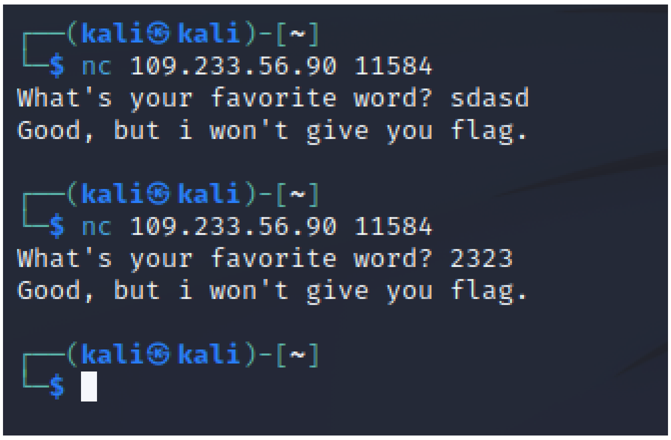
Buffer Overflow Report - 22.5.2 - 22

Tester's name: Stanley Ford

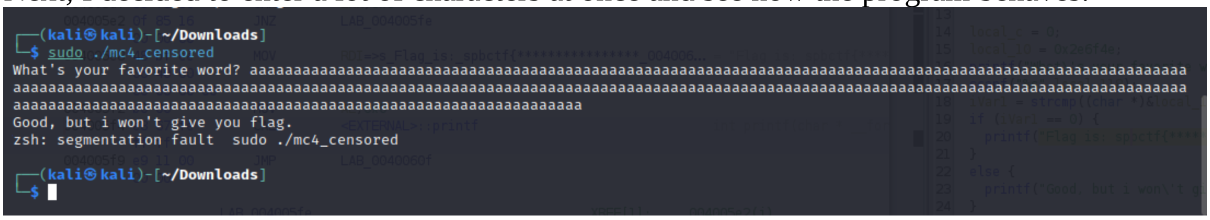
Lab 22.5.2 "Buffer overflow":

- General information:
  - Testing period: 01.04.2023
  - Test object: 109.233.56.90:11584
- Description of actions:

  I connected netcat to the server, figured out how its functionality worked: I entered random values and received a simple answer from it.



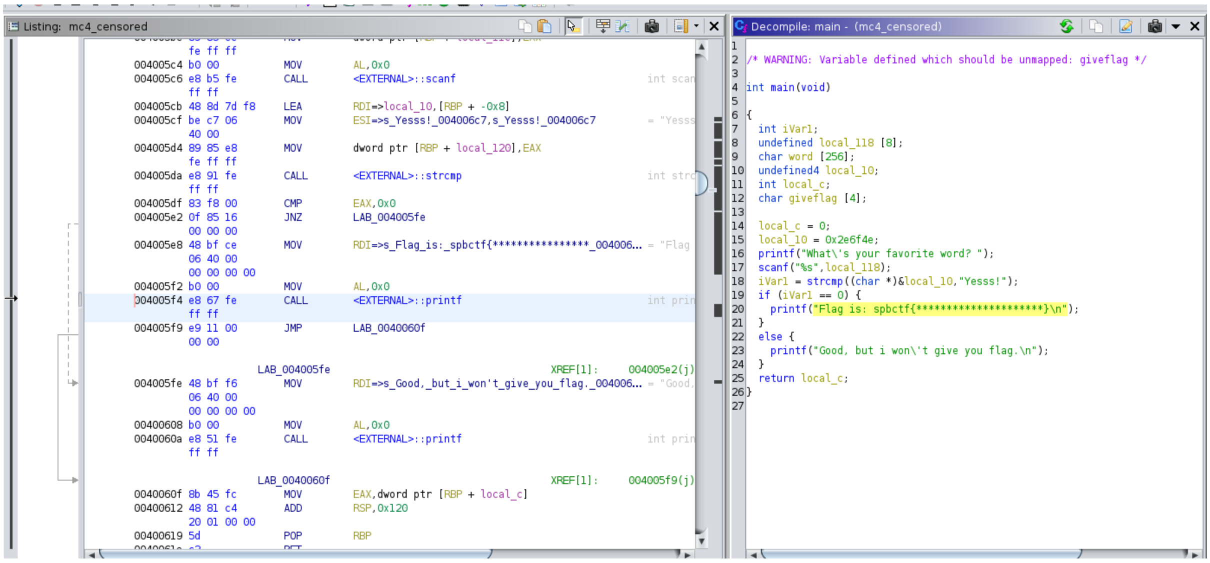
  Next, I decided to enter a lot of characters at once and see how the program behaves:



  When there are a lot of symbols, a terrible segmentation fault occurs and something breaks. It's time to figure it out.

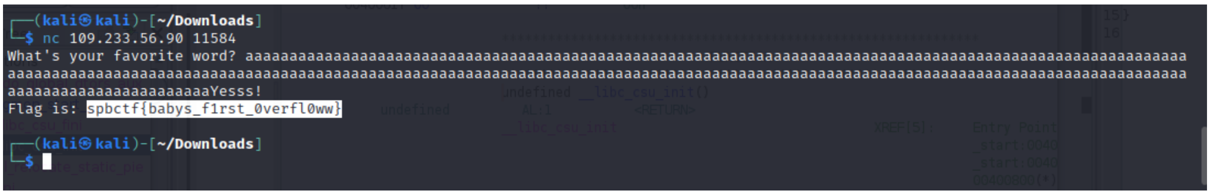
  With the help of the Ghidra utility, I was able to read the C code recovered from the original mc4_censored file, which is written in assembler.

The main function looks like this:



As you can see, the iVarl variable contains the value obtained by comparing the char* and "Yesss!" strings. If these strings match (0 is returned), we will get a flag. If not, then the conclusion is negative.

As we type, our codeword is written to an array local_118 8 elements long and compared by that value. By trial and error, I managed to find the value at which the program crashes and the buffer is overflowing: 264 characters. Add the check word "Yesss!" to the end of the line and get the following:



Flag: spbctf{babys_f1rst_0verfl0ww}

Lab work is done.

Self-Assessment Questions:

- List the tools (programs and utilities) that you used to solve this lab:

  Ghidra, netcat

- List the vulnerabilities you've discovered:

  Buffer Overflow

- Give advice on how to improve security: Add a check to scanf for the number of characters to be entered.