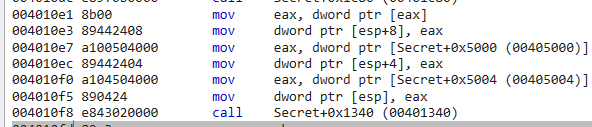
The first printed message is referenced in the main function – from address 4010e1(on my machine) the program was moving 3 variables to eax and then into the stack (using the address of esp), and the call to the main function is at address 4010f8.



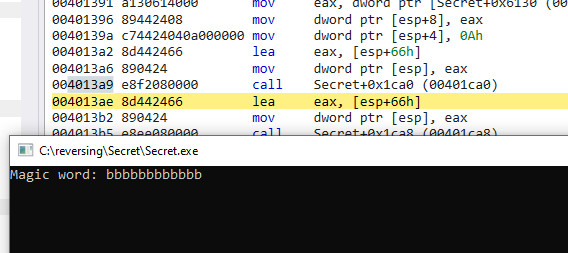
In the main function the string is referenced at address 401385:



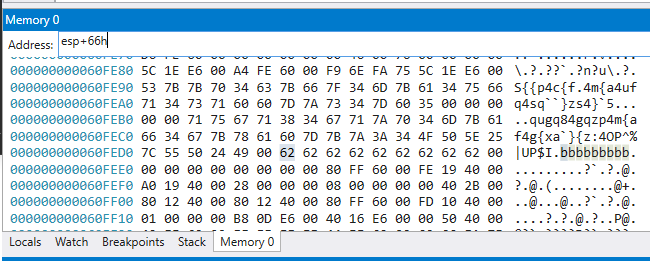
And to make sure that the address 403024 is the string’s address:



After running the program few more steps after printing the message, we see that at address 4013a9 the programs recieves input from the user.



We notice that at adress 4013ae the address of esp+66h loads into register eax, lets take a look at it’s content:



even if we write as input "bbbbbbbbbbbbbbbbbb", the program stores only the 9 first characters.

For now lets just let the program run. it doesn’t print anything else.

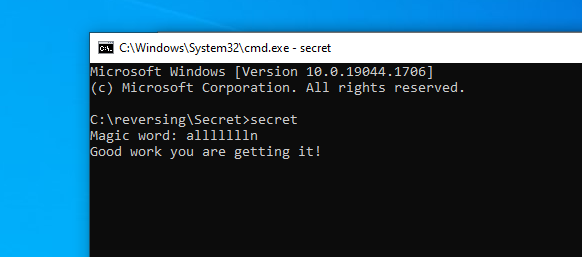
There are two jump instructions that would kick us out of the program.

Lets change eip to 401401, the address after the second jump.

then we see that there is a loop from address 4013cd to 4013eb, that’s probably where the action is.

That loop calculates the ascii sum of the first 9 chacarters, later it subtracts 34eh from the sum and it should be equal to 14h. In other words, the sum of the first 9 characters should equal to 364h. The string "allllllln" should satisfy the requirements.

As we see, it does:



The following python script imitate the program’s behaviour:

correct = "S{{p4c{f4m{a4ufq4sq``}zs4}`5"  
userinput = input("Magic Word: \n")  
if len(userinput) > 3:  
 char\_sum = 0  
 for char in userinput[:9]:  
 char\_sum += ord(char)  
 char\_sum -= 0x34e  
 char\_sum -= ord(userinput[0])  
 if char\_sum == 0x14:  
 output = ""  
 for char in correct:  
 output += chr(ord(char) ^ char\_sum)  
 print(output)

Example:

