Author: Saleh Bin Muhaysin

Email: SalehMuhaysin@gmail.com

Link: https://github.com/salehmuhaysin

Introduction:

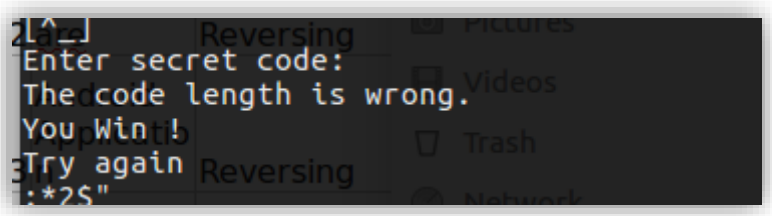
In this write up I will write how I solved the challenge called "**Back to Basics**" in Kaizen Arabia CTF which belongs to Reverse Engineering category. The request of the challenge as following:

"Olympic City recently hired a developer to build a software for smart grids. The software needs to be activated using a serial number. However, it seems that the software is sloppy. Can you crack its serial number?"

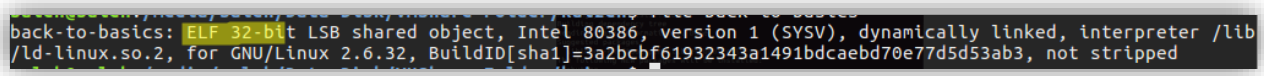
So what we want is the serial number of the executable.

Step 1 – Overview:

As usual I started with the "**strings**" results, but unfortunately nothing interesting except



This is an indication that the executable may ask for a secret code and check it against the correct secret code and return "**You Win !**" if it is correct. Let's see what is the type of the file:

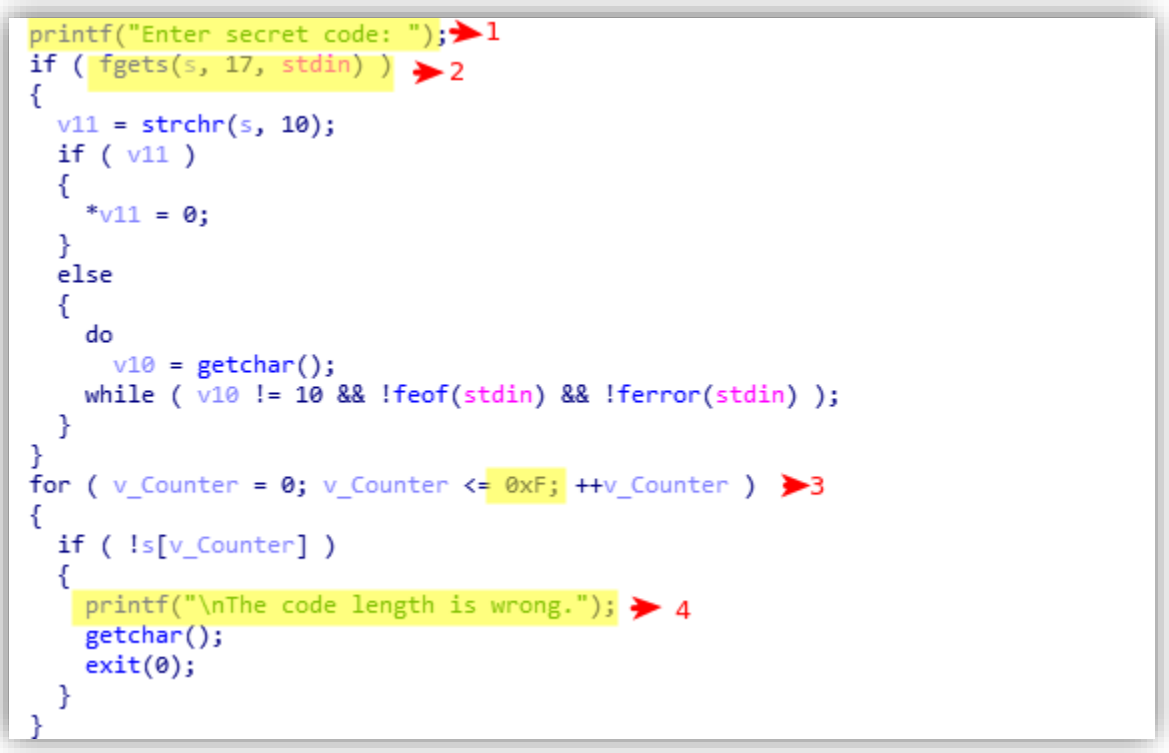


Here we can see that the file is ELF 32-bit which is a Linux executable, when you run it on 64-bit Operating system it will not work and give the following error message:

So, we have to run it under 32-bit operating system to be executed correctly.

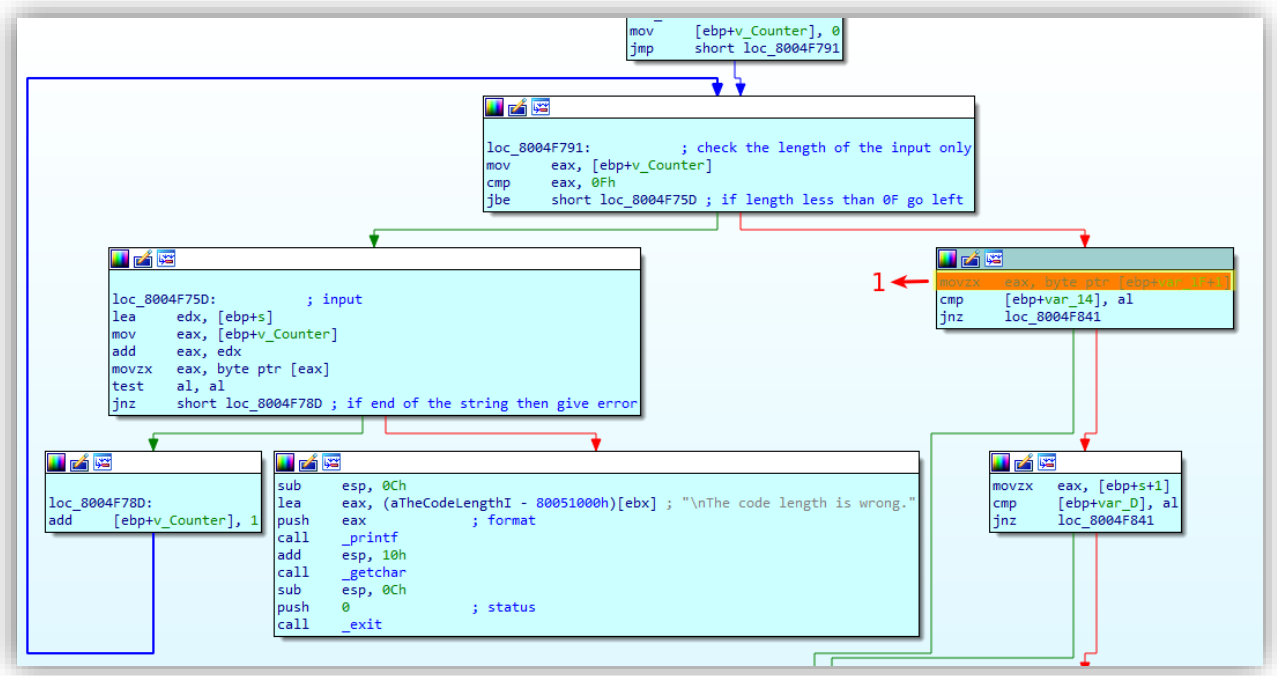
Step 2 – Analysis:

I usually start with decompiled code of the main function looking for anything interesting, luckily there some interesting points to look at:

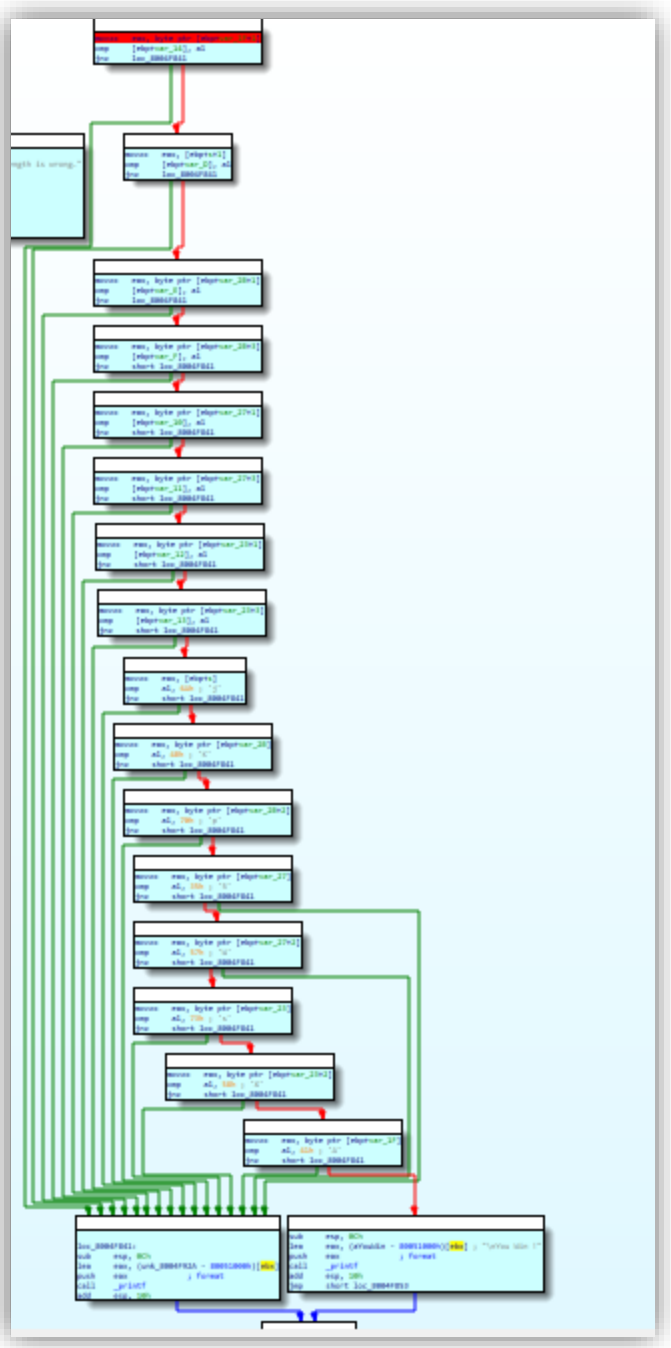


In first two points (1 and 2), it asks the user to enter a secret code with maximum length 17 characters including the final null character. And then in point 3 it checks every character in the given secret code if less than 0xF (16) characters (not including the final null character) print error message “The code length is wrong” (point 4) then exit.

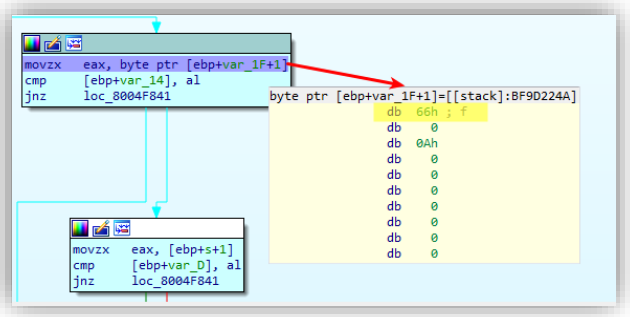
From this we can conclude that the secret key is exactly 16 characters, and for testing I will use the secret key “*0123456789abcdef*”, then set a breakpoint after the for loop:

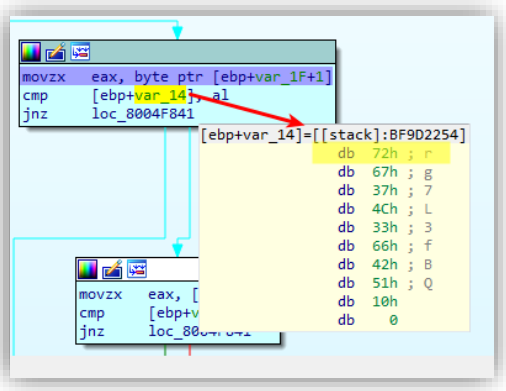


From there we can see that all the following blocks check the given secret code character by character if it is correct or not, if all characters correct it print the message "**You Win !**"



To get the correct secret code, in each block I check the comparison and see what is the current character of my test secret code and what is the character it compares it with, and then change my test secret code in memory to match the character of the correct secret code. Let’s start with the first block, and with all the next blocks I did the same.





Here we can see that it compares the character "f" from the given secret key with "r" and then if it is different it will print "**Try Again**" and stop the executable. Before it stores the character into *eax* register, I will replace the value "f" in address "***[ebp+var_1F+1]***" with the value "r" (using the Hex view), which will make the compression correct. After that continue with all the next compression with the same steps (check what is the correct character of the secret key and then replace it in the test key "*0123456789abcdef*").
So the sequence will be as following:

0123456789abcdef	The test secret key.
0Q2B4f638La7cger	After the first 8 comparisons blocks
jQKBpf53WLs7XgAr	After the last comparison block.

Finally, we test the secret key we got (**jQKBpf53WLs7XgAr**) as input to the "**Back to Basics**" executable.

```
./back-to-basics
Enter secret code: jQKBpf53WLs7XgAr

You Win !
```

The flag is the secret code:

jQKBpf53WLs7XgAr

Conclusion:

In this write-up we solved the Kaizen Arabia CTF called "Back to Basics", in general first we started with basics overview of the Linux 32-bit executable, then found the length of the secret code it requires, finally traced the execution of the comparison of the characters to find the correct code.