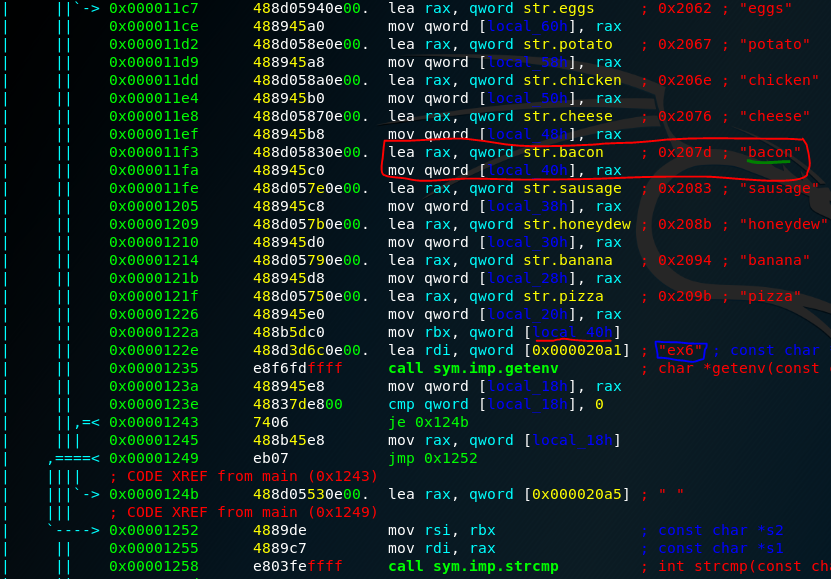
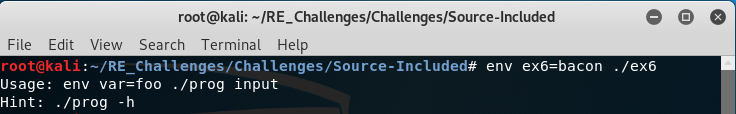
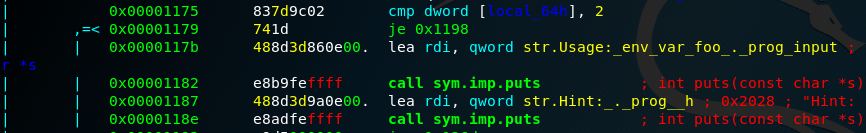
***Ex6***

This challenge is not very complicated but with out using the help you would need to be aware of a little bit of information about the impact of the context in which a program is executed, the environment. The environment is a series of variables passed to a program so the program in question will know more about the context in which it was executed if need be. As you may have noticed by now the function prototype for the “main” function in C is “int main(int argc, char\*\* argv);”, while this is true from the perspective of the programmer it is not completely true. The full prototype is “int main(int argc, char\*\* argv, char\*\* envp);” where envp is a pointer to characters pointers, simply array of strings, just like argv but this parameter is handled and passed to main by code that initializes and runs before main.

You will see several strings being loaded into the stack if you ran a “[pdf@main](mailto:pdf@main)” in radare2. Pay attention to the function call to “getenv” first, this is a function that will return the value stored in the environment variable passed to it, circled in blue, if it exists or returns NULL if the variable does not exist. From there we see that the value returned from “getenv” is compared with another value. Now lets see where this value comes from to see what we need to set our environment variable to. Back tracking a little we see this value comes from the stack variable local\_40h, underlined in red. If we walk backwards through the code even more we will get back to the sequence of strings being pushed onto the stack and can identify which one is place in “local\_40h”, circled in red and underlined in green.

What that didn’t work!? No worries you basically have it all figured out right now. I just left something in this challenge that isn’t “needed” per say. If you have been reading through the challenges assembly code as you have went along instead of just reading the help and moving on you may have noticed something that has been kind of consistent in every exercise.

The code that prints the “Usage” and “Hints” message appears right before a comparison in every challenge thus far.

That is because we are checking to see what our “argc”, argument count, is. In linux the 1st argument to a program is the name of that program itself so an argc of 2 means the program takes one argument. This may see odd since we never actually use ANYTHING from our “argv”, argument vector, which is an array of arguments that we pass to the program being executed. No worries you are reading the code right but if it expects a parameter but never uses it then we can give the program anything we want!

