SECURE CODING CSE2010

K Roma Pai -18BCE7165

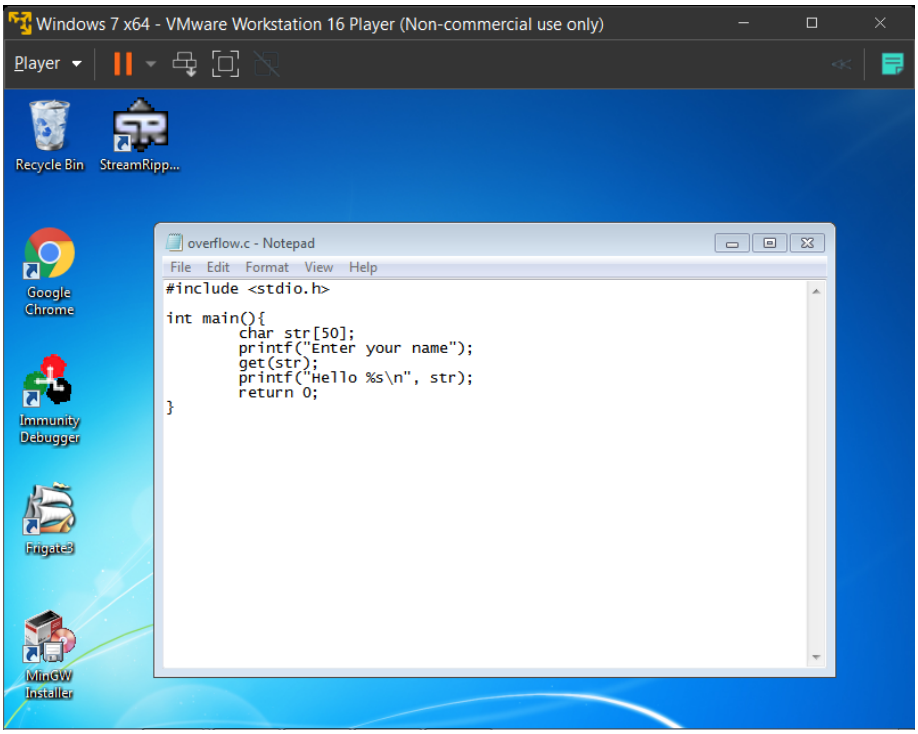
Working with memory vulnerabilities

TASK

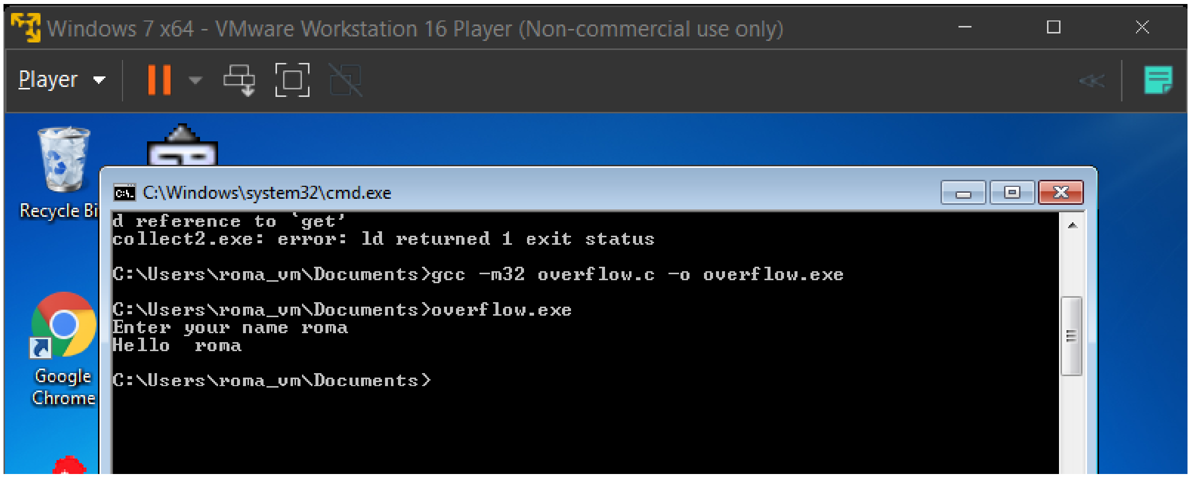
- **Attach the debugger (immunity debugger or ollydbg) and analyse the address of various registers listed below**
- **Check for EIP address**
- **Verify the starting and ending addresses of stack frame**
- **Verify the SEH chain and report the dll loaded along with the addresses. For viewing SEH chain, goto view → SEH**

Firsty download Immunity Debugger, and an extension called mona.py, and a compiler for c

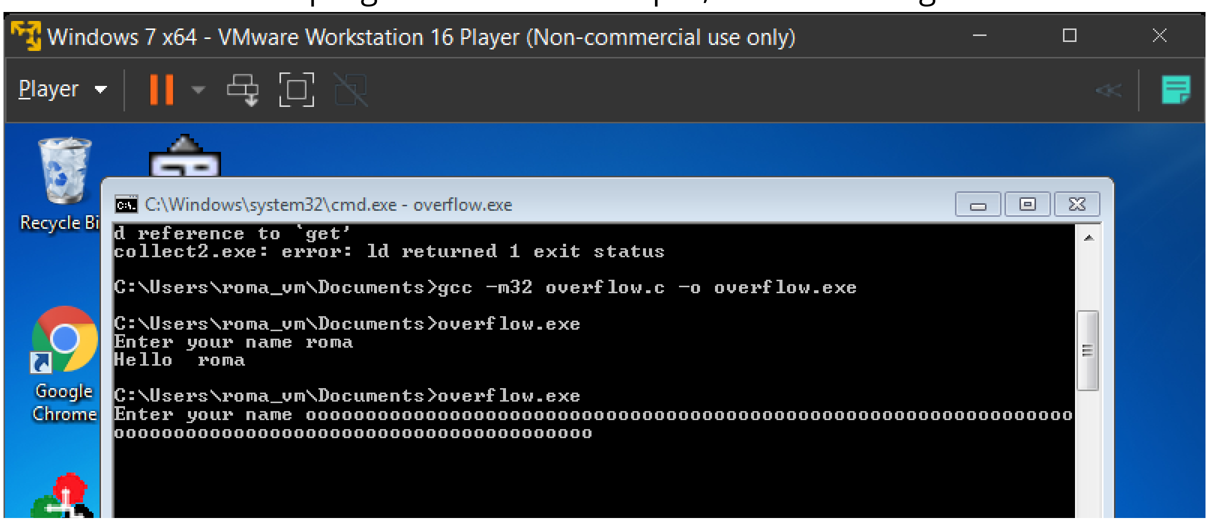
Next, we write a simple C program which we can use to see the overflow



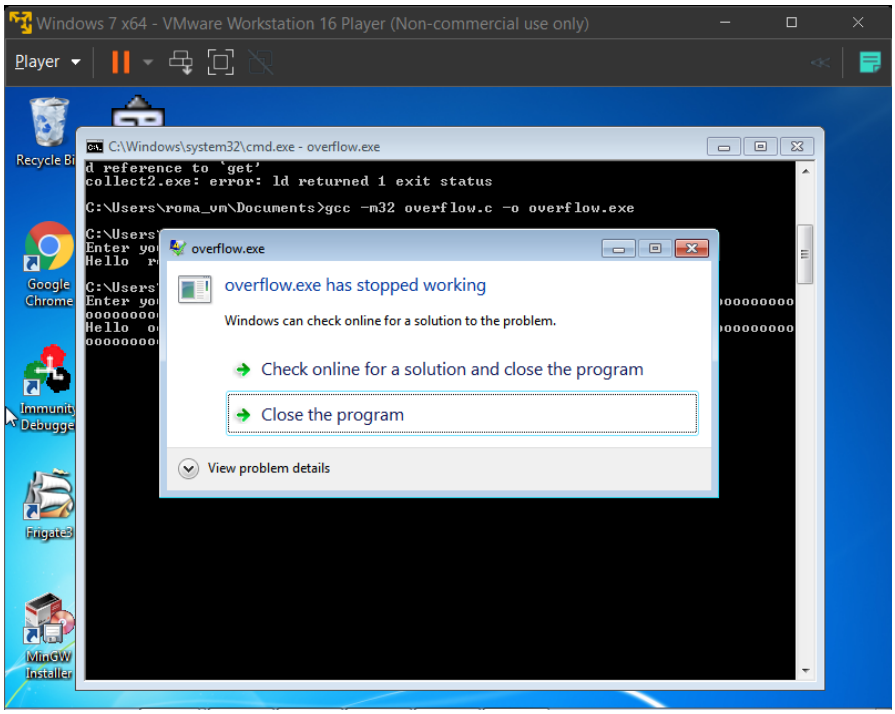
Compile and run the program with a valid input (length <buffer size)



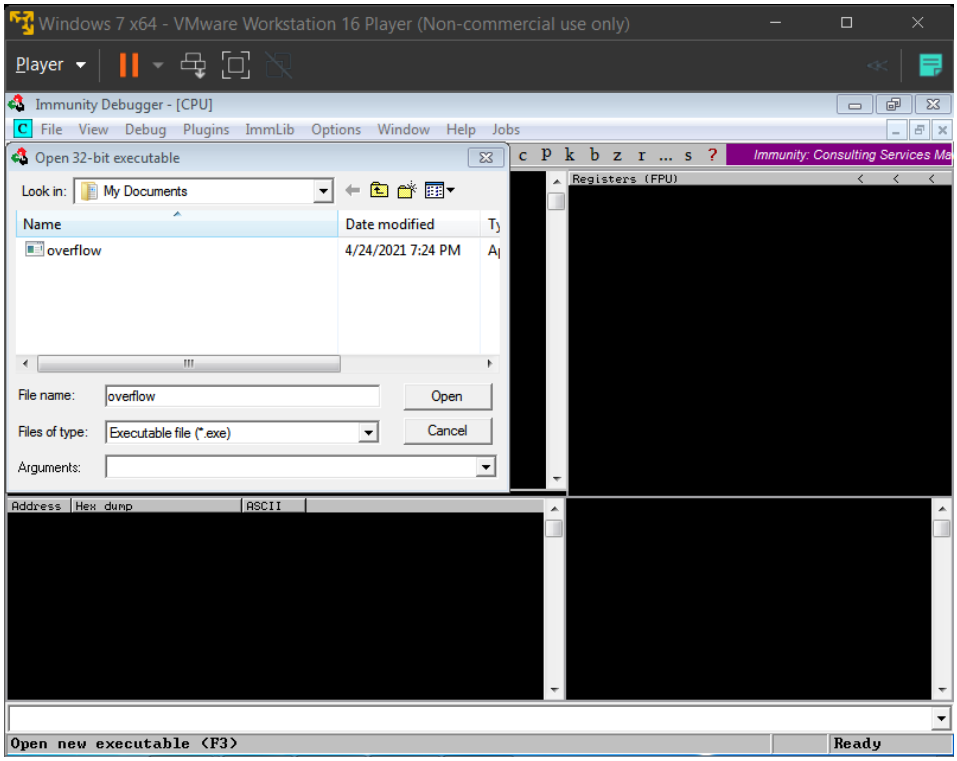
Now run the program with invalid input, i.e overflowing the buffer

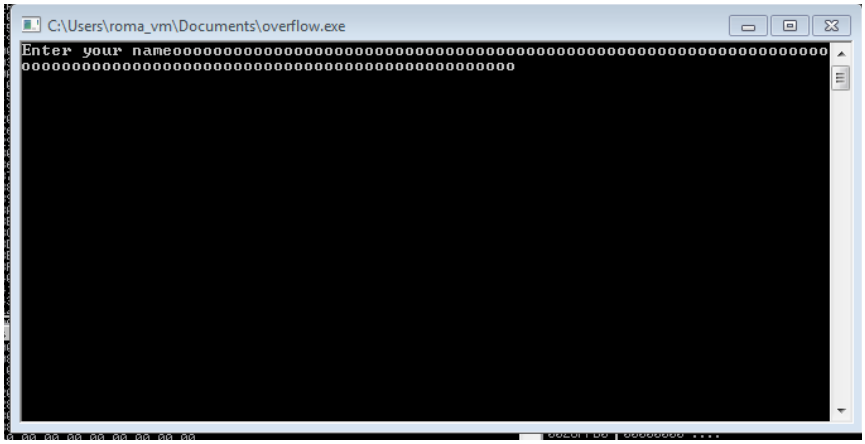


We see that the program crashes due to buffer overflow error

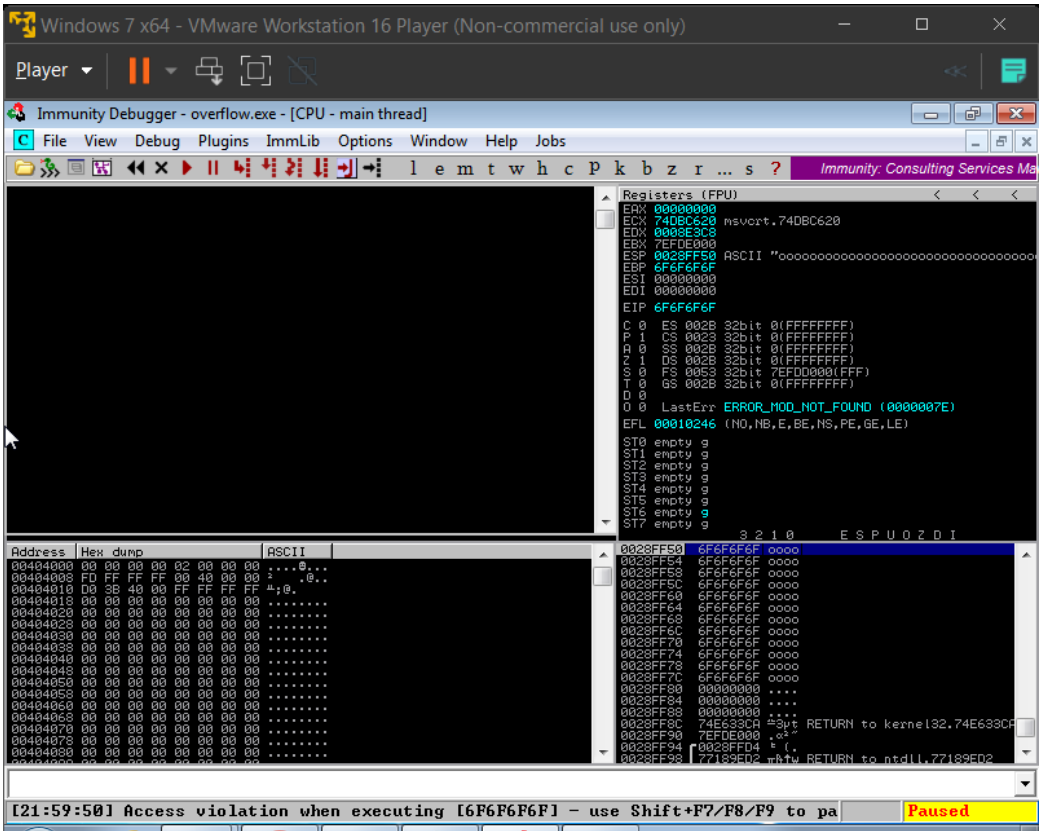


Now, open Immunity Debugger and open the executable generated above and enter invalid input

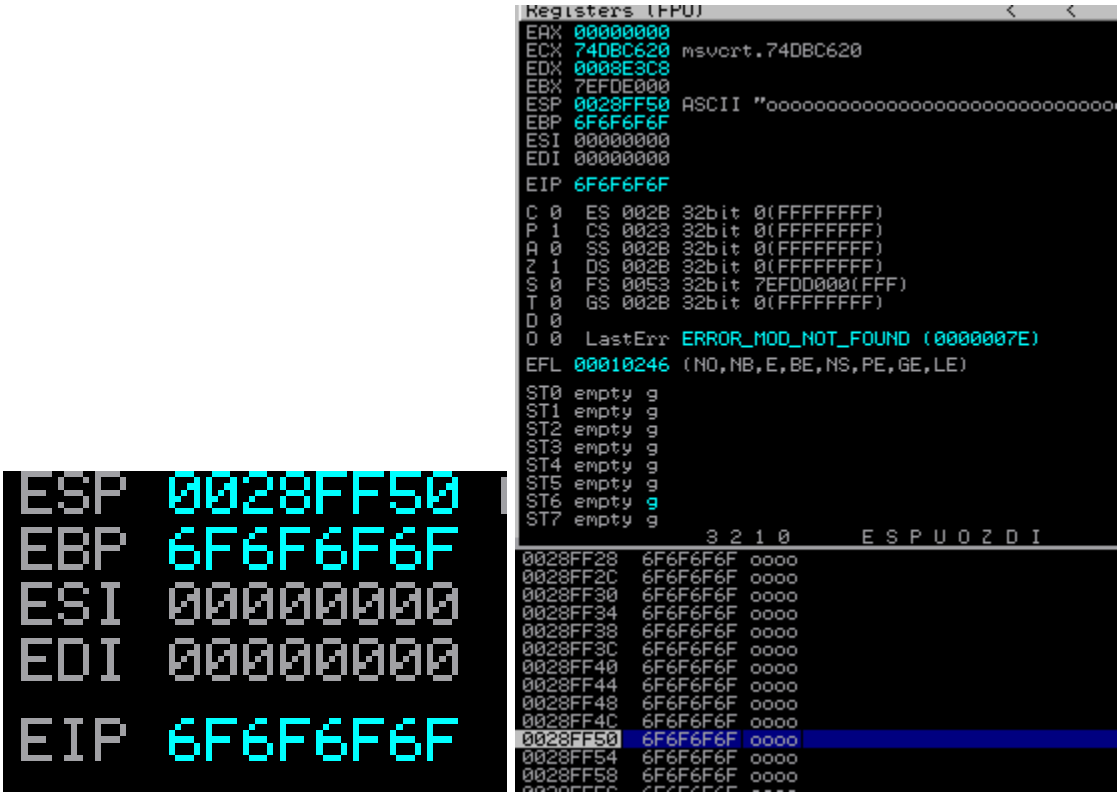




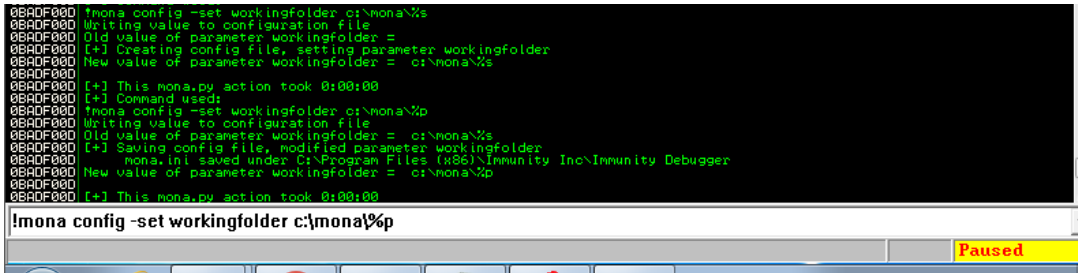
We see that the debugger stops working.



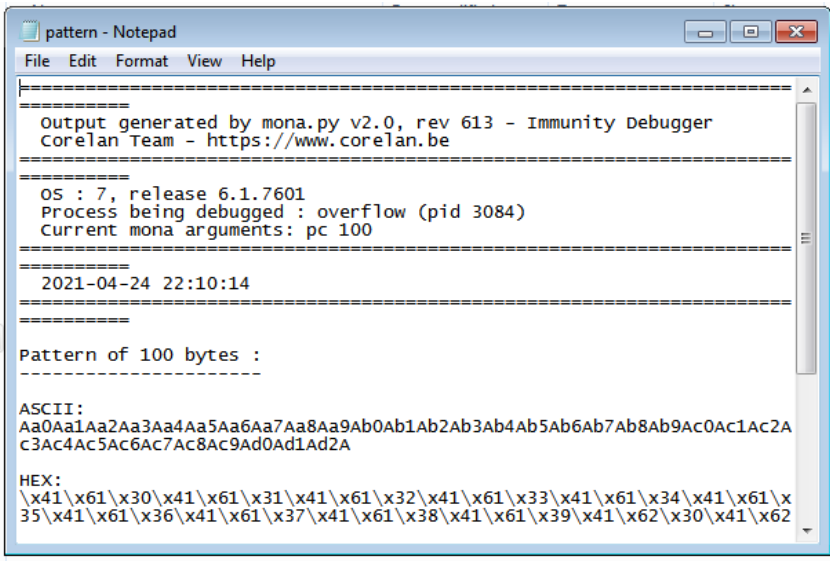
Note down the EIP value in the stack, and we see that the value is filled with “6F” which is the value of the letter “o” which we had input earlier

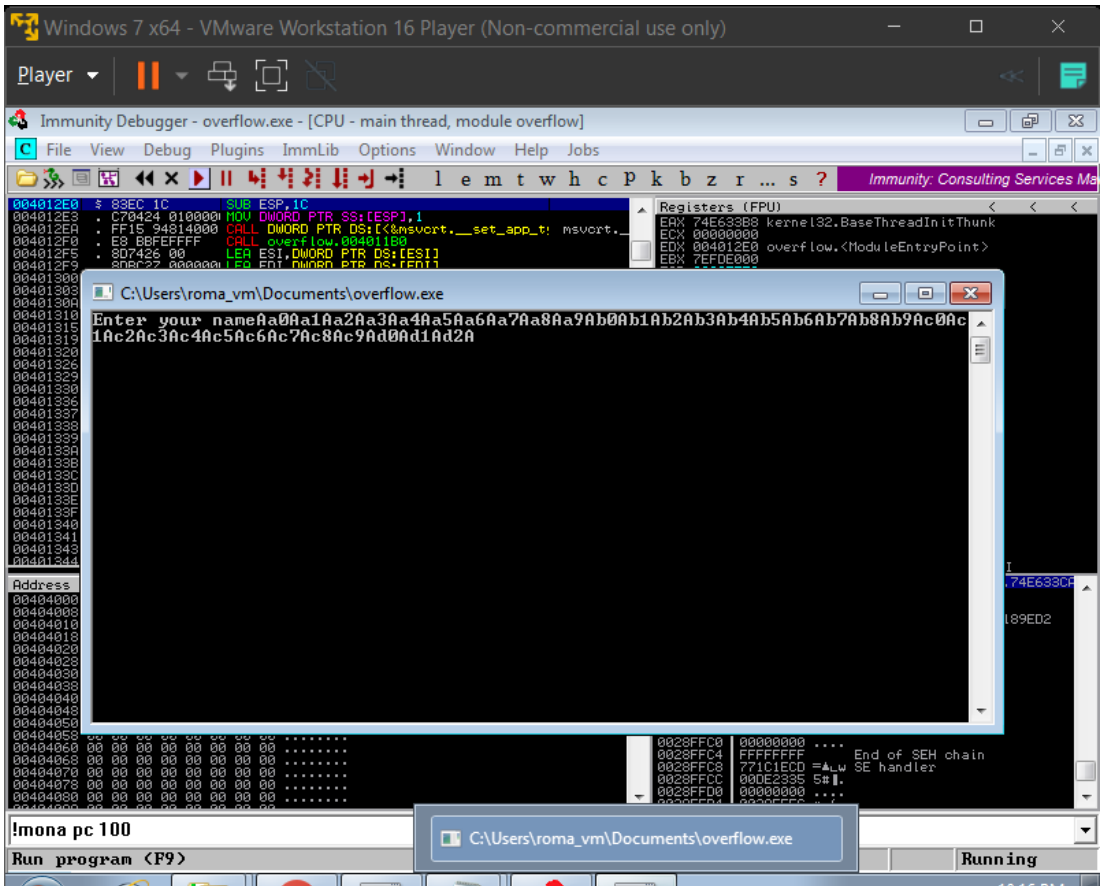


After setting up the working directory for mona, get a pattern code for 100 bytes

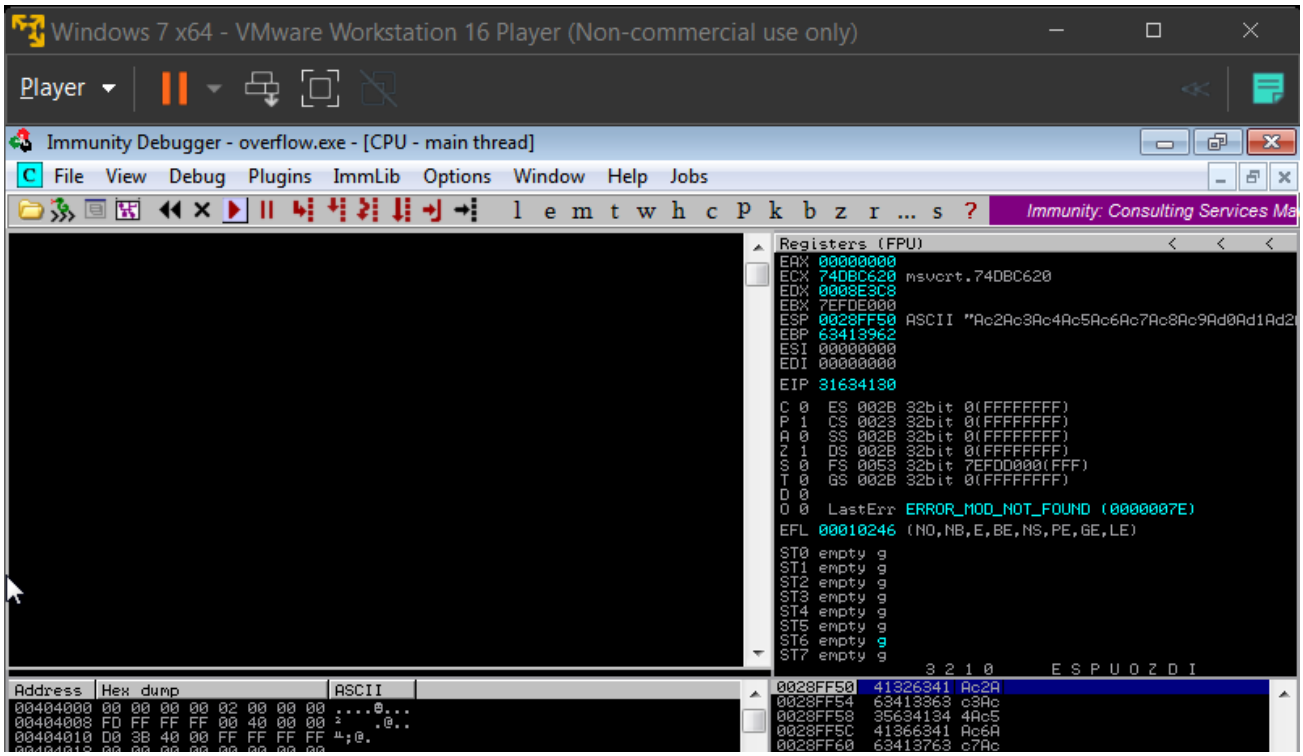


We get the pattern.txt file generated, now we can copy this pattern into the input for the executable to check and it will crash

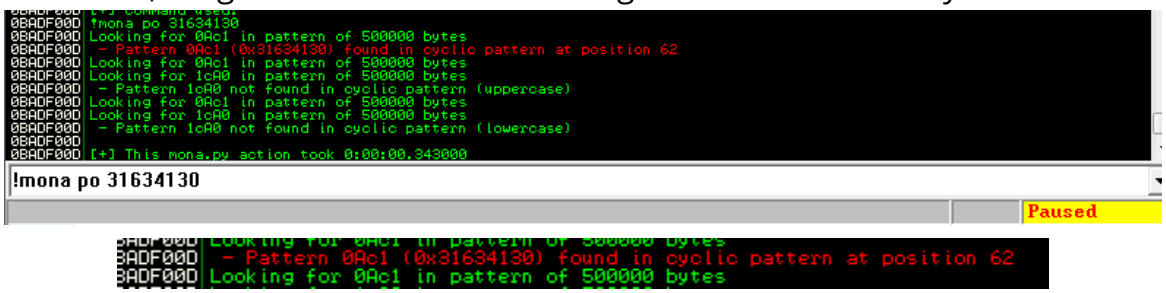




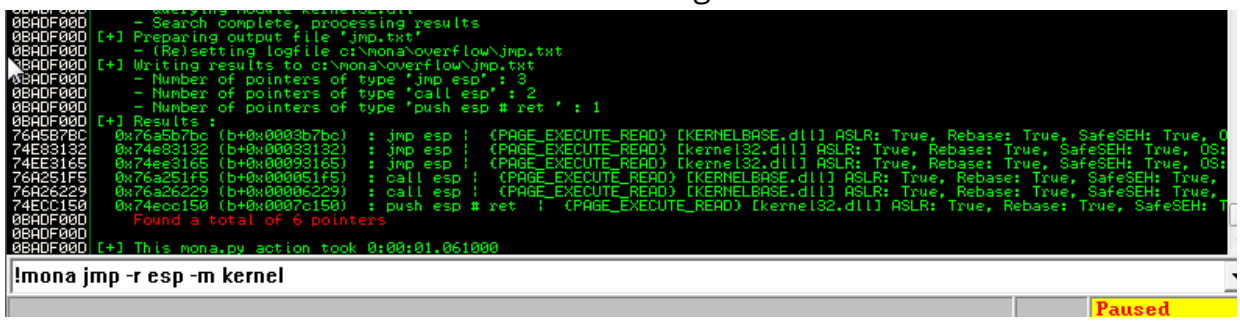
Now, when we go back to the debugger we see that there is a new value for the EIP



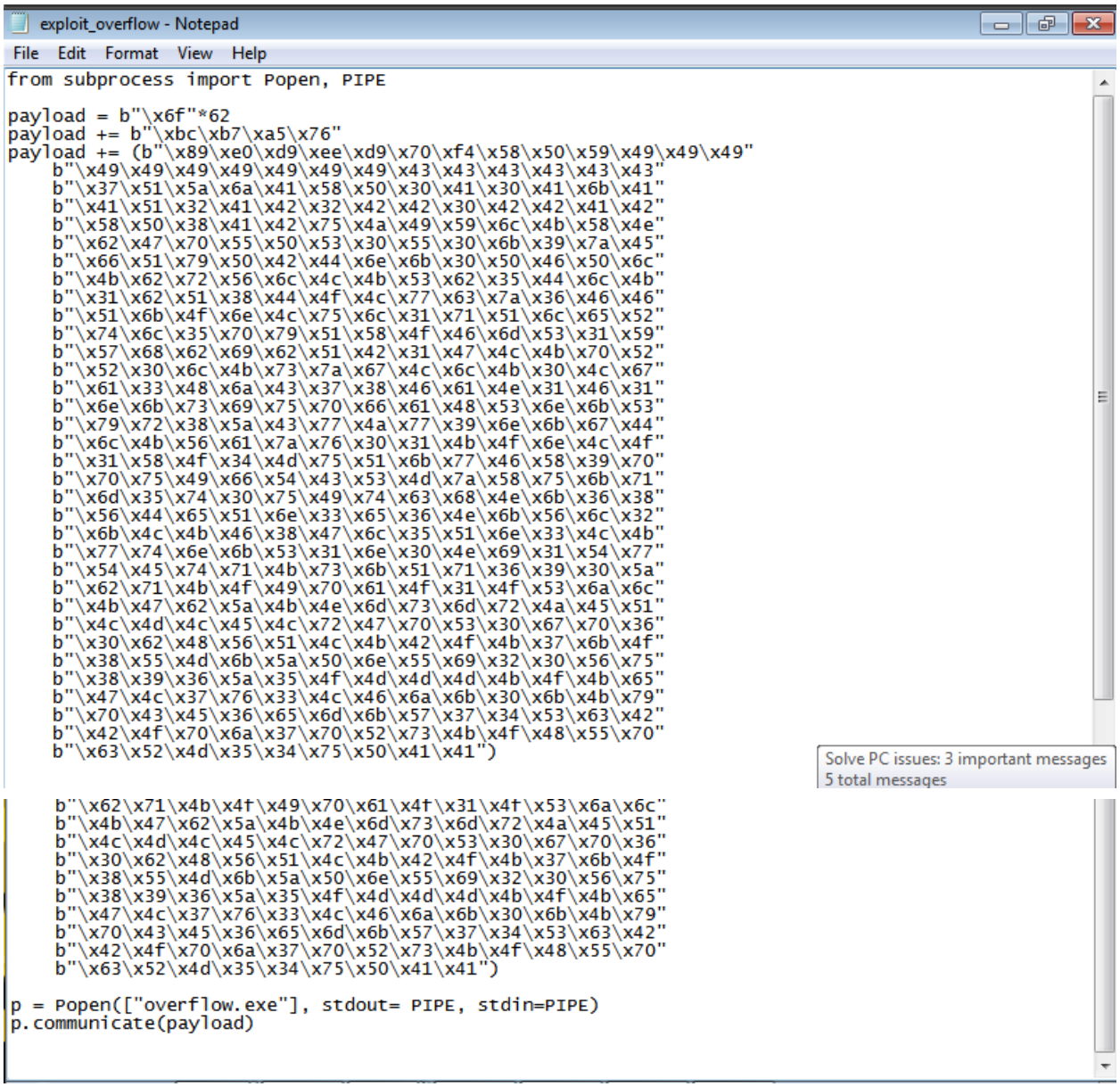
Next, we get the offset for the EIP register from its memory address



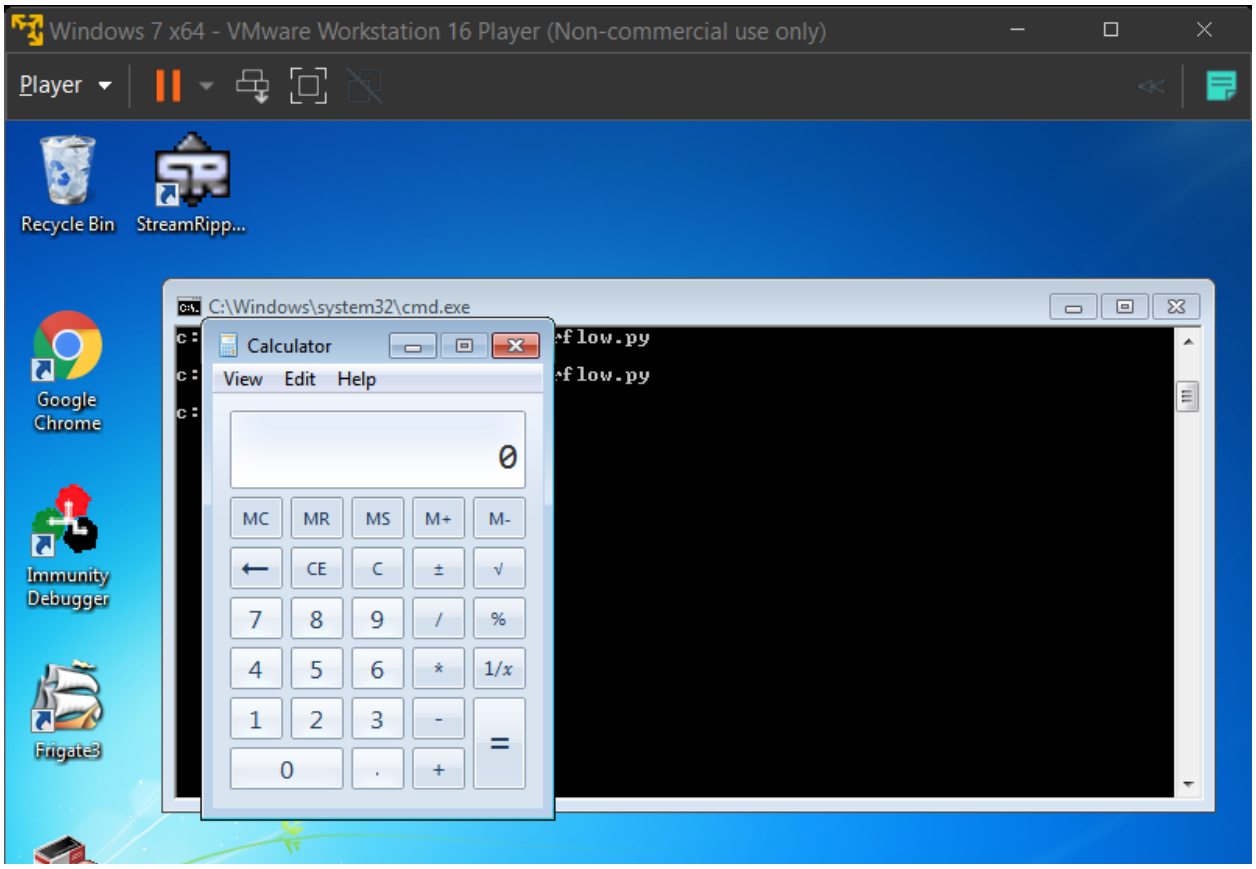
We will need 62 jump bytes, and after executing the following command we see the address of the ESP register value



Now, we need to write the exploit.
First we fill in some junk values into the buffer, then we fill in the address of the ESP register which we received from the above step. Then when we have a buffer overflow error we enter the calculator shell code



When we run the program, we see that the program crashes and the calculator opens up



The following dll shows up in the SEH chain