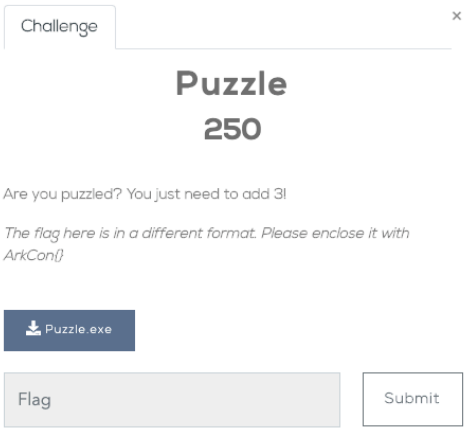
Puzzle

Author: Emmanuel Ouanounou



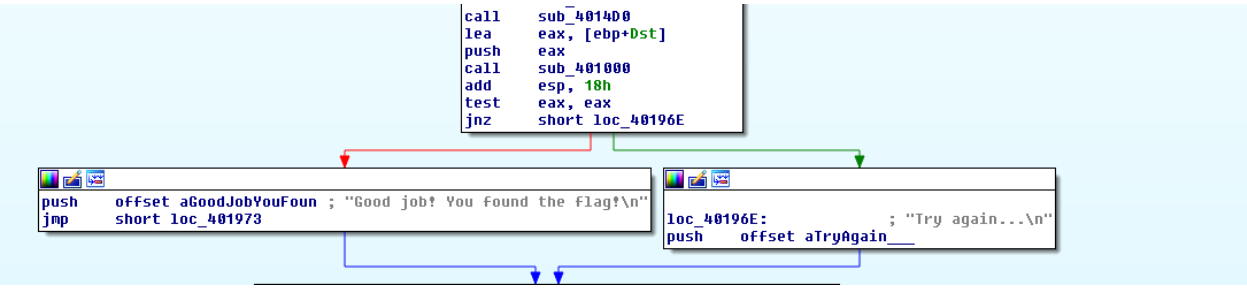
The aim of this challenge was to find a flag by reversing an executable. The idea behind it is the following: when you reverse an application, you need, somehow, to find the flag somewhere in the code, or a hash of it, or any information which would allow you to get it. Our idea was then, that we do not want to have the flag to appear in any way inside the executable. But how could we do it?

Let us try to solve the challenge, and then we will find out.

First, we run the executable:

```
C:\Users\user\Documents\Debug>Puzzle.exe 123
Try Again…
```

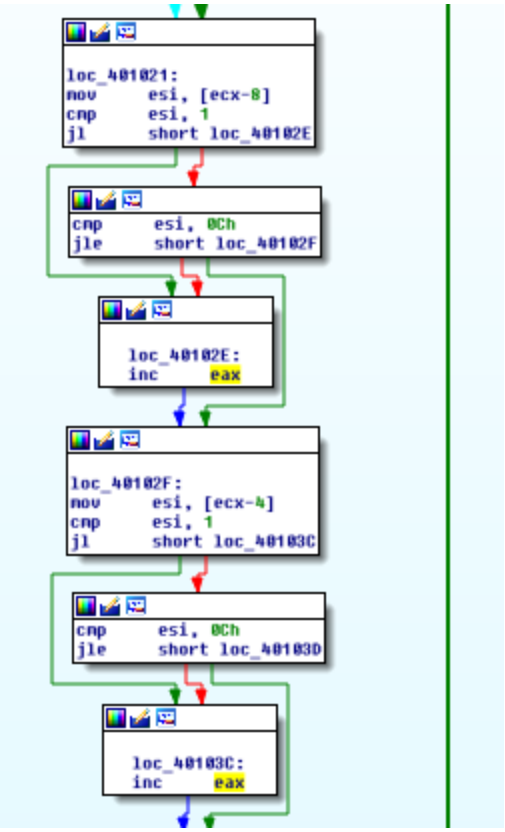
From this point, we understand that we need to reverse the program. Let's look for "Try Again…" in the code:



So we know that we need sub_40100 to return 0.

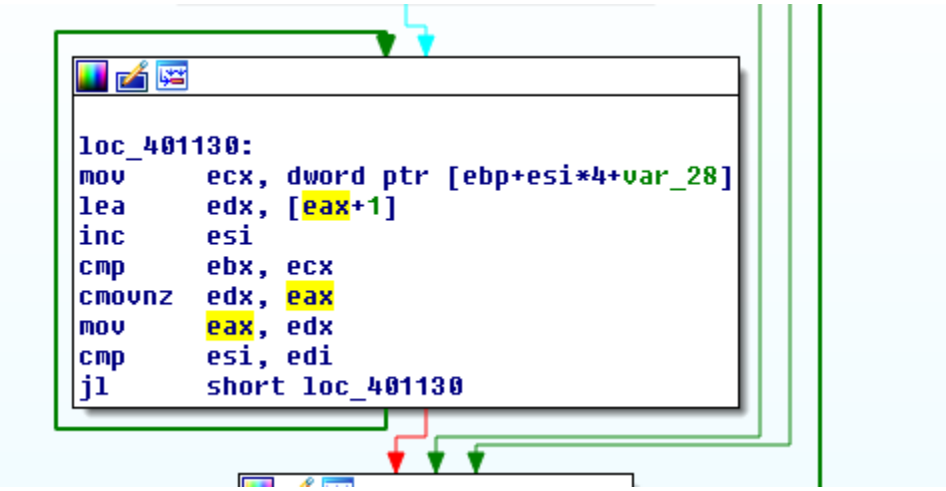
We also see before that, sub_4017B0 changes our input (argv) to numbers (hex to int).

Let us now look at the sub_40100 function:

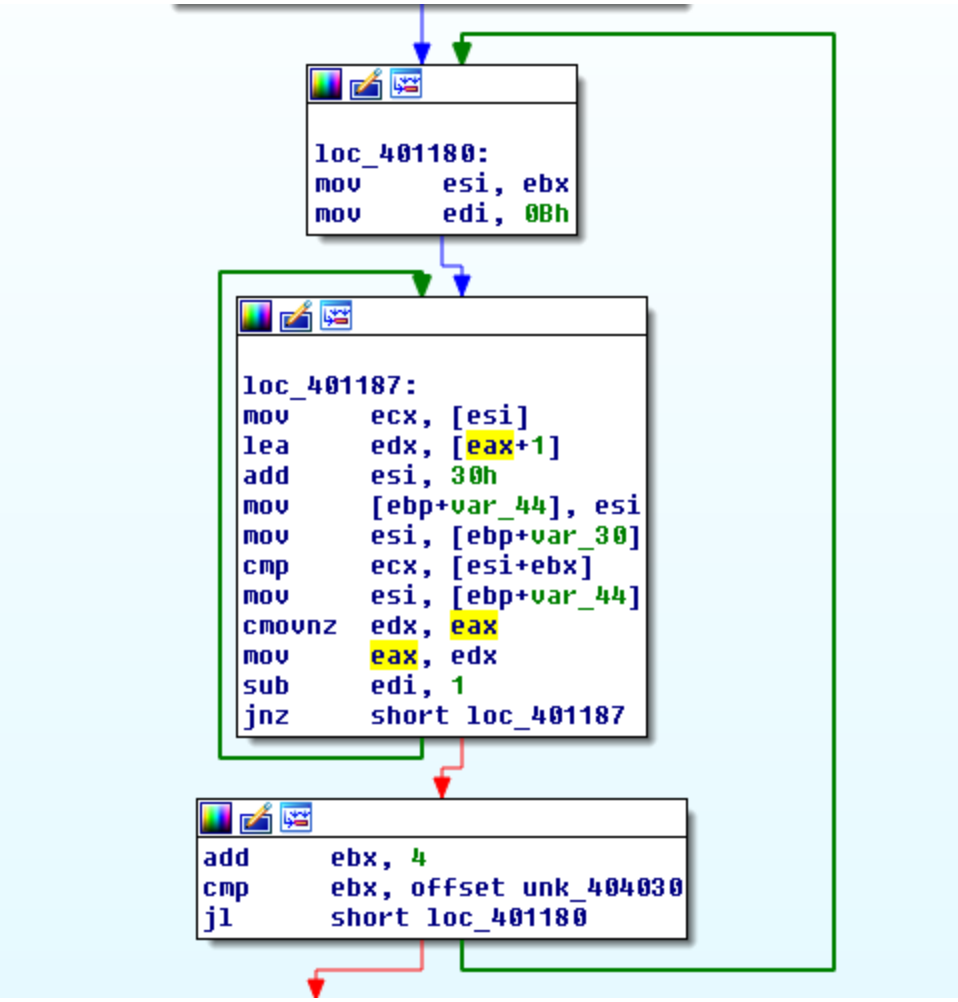


Here they check if all the numbers are between 1 and 12.

The following piece of code then checks if there is a number which appears twice in the user input:



Then, the following piece of code checks our input in front of a huge amount of numbers in memory:



If we look closer, it seems that the manipulations belong to a 2-dimensional array (of size 12x12). For each "column" j of this array, we check if the user input has a number at the j^{th} position. If yes, then the result is wrong (because it increments eax).

We see that the memory which is checked is different for every execution.

If we now step back and try to understand where we are, according to the name of the challenge, it is a puzzle. And the hint says: "Are you puzzled? Just add 3".

The checks that are done look like in a Sudoku, but a 12x12 one. Indeed, it checks if all numbers are between 1 and 12, then it checks if there is a number which appears twice on our input, and then, it checks if all the columns do not contain a duplicate when inserting our input.

By the way, the last loop of the checks, checks actually each 3x4 "cell" for duplicates ;-).

From this point, we can either create a program that solves the problem by giving it the checks and finding a solution… Or…

We can look at the memory by debugging the program and fill the line which is missing inside our Sudoku!

```
C:\Users\user\Documents\Debug>Puzzle.exe 27A5CB461893
Good job! You found the flag!
```