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: