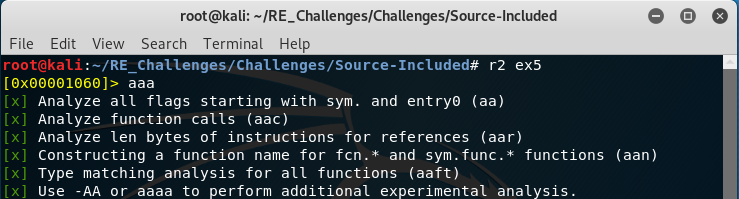
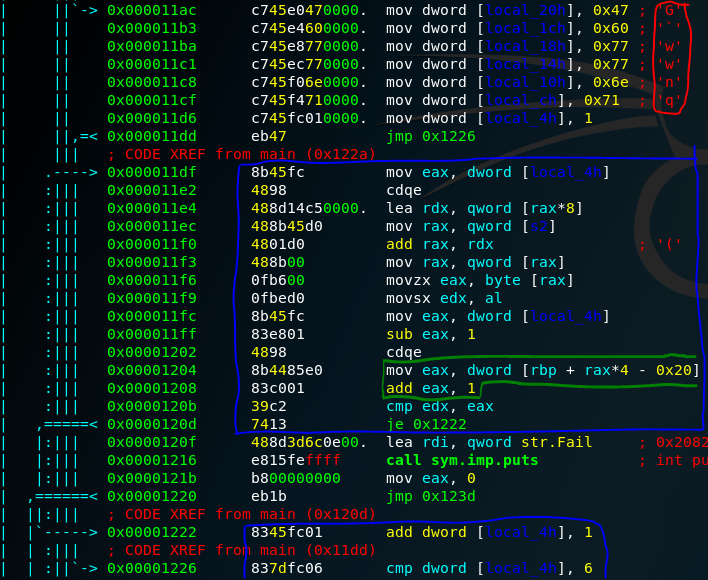
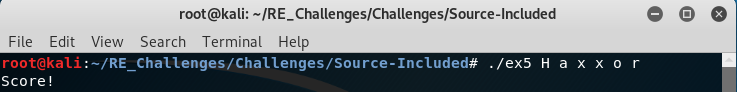
***Ex5***

We can think of this challenge as a mash up of exercises 1 and 2, so hopefully those are still fresh one your mind because this one will require just a little more attention to the assembly instructions. Don’t worry if you still are struggling with reading instructions, it will come in time. Like all things it just takes practice and just about all you can do is practice. However, just reading the instructions line by line is a pretty painful way of learning.

We start off as usual with radare2 and basic analysis commands. Have you tried playing around with permutations to the analysis command or ever leaving it out yet? Give it a shot, you won’t break anything. Well, even if you do isn’t that half the fun anyways?

This one will be fun, where to begin? Like always we want to look inside the main function with a “pdf@main” to get the higher level overview of what a program is doing. Scrolling through we have a red block with 6 hard-coded characters being pushed into the stack and then a loop structure in the blue block. As mentioned, this is a challenge similar to ex1 and ex2 so we have good reason to suspect caesar ciphering to be included in this challenge. That is what the green block is for. Since we are pushing each of those 6 characters onto the stack we need a way to reference them and add one, the mov command is using arithmetic in its source operand to calculate the offset into the stack relative to the base pointer of the stack. The value of rax prior to this move is the loop index counter i-1 and is being multiplied by 4 because each of those hex character values are not actual char types but of type int, char=1 byte & int= 4 byte for our architecture.

Well seeing that each of the values we stored on the stack is being referenced and incremented by one we can increment the hex of each value and pass that to our program to make the comparison pass each loop iteration.