

Rétro-ingénierie pour tous !
Le pouvoir ne se donne pas, il se prend.

Julien (jvoisin) Voisin

- Tails et OTR
- Nos-oignons
- Radare2
- Stagiaire chez NBS-system

Quizz

- Compilateur

Quizz

- Compilateur
- Assembleur

Quizz

- Compilateur
- Assembleur
- CTF

Quizz

- Compilateur
- Assembleur
- CTF
- Firmware



La rétro-ingénierie, c'est quoi ?

Qu'est-ce que la rétro-ingénierie ?



La **rétro-ingénierie** consiste à étudier un objet pour en déterminer son fonctionnement interne.

Qu'est-ce que la rétro-ingénierie ?



La **rétro-ingénierie** consiste à étudier un logiciel pour en déterminer son fonctionnement interne.

C'est quoi un programme ?

- code source
 - ordinateur
 - programmeur
- } programme

Code source

```
void  
main(int argc, char** argv){  
    if (argc == 3) {  
        puts("MiNET");  
    }  
}
```

Assembleur

```
lea ecx, [esp + 4]
and esp, 0xffffffff0
push dword [ecx - 4]
push ebp
mov ebp, esp
push ecx
push eax
cmp dword [ecx], 3
jne 0x8048344
sub esp, 0xc
push 0x80484d0
call 0x080482ec
add esp, 0x10
mov ecx, dword [ebp - 4]
leave
lea esp, [ecx - 4]
ret
```

Code machine

```
- offset -  0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
0x08048320 8d4c 2404 83e4 f0ff 71fc 5589 e551 5083  .L$.....q.U..QP.
0x08048330 3903 7510 83ec 0c68 d084 0408 e8af ffff  9.u....h.....
0x08048340 ff83 c410 8b4d fcc9 8d61 fcc3  ....M...a..
```

Comment ça marche un programme?

```
gdb-peda$ r
Starting program: /home/jvoisin/prez/SudTelecom/pwn/a.out
[-----registers-----]
EAX: 0x1 (b'\x01')
EBX: 0xf7faf000 --> 0x1a8da8
ECX: 0xffffd110 ("\001")
EDX: 0xffffd134 --> 0xf7faf000 --> 0x1a8da8
ESI: 0x0
EDI: 0x0
EBP: 0xffffd0f8 (")
ESP: 0xffffd0f4 --> 0xffffd110 ("\001")
EIP: 0x80484ca (<main+14>:      sub      esp,0x4)
[-----code-----]
0x80484c6 <main+10>: push    ebp
0x80484c7 <main+11>: mov     ebp,esp
0x80484c9 <main+13>: push    ecx
=> 0x80484ca <main+14>: sub     esp,0x4
0x80484cd <main+17>: mov     eax,ecx
0x80484cf <main+19>: cmp     DWORD PTR [eax],0x1
0x80484d2 <main+22>: jg      0x80484db <main+31>
0x80484d4 <main+24>: mov     eax,0x1
[-----stack-----]
0000| 0xffffd0f4 --> 0xffffd110 ("\001")
0004| 0xffffd0f8 (")
0008| 0xffffd0fc --> 0xf7e1fa83 (<__libc_start_main+243>:      mov     DWORD PTR [esp],eax)
0012| 0xffffd100 --> 0x8048500 (<__libc_csu_init>:      push    ebp)
0016| 0xffffd104 (")
0020| 0xffffd108 (")
0024| 0xffffd10c --> 0xf7e1fa83 (<__libc_start_main+243>:      mov     DWORD PTR [esp],eax)
0028| 0xffffd110 ("\001")
[-----]
Legend: code, data, rodata, value

Breakpoint 1, 0x80484ca in main ()
gdb-peda$
```

Une transformation à sens unique



La compilation n'est ni **injective** ni **surjective**.

Une transformation à sens unique



La compilation n'est ni **injective** ni **surjective**.

- Un code source peut être compilé d'une infinité de manières

Une transformation à sens unique

La compilation n'est ni **injective** ni **surjective**.

- Un code source peut être compilé d'une infinité de manières
- Tous les codes sources ne peuvent être compilés

Une transformation à sens unique

La compilation n'est ni **injective** ni **surjective**.

- Un code source peut être compilé d'une infinité de manières
- Tous les codes sources ne peuvent être compilés
- Un binaire peut être décompilé d'une infinité de manières

Une transformation à sens unique

La compilation n'est ni **injective** ni **surjective**.

- Un code source peut être compilé d'une infinité de manières
- Tous les codes sources ne peuvent être compilés
- Un binaire peut être décompilé d'une infinité de manières
- Tous les binaires ne peuvent être décompilés

Une transformation à sens unique

La compilation n'est ni **injective** ni **surjective**.

- Un code source peut être compilé d'une infinité de manières
- Tous les codes sources ne peuvent être compilés
- Un binaire peut être décompilé d'une infinité de manières
- Tous les binaires ne peuvent être décompilés
- Un binaire peut être désassemblé de plusieurs manières

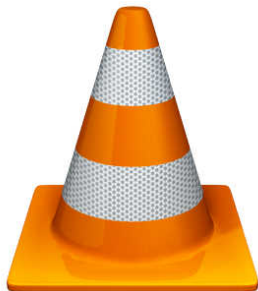
Pour faire quoi ?

- Espionnage industriel



Pour faire quoi ?

- Espionnage industriel



Pour faire quoi ?

- Espionnage industriel
- Cracking



Pour faire quoi ?

- Espionnage industriel
- Cracking
- Création d'exploits



Pour faire quoi ?



- Espionnage industriel
- Cracking
- Création d'exploits
- Audits de sécurité

MS15-020

Pour faire quoi ?

- Espionnage industriel
- Cracking
- Création d'exploits
- Audits de sécurité
- Correction de bugs



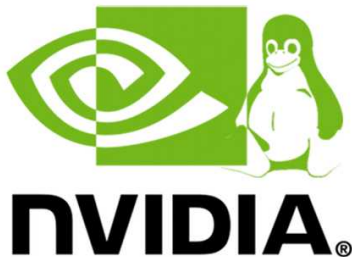
Pour faire quoi ?

- Espionnage industriel
- Cracking
- Création d'exploits
- Audits de sécurité
- Correction de bugs
- Analyse de malwares



Pour faire quoi ?

- Espionnage industriel
- Cracking
- Création d'exploits
- Audits de sécurité
- Correction de bugs
- Analyse de malwares
- Inter-opérabilité



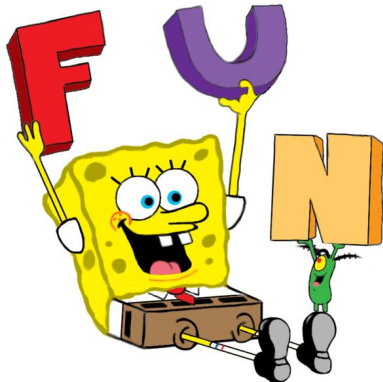
Pour faire quoi ?

- Espionnage industriel
- Cracking
- Création d'exploits
- Audits de sécurité
- Correction de bugs
- Analyse de malwares
- Inter-opérabilité



Pour faire quoi ?

- Espionnage industriel
- Cracking
- Création d'exploits
- Audits de sécurité
- Correction de bugs
- Analyse de malwares
- Inter-opérabilité
- Pour le fun



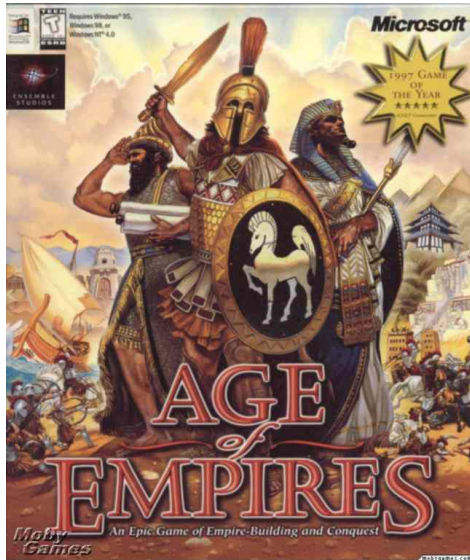


Mon week-end

Mon enfance



Mon enfance



Mon enfance



ONLY ?



CHALLENGE
ACCEPTED !



L'entretien d'embauche

pwnme.c

```
#include <stdio.h>
#include <string.h>

void
greet(char* name){
    char buf[64];

    strcpy (buf, name);
    strcat (buf, " est le plus beau.");

    printf ("%s\n", buf);
}

int
main(int argc, char **argv) {
    if (argc < 2) {
        return 1;
    }

    greet (argv[1]);

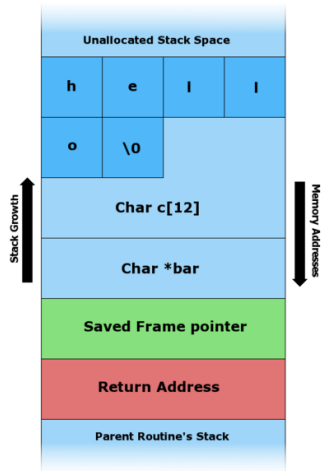
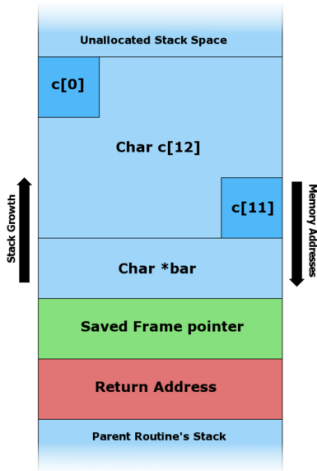
    return 0;
}
```

Rappel

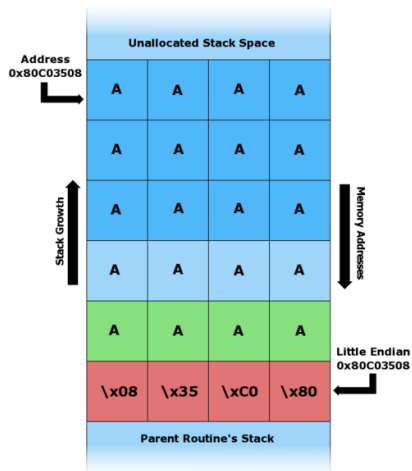
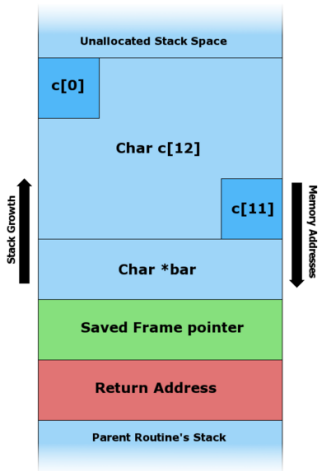
```
gdb-peda$ r
Starting program: /home/jvoisin/prez/SudTelecom/pwn/a.out
[-----registers-----]
EAX: 0x1 (b'\x01')
EBX: 0xf7faf000 --> 0x1a8da8
ECX: 0xffffd110 ("\001")
EDX: 0xffffd134 --> 0xf7faf000 --> 0x1a8da8
ESI: 0x0
EDI: 0x0
EBP: 0xffffd0f8 (")
ESP: 0xffffd0f4 --> 0xffffd110 ("\001")
EIP: 0x80484ca (<main+14>:      sub      esp,0x4)
[-----code-----]
0x80484c6 <main+10>: push    ebp
0x80484c7 <main+11>: mov     ebp,esp
0x80484c9 <main+13>: push    ecx
=> 0x80484ca <main+14>: sub     esp,0x4
0x80484cd <main+17>: mov     eax,ecx
0x80484cf <main+19>: cmp     DWORD PTR [eax],0x1
0x80484d2 <main+22>: jg      0x80484db <main+31>
0x80484d4 <main+24>: mov     eax,0x1
[-----stack-----]
0000| 0xffffd0f4 --> 0xffffd110 ("\001")
0004| 0xffffd0f8 (")
0008| 0xffffd0fc --> 0xf7e1fa83 (<__libc_start_main+243>:      mov     DWORD PTR [esp],eax)
0012| 0xffffd100 --> 0x8048500 (<__libc_csu_init>:      push    ebp)
0016| 0xffffd104 (")
0020| 0xffffd108 (")
0024| 0xffffd10c --> 0xf7e1fa83 (<__libc_start_main+243>:      mov     DWORD PTR [esp],eax)
0028| 0xffffd110 ("\001")
[-----]
Legend: code, data, rodata, value

Breakpoint 1, 0x80484ca in main ()
gdb-peda$
```

Stack smashing



Stack smashing



Hu-ho.

```
jvoisin@kaa 10:33 ~/prez/SudTelecom/pwn ./a.out jvoisin
jvoisin est le plus beau.
jvoisin@kaa 10:33 ~/prez/SudTelecom/pwn ./a.out $(python -c 'print "A"*128')
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA est le plus beau.
zsh: segmentation fault ./a.out $(python -c 'print "A"*128')
jvoisin@kaa 10:33 ~/prez/SudTelecom/pwn █
```

```

Program received signal SIGSEGV, Segmentation fault.
[-----registers-----]
EAX: 0x63 (b'c')
EBX: 0xf7faf000 --> 0x1a8da8
ECX: 0xf7fd6000 ('A' <repeats 76 times>, "BBBB est le plus beau.\n")
EDX: 0xf7fb0878 ("" )
ESI: 0x0
EDI: 0x41414141 (b'AAAA')
EBP: 0x41414141 (b'AAAA')
ESP: 0xffffd080 (" est le plus beau.")
EIP: 0x42424242 (b'BBBB')
[-----code-----]
Invalid $PC address: 0x42424242
[-----stack-----]
0000| 0xffffd080 (" est le plus beau.")
0004| 0xffffd084 (" le plus beau.")
0008| 0xffffd088 ("plus beau.")
0012| 0xffffd08c (" beau.")
0016| 0xffffd090 --> 0xf7002e75
0020| 0xffffd094 --> 0xffffd0b0 ("\002")
0024| 0xffffd098 ("" )
0028| 0xffffd09c --> 0xf7e1fa83 (<__libc_start_main+243>:      mov     DWORD PTR [esp],eax)
[-----]
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0x42424242 in ?? ()
gdb-peda$

```

Exploit !

```
+-----+ +-----+ +-----+
|      | |      | |      |
| nopsled | | shellcode | | call ecx |
|      | |      | |      |
+-----+ +-----+ +-----+
      ^                               |
      +-----+
```

Challenge accepted



CHALLENGE
ACCEPTED !

Conclusion

- La rétro-ingénierie, c'est cool

Conclusion



- La rétro-ingénierie, c'est cool
- Age of Empire, ça marche sans CD

Conclusion



- La rétro-ingénierie, c'est cool
- Age of Empire, ça marche sans CD
- Poper des shells, c'est marrant

Conclusion

If you don't **root** it, you don't **own** it!

