**Crackme64**

Author: Shoham Zeharya

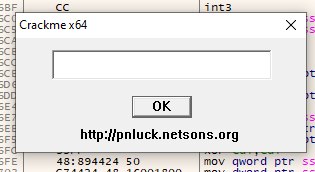
Just as a matter of convenience, I opened the executable in cff explorer went to **Optional Header** > **DllCharacteristics** > And disabled the option "**DLL can move**". This will cause the executable to load at the preset ImageBase.

When I open the executable in IDA it shows me the entry point immediately. But when i press run in x64dbg, i get an execption:

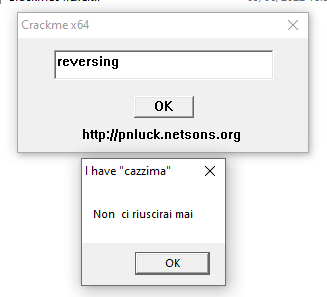


I think it might be the peb+2 value. I restart the program, then in the menu up top i open **Debug**, then **advanced options**, then **hide debugger.**

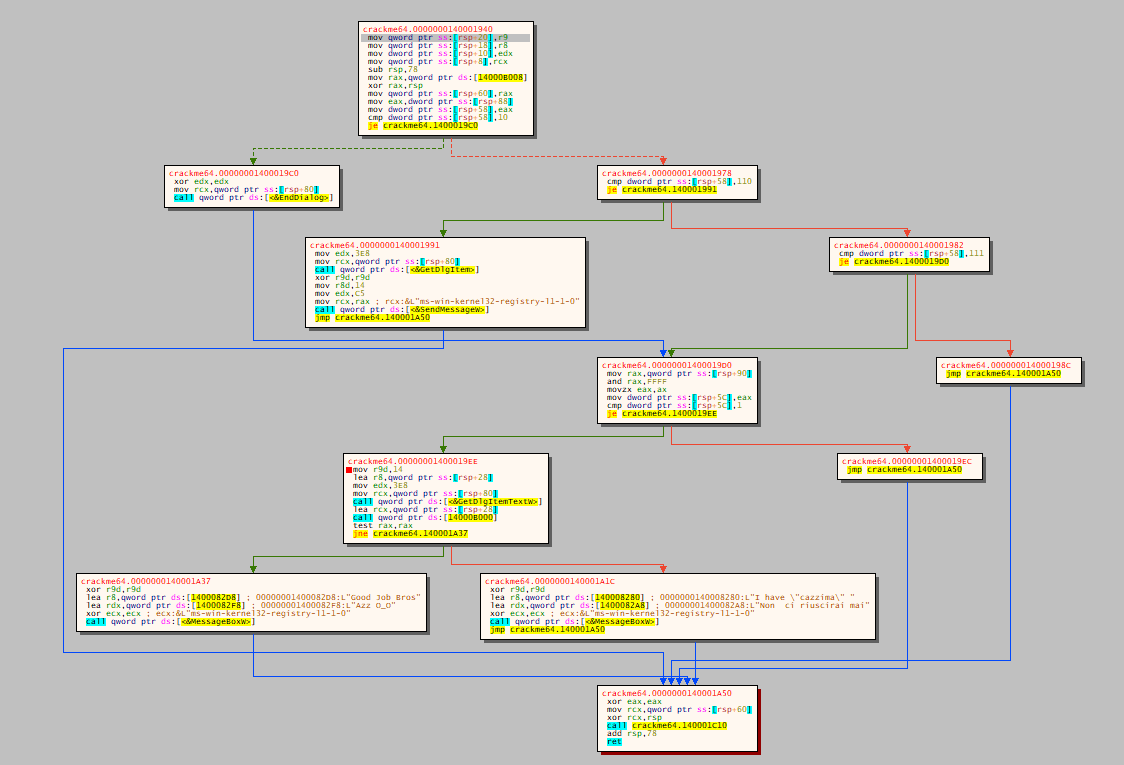
And BAM. The program started:



When giving any input and pressing ok, these is a pop up:

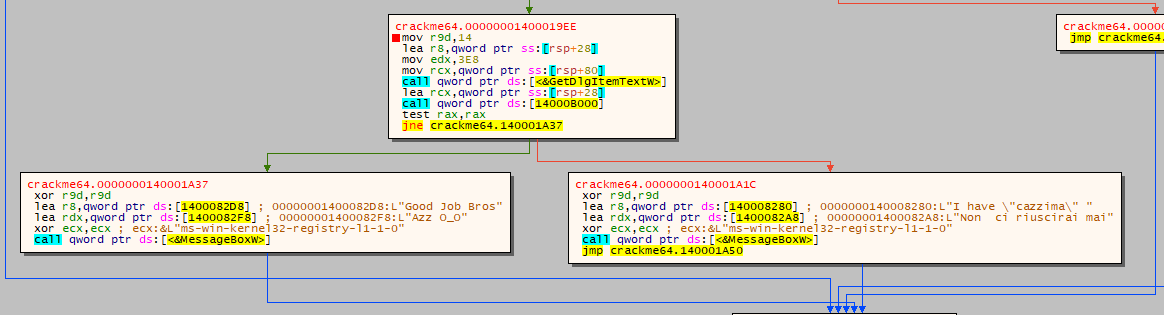


So we need to find any of the strings in the pop up. I looked up at the address of the main in IDA, and I saw that there is a call to a function named DialogBoxParamW, and one of the parameters is a function named DialogFunc (probably IDA gave that name). This function starts at 140001940. We open the fucntion and this what we see:



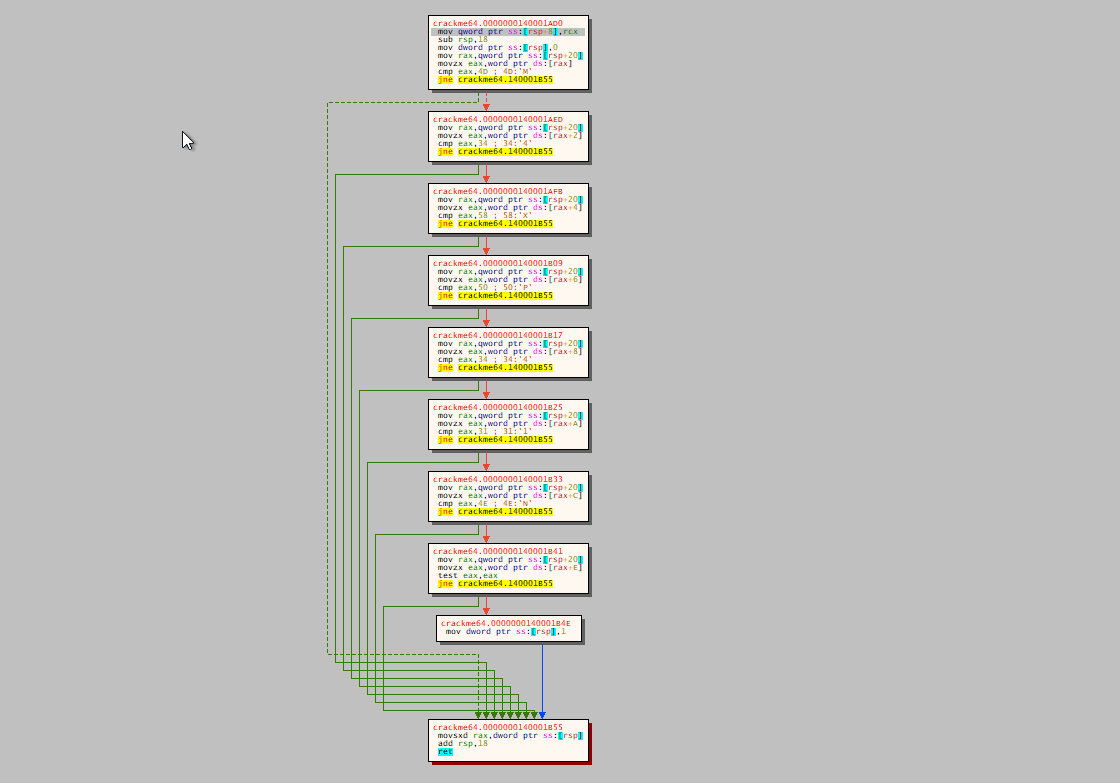
Looking at the lower section, we can seepassing the strings we saw in the pop up, and then a call for a function that creates a message box. At the other side of the test right before that a similar behaviour but with strings that look much nicer.

Let’s put a breakpoint in x64dbg. When putting the breakpoint at the beggining of the function, the window just gets stuck. I’ll put it at the beggining of the interesting section:



We can see that if we change the jne I pointed at to je, the behaviour of the program will be the complete opposite. But since we are not meant to patch, the function at 14000b000 seems a bit suspicous, because we use the return value from it to decide if the input was good.

The interesting function looks like that:

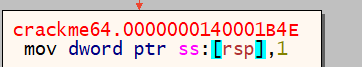


In short, it checks byte after byte that the input was "M4XP41N". If it was, the value of rax is 1, else the value of rax will be 0.

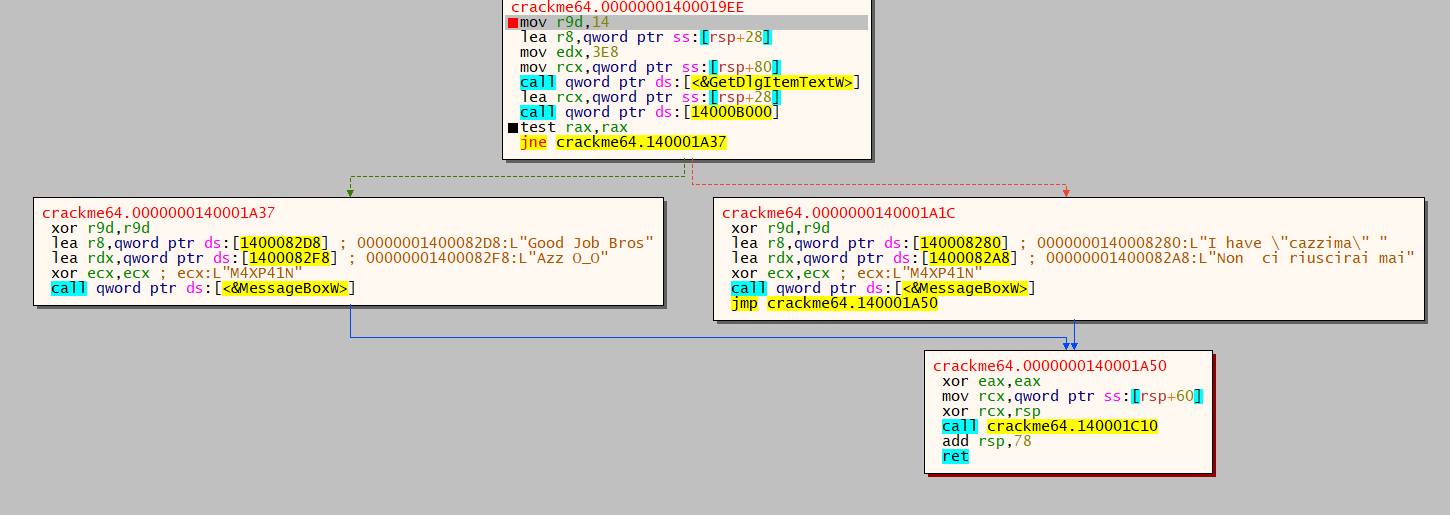
At the begginning, we move 0 to the address of rsp, which is the top of the stack:



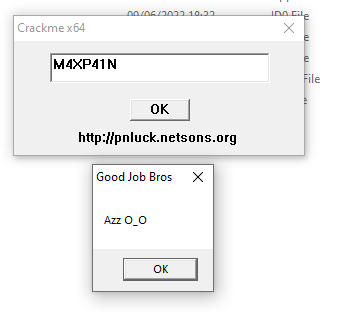
And if the input was equal we move 1 to the top of the stack:



Back to the function that called to the function that checks the input, we see that if the value of rax is 0, we go to the bad part, and if the value of rax is 1, we go to the good part.



Let’s make sure that it works on the original file:



Another one done.