un autre chipset que le CPU, y compris la mémoire associée
- **Automatiser l'exploitation** peut engendrer
  - des vers, virii, trojans, rootkits, botnets et autres menaces
  - un DDoS
  - le détournement (voire la compromission) d'un ou plusieurs systèmes d'information

À armes égales

- Un rootkit classique tourne au niveau noyau
- Un anti-rootkit classique tourne aussi au niveau noyau

- Un rootkit avancé utilise les mêmes systèmes de protection avancés qu'un anti-rootkit avancé
  - randomisation des adresses[5]
  - module kernel invisible (caché)
  - protection de son espace mémoire
  - prise d'information (*fingerprinting*)
  - discrétion (faible activité système et réseau) et hibernation
  - connaissance des anti-rootkit connus le rendant plus efficace

[5] *Address space layout randomization* (ASLR)

Ligne de conduite

Développer correctement

Mettre, tenir et se tenir à jour
Renforcer l'OS
Patcher le noyau
Surveiller et logguer

Sensibiliser

Solutions logicielles

W^X
PaX, {RSBAC|RBAC}, grsecurity
Exec Shield
Openwall
DPE[a] de MS

[a] Data Prevention Execution

coreImpact, gdb, metasploit, nagios, nessus, nmap, valgrind..

Pour aller un peu plus loin I

- http ://en.wikipedia.org/wiki/Buffer_overflow
- http ://en.wikipedia.org/wiki/Return-to-libc_attack
- http ://en.wikipedia.org/wiki/W%5EX
- http ://fr.wikipedia.org/wiki/Dépassement_de_tampon
- http ://ouah.org/
- http ://www.grsecurity.net/
- http ://www.ibm.com/developerworks/linux/library/l-sp4.html
- http ://www.securiteinfo.com/attaques/hacking/buff.shtml

Pour aller un peu plus loin II

- Autodafé : an Act of Software Torture, Martin Vuagnoux (EPFL)
- Exploitation avancée de Buffer OverFlows, Security and Cryptography Laboratory (LASEC)
- Inside the buffer overflow attack : Mechanism, Method & Preventin, Mark E.Donaldson (SANS Institute)
- Smashing The Stack For Fun And Profit, Aleph One - Phrack 49
- TP Sécurité des Réseaux