Reverse Engineering

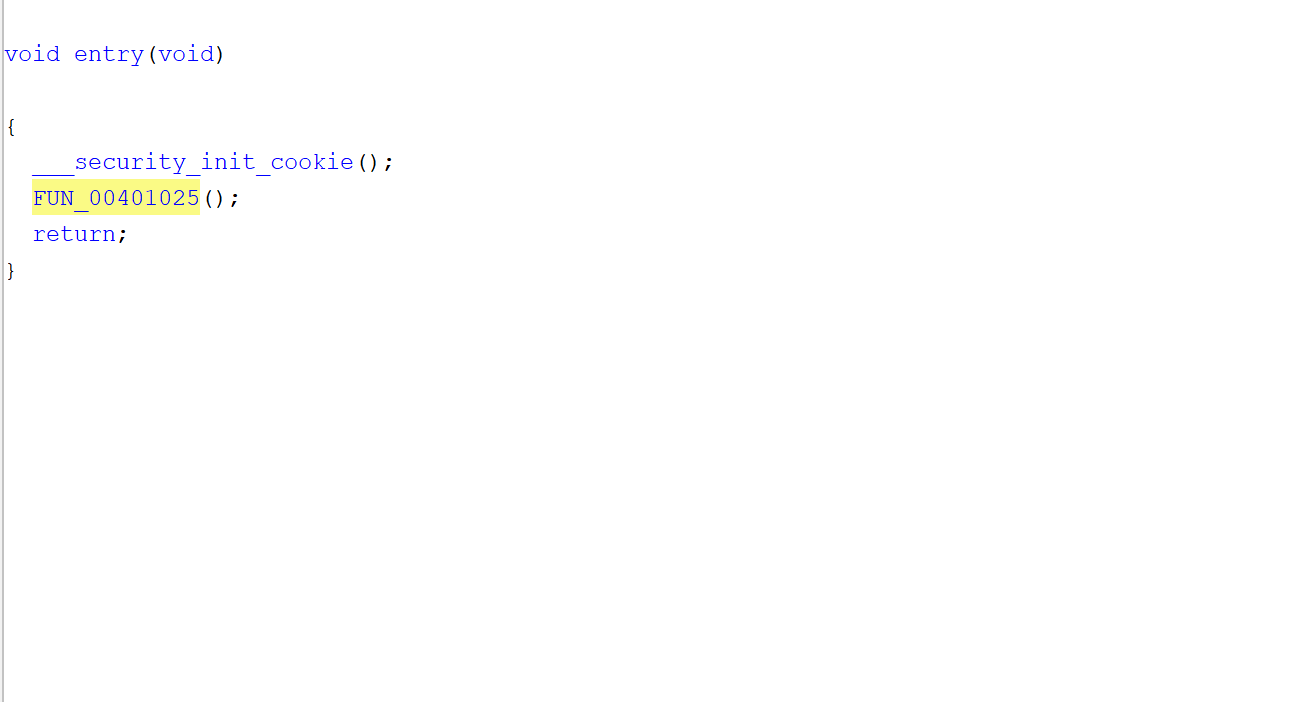
Exercise Writeup

Exer2\_Julia

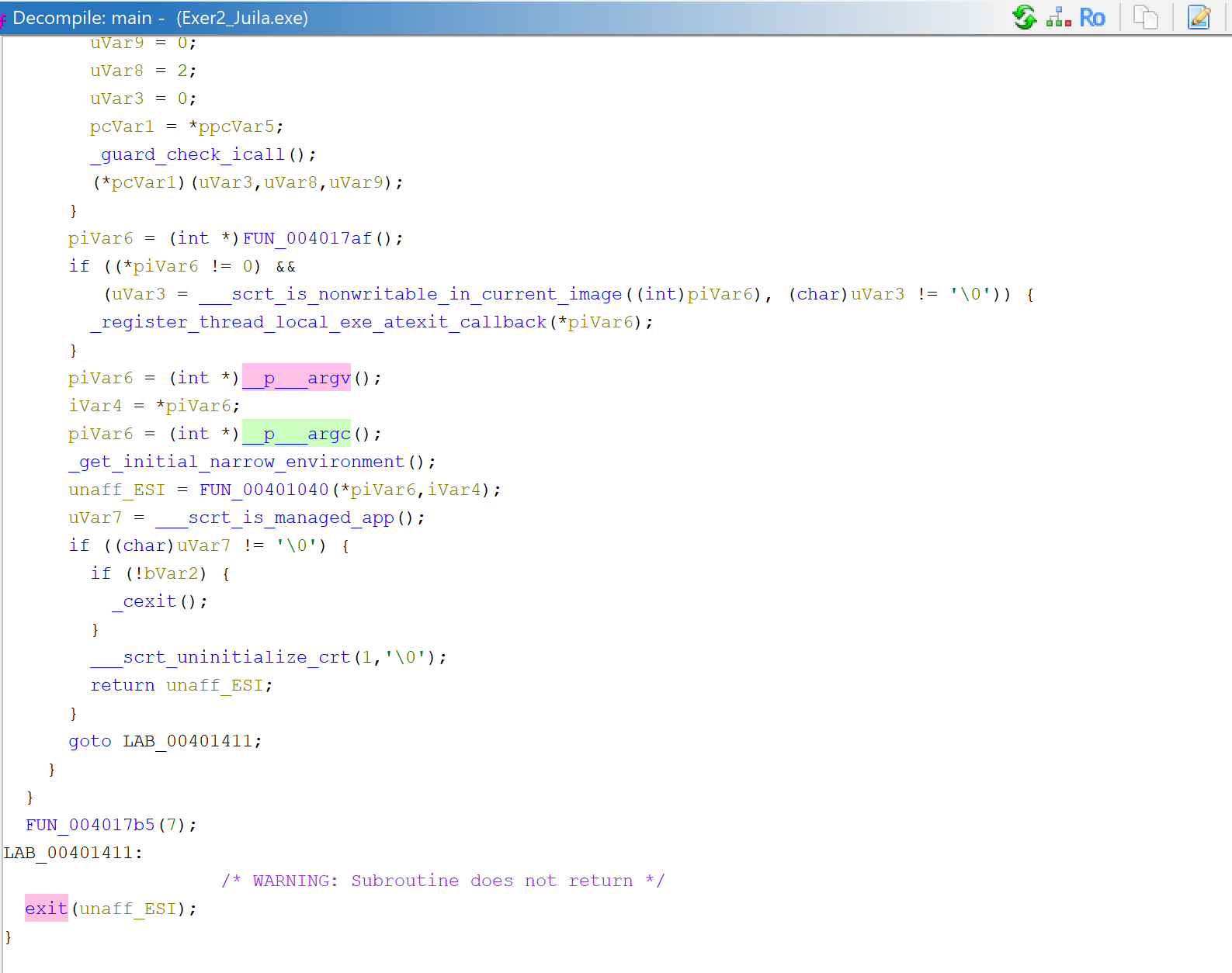
**Sample**: Exer2\_Julia.exe

**Link**: https://www.begin.re/julia

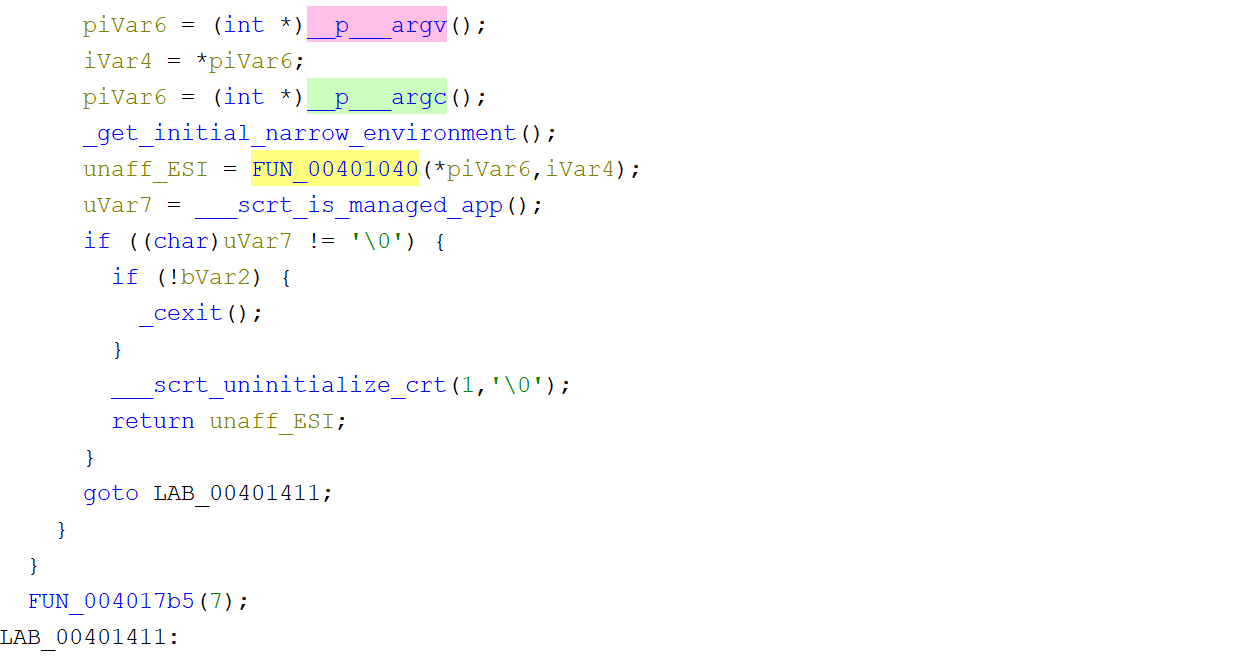
We have a **Windows Executable** which calls to a single function from entry.



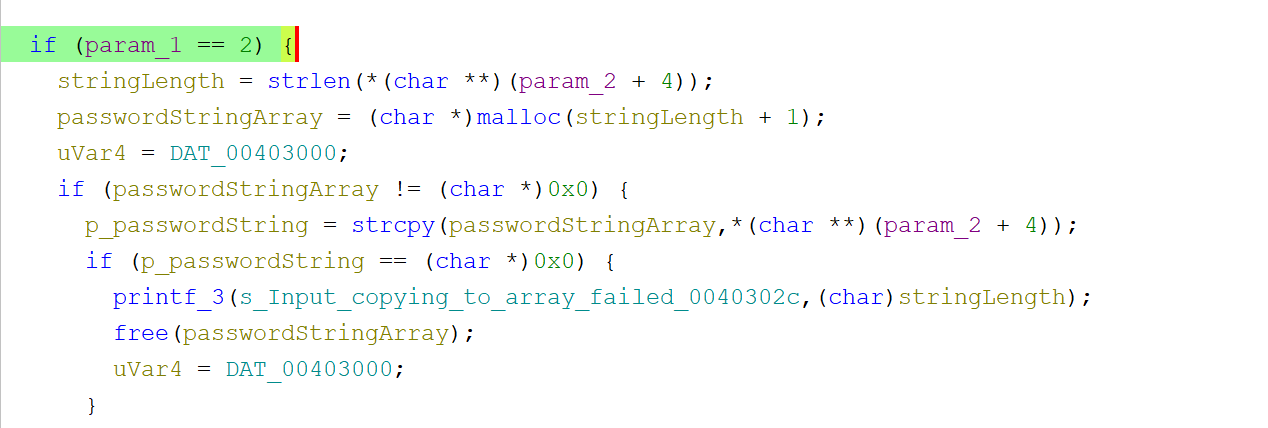
We can ensure that it’s the **main()** function as it can call **argv()** and **argc()** (in this case by their references). Since the function also doesn’t return and instead exits after executing, we can assume it’s the main().

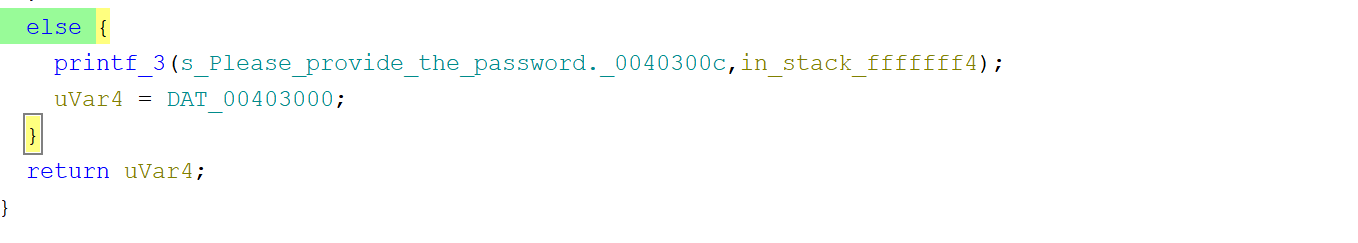


We can see references to \_\_p\_\_argv() and \_\_p\_\_argc() are being set to **piVar6** and **iVar4**, which are then passed as parameters to the **FUN\_00401040()**.

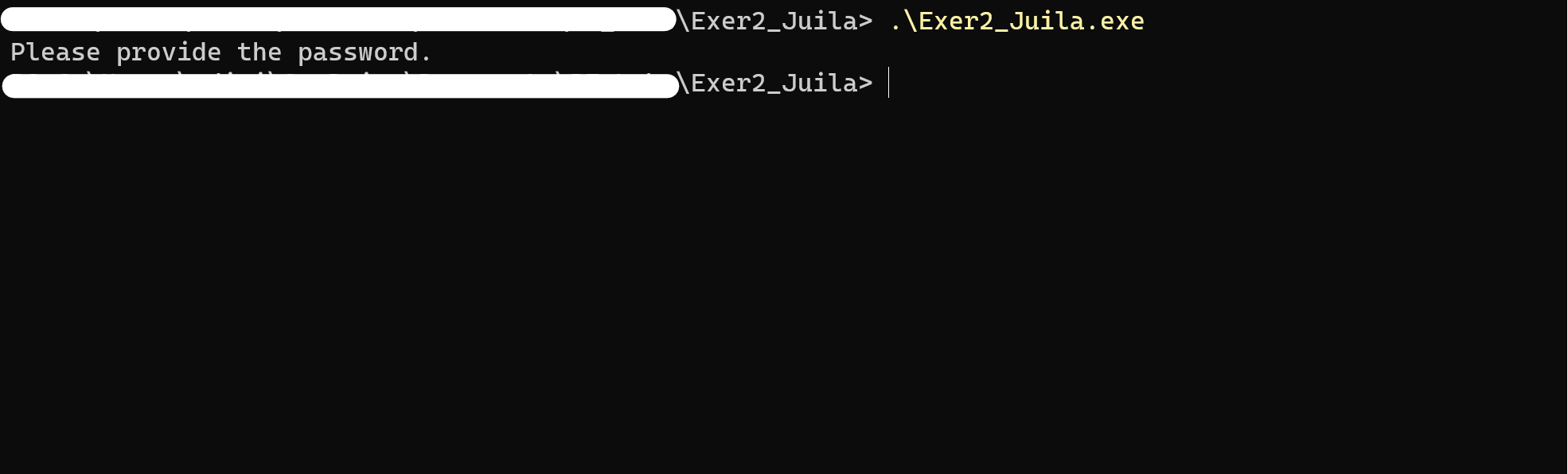


Within **FUN\_00401040()**, we find the first line to be checking if argc() == 2, in other words whether two arguments were inputted into the command line or not. Scrolling down will tell us that if argc() != 2, the message “Please provide the password.” is returned and displayed.

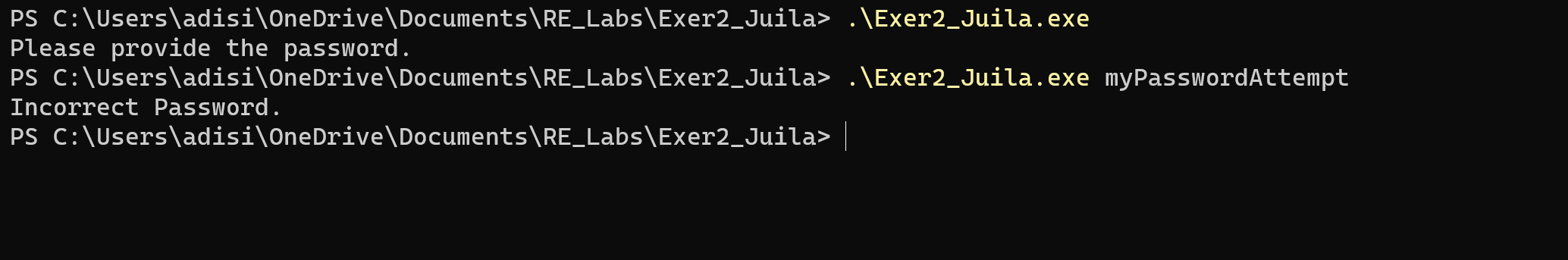




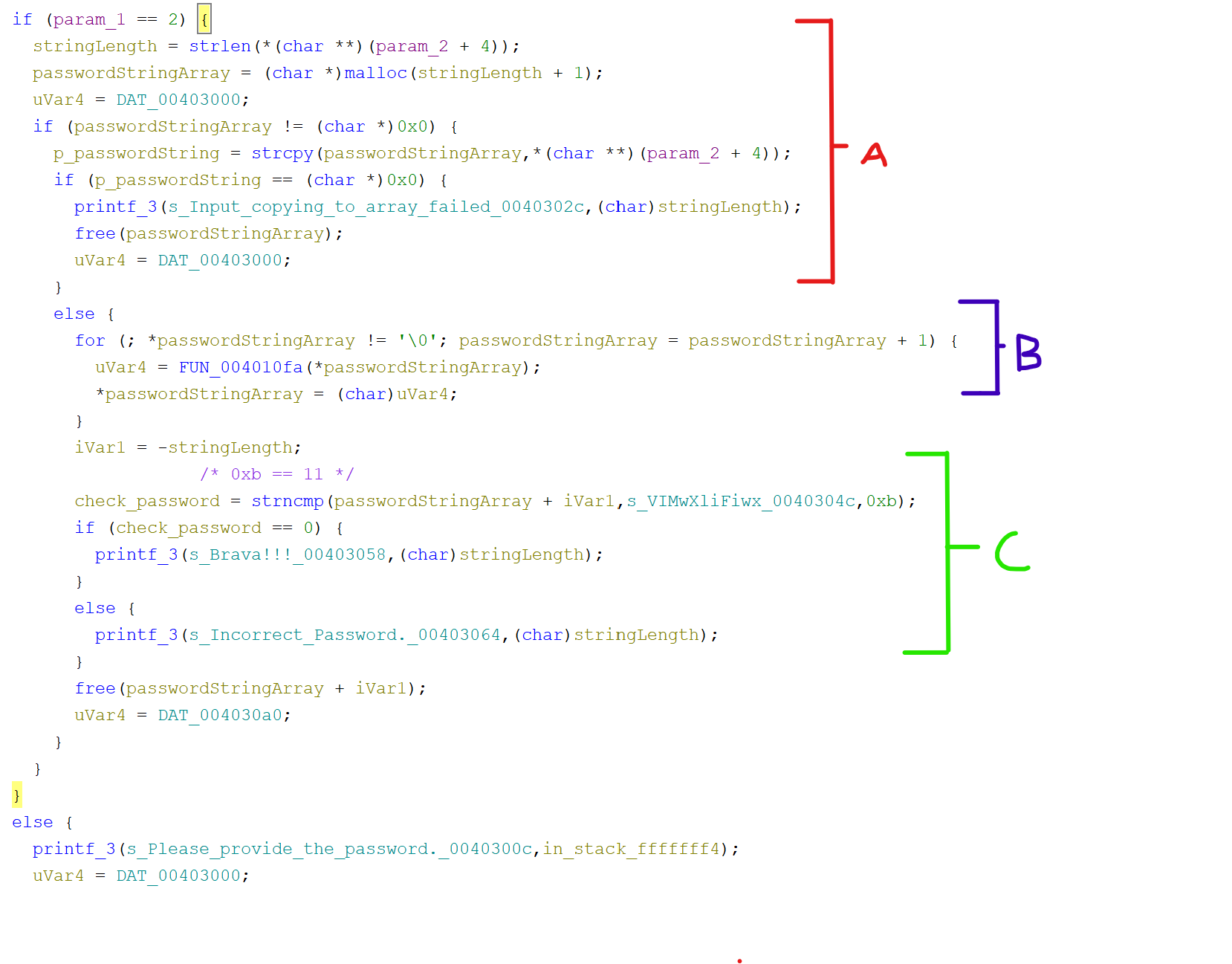
Here’s the function in action.



We can also see that if I provide a second argument as the incorrect password, it will return and display “Incorrect password.”



After looking around the function and renaming some variables, we can better understand what’s going on within the if-else statement.

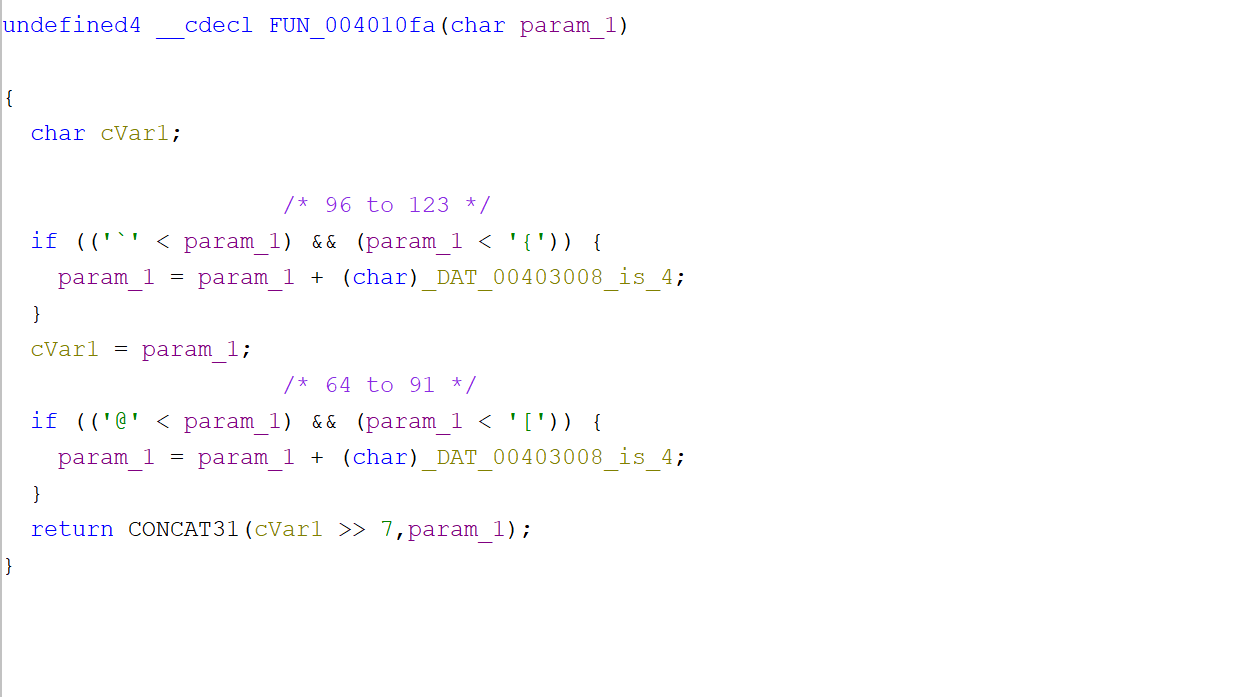


In *section A*, a char-array **‘passwordStringArray’** with its memory set to equal the length of param\_2. It’ll then check if it’s empty to output “input copying to array failed” and break out the loop.

If the passwordStringArray isn’t empty, it’ll move onto se*ction B*, where each char is processed by the **FUN\_004010****fa()** function.

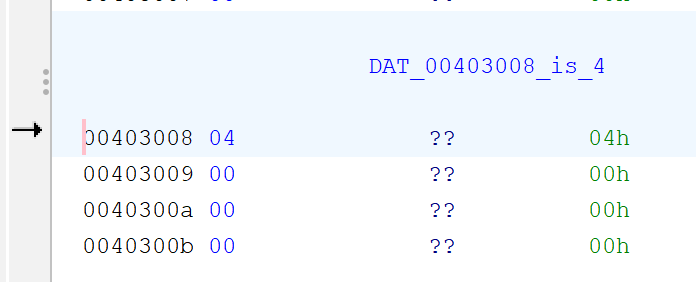
In *section C,* he processed PasswordStringArray is then checked against the string “**VIMwXliFiwx**” with strncmp(). If they’re equal, the program will print “Brava!!”, otherwise “Incorrect Password”.

We can see that FUN\_004010fa() is doing some transformation to param\_2, then checked against the string “**VIMwXliFiwx**”. We can see the transformation code within the function here:

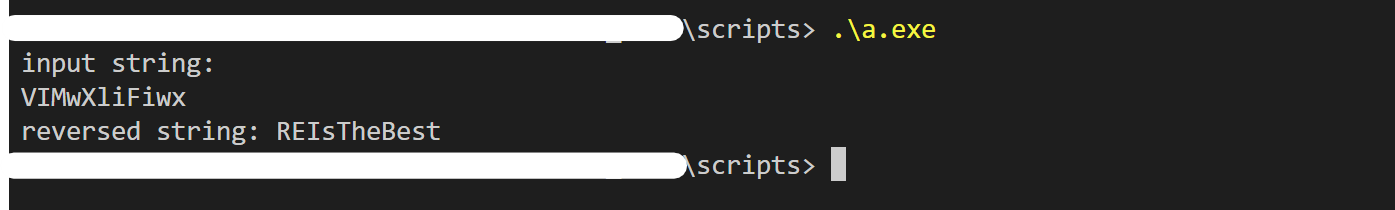


The function takes a character as input and transforms it based on two specified rules. If the character falls within the ASCII range of 'a' to 'z' or 'A' to 'Z', it shifts the character by a certain amount (**\_DAT\_00403008**). Then, it returns the transformed character. Essentially, it’s just a simple Caeser cypher.

The reference tells us that \_DAT\_00403008 is equal to int 4.

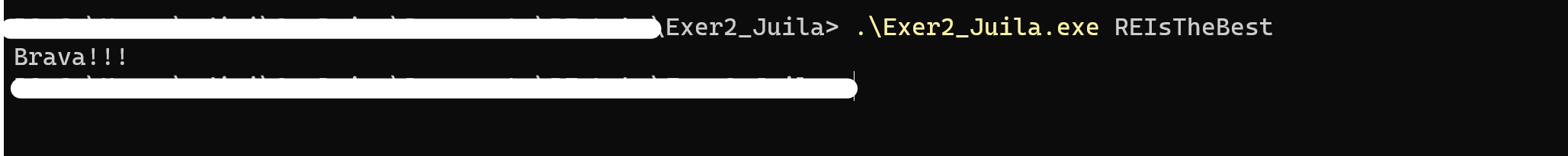


From here, we can write our own C++ program identical to FUN\_004010fa(), but instead sets \_DAT00403008 as –4.



We know that the transformed param\_2 is checked against the string “**VIMwXliFiwx”**. So reverse Caeser-shifting this string should give us the correct password. Doing so with the script outputs “**REIsTheBest**”

We then verify if it’s the correct password by running ‘**Exer2\_Julia.exe REIsTheBest**’.



“Brava!!!” is returned, indicating “REIsTheBest” is the correct password.

---- END ----