Proactive Computer Security
Assignment 2

Nikolaj Dybdahl Rathcke (rfq695)

May 6, 2016

I have (tried) to solve task 1, 2 and 3 tasks in the 'easy' category. The files are located in in the `src` folder, where task 1 is in `src/1_1.asm`, task 2 in `src/1_2.asm` and task 3 in `src/1_3.asm`. However, I did not make task 3 work, so no point in running it.

The code should be comprehensible as I have tried to use labels that describes what that code segment does, i.e. if if we write to `stdout`, that segment is labeled "write". Though there are some jumps, it should be readable as I have tried my best to comment on most of the instructions.

1 Challenges

In task 1, the biggest challenge was understanding how `sys_execve` works. Mainly, it was difficult to place the correct things in the registers. Had to remember to `NULL` terminate strings and when to use a pointer to a string instead of just the string itself. Namely, it took some time to understand that I needed do move the stackpointer into the registers. After that was settled I had some issues where I did not clear registers, but this eventually worked as well.

In task 2, I really only encountered one new challenge. I still had the same troubles as in task 1, but with a little focus a some help from the program `gdb`, I could work out what was in the registers and make it work. The only problem I had was how to increment the ASCII value. It did not really occur to me that you needed to pop it from the stack, increment it, and push it back.

For task 3, the idea was to split it into 3 parts. If the number is 100, I wanted to print that, and then having a loop for all number on 2 digits and lasts printing all numbers on 1 digit. Each time decrementing the ASCII value for each digit in the number. After each print, I wanted to compare it to either 10, so I knew when to just print single digit numbers. Unfortunately, I didn't get it to work.

2 NULL bytes and minimizing the size of the program

Unfortunately, in task 1, I could not give `wall` the argument "hello" as it gave me an error. However, from the discussion forum I was told it was fine if it just printed the error. The program has no `NULL` bytes. I didn't do much to avoid it, but I had a few when the program worked and I figured out where they were and fixed it. The only real trouble was I wanted to `NULL` terminate the "hello" string. I didn't know how to fix it, so I removed it in the end, which means if someone runs the program where `wall` works properly, it might also print something after the hello. But for me, it made no difference. As for the size, I did not spend a lot of time optimizing, but I found a few unnecessary instructions. The type was mainly registers I cleared which didn't need to be cleared.

In task 2 I tried to clear the `NULL` bytes, but I was left with one which I cannot locate. Again, I didn't spend much time making the program smaller as I tried to make task 3 work and really just ran out of time.