Sécurité des serveurs et des applications, Partie 2

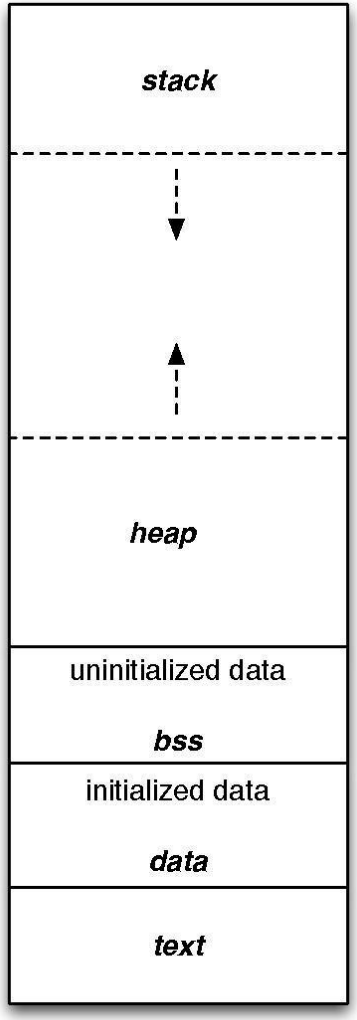
Sécurité des exécutables

Ce qu’on va voir

- Structure d’un exécutable
- Exécution d’un programme
- Attaques
- Contre-mesures

Structures d’un programme

- text segment : code à exécuter
- data segment : données statiques initialisées (par exemple : chaines de caractères)
- bss segment : données statiques non initialisées (variables “static” ou globales)
- heap : mémoire dynamiquement allouée (malloc, free)
- stack : mémoire statiquement allouée (variables locales, appel de fonctions)



Exemple

Exécution d’un programme

1. création des structures noyaux
   - thread
   - virtual memory
   - flux (stdin, stdout)
2. peuplement des segments du programme
   - argc, argv
   - variables d’environnement
3. chargement des librairies dynamiques

source : https://lwn.net/Articles/631631/

Un peu d’architecture x86

- 6 registres de calcul : EAX, EBX, ECX, EDX, ESI, EDI
- 2 registres de pile :
  - ESP : pointeur du haut de la pile
  - EBP : pointeur du bas de la pile
- 6 registres de segment : SS (stack), CS (code), DS (data), ES, FS, GS (extra data segments)
- EFLAGS register (32 boolean registers)
- EIP : pointeur vers la prochaine instruction

source : https://en.wikibooks.org/wiki/X86_Assembly/X86_Architecture

Un peu d’assembleur x86 (intel)

- mov
  - mov AX, BX
  - mov AX, 1 ou mov AX, a32b1h
  - mov AX, 0xdeadbeef
  - mov AX, [BX]
- push
- pop
- add, sub, etc

Convention d’appel de fonction (cdecl pour x86)

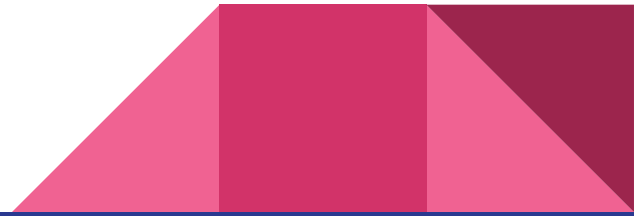
- call
  - sauvegarde de l’ancienne adresse de ebp : push EBP
  - création de la nouvelle fenêtre de pile : mov EBP, ESP
  - empilement des arguments (de droite à gauche) : push x_n, ..., push x_1
  - appel de fonction : call function (sauvegarde de EIP puis saut)
- ret
  - récupère la valeur de EIP et y saute
  - l’appelant doit nettoyer la pile

Exemple

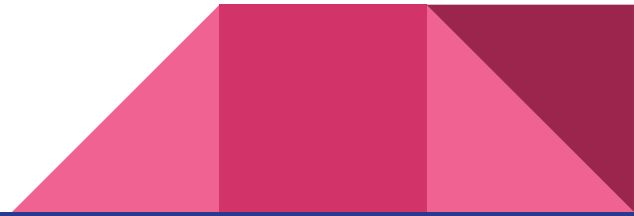
$LD_PRELOAD

```
$ ls
ma_lib.c ma_lib.h

$ gcc -c -fPIC ma_lib.c

$ gcc -shared ma_lib.o -o ma_lib.so

$ LD_PRELOAD=ma_lib.so un_executable
```

shellcode

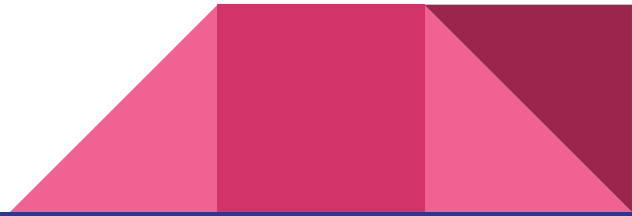


Buffer overflow



format strings

- %x : affiche un unsigned int
- %s : affiche une chaîne de caractères (s’arrête de lire quand 0x00)
- %n : sauvegarde le nombre de caractères écrits jusqu’à présent



Contre-mesures

- NX
- ASLR (adress space layout randomization)
- canari