

Sri Lanka Institute of Information Technology

B.Sc. Special Honours Degree in

Information Technology

Field of Specialization: Cyber Security | 4th Year

**Offensive Hacking Tactical and Strategic**

**Report on Cracking TreeDBNotes Pro**

**Using Olly Debugger**

IT16039346 – Udugahapattuwa D. M. R.

Contents

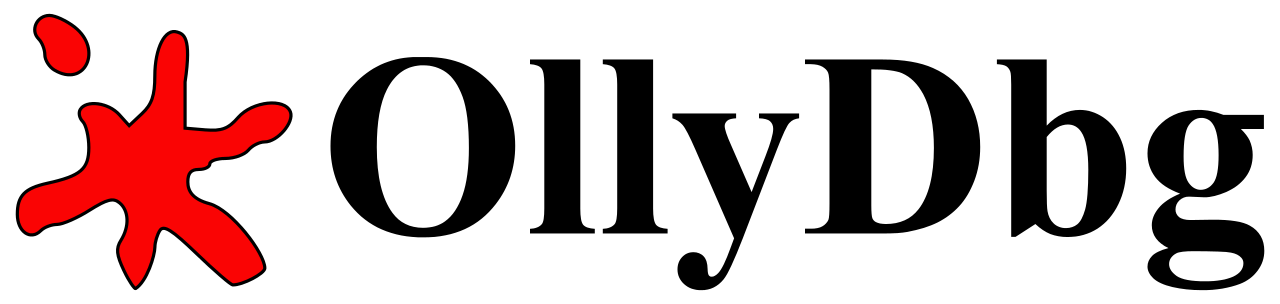
[1. What is Olly Debugger 3](#_Toc7621999)

[2. Cracking an unregistered software with OllyDbg 3](#_Toc7622000)

[2.1 The software to crack 3](#_Toc7622001)

[2.2 Procedure to get the registered version of TreeDBNotes 6](#_Toc7622002)

# 1. What is Olly Debugger

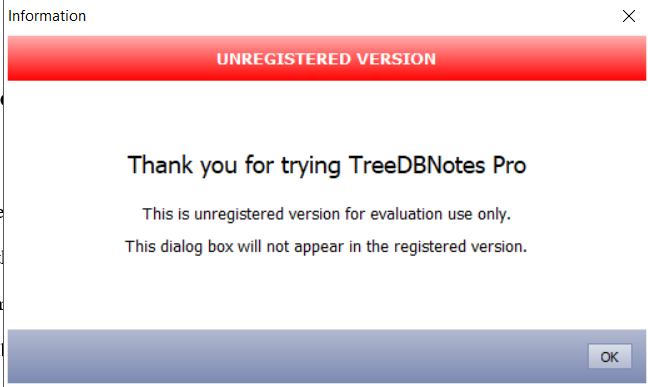


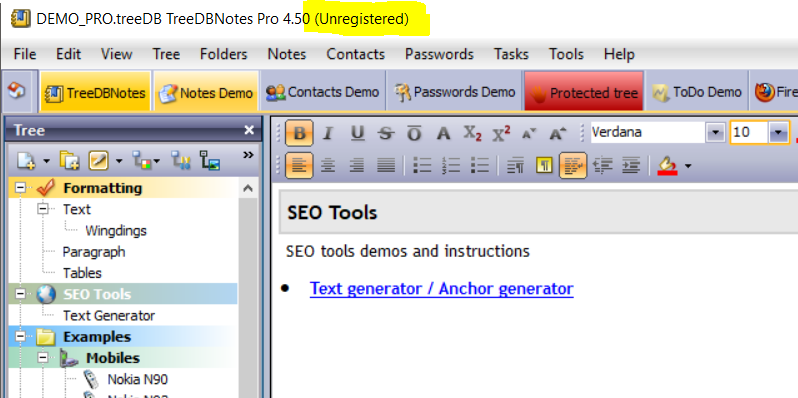
OllyDbg is a debugger for 32 Bit assembler level analysis, equipped with several tools and capabilities to carryout reverse engineering techniques on executable files. Targeting binary code analysis in Olly is extremely useful when the source code of an application is not available. It can identify procedures, switches, strings, variables, API calls, tables, registers as well as locate routines from libraries and available object files.

# 2. Cracking an unregistered software with OllyDbg

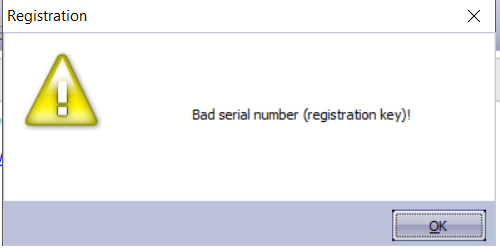
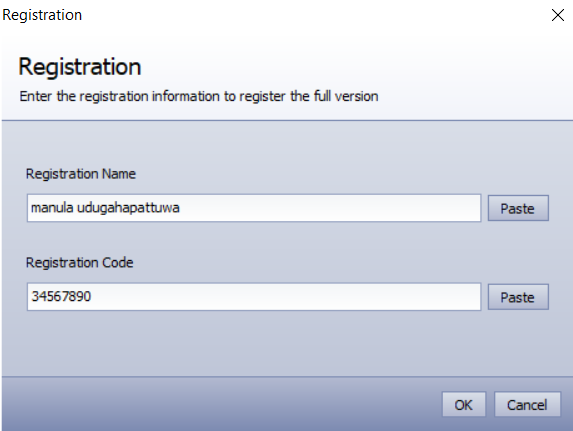
## 2.1 The software to crack

I have chosen the software named **TreeDBNotes Pro** version to crack. It lets the users play with the program with restricted capabilities if you are not a registered user. My target is to become a registered user by using reverse engineering techniques available in the debugger OllyDbg.

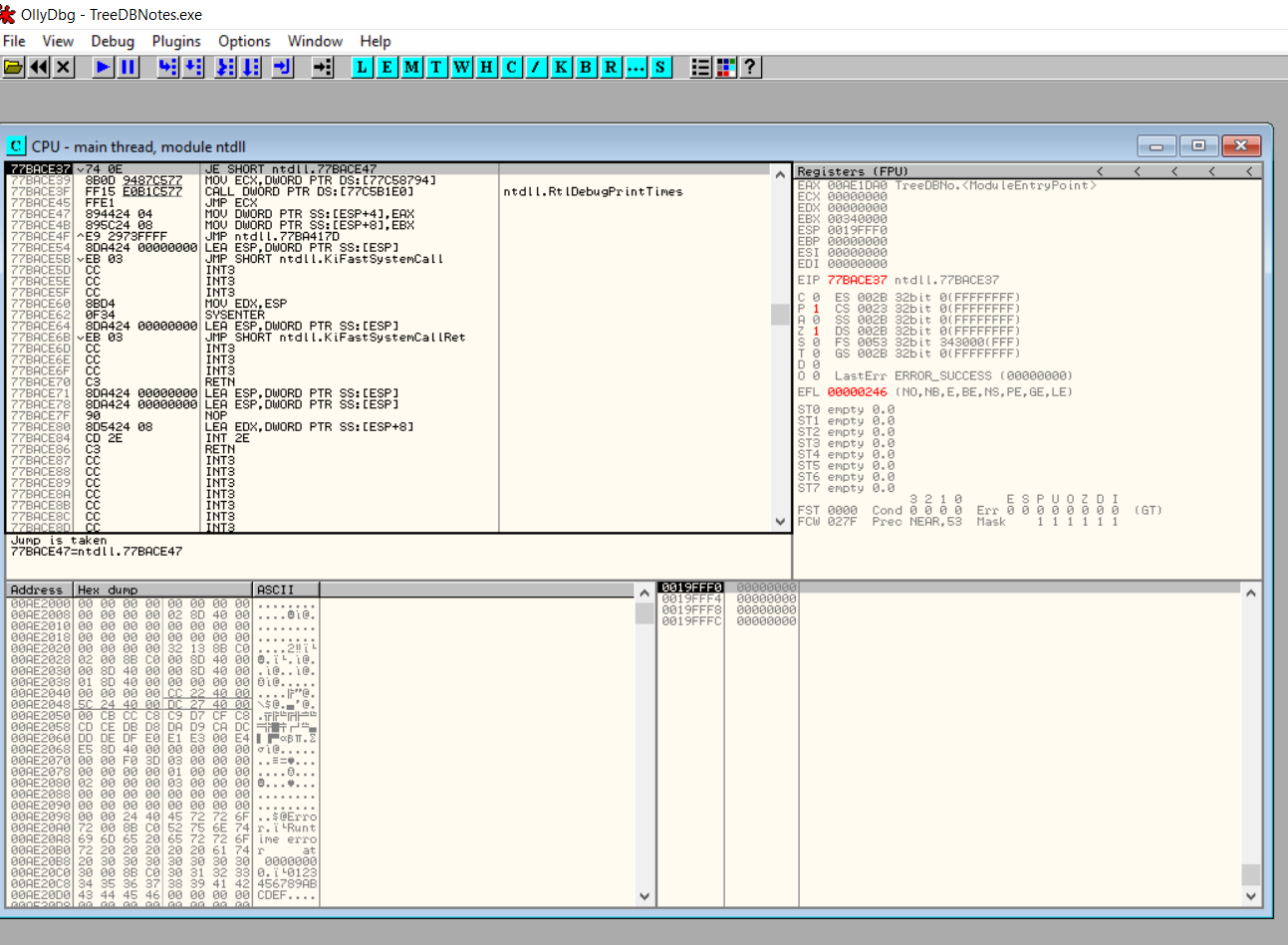




Step 01 is to play with the program till you notice the important text that may be useful when trying to crack the source code. Eg: Registration key, code, Bad serial.

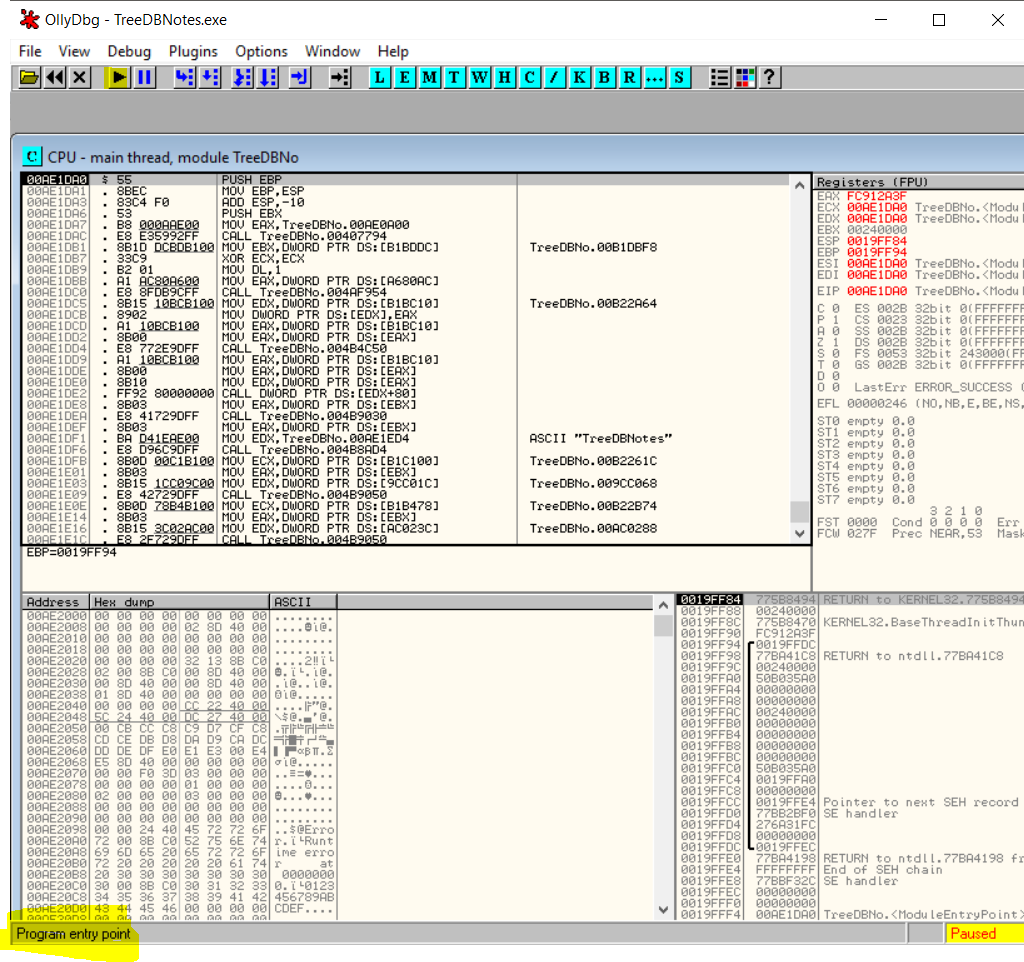


Then load the “TreeDBNotes.exe” file onto OllyDbg to analyse the code and figure out a way to activate/register the program using reverse engineering the registration process so that we have full access. By simply drag and dropping the exe file over the Olly window will open in it up.

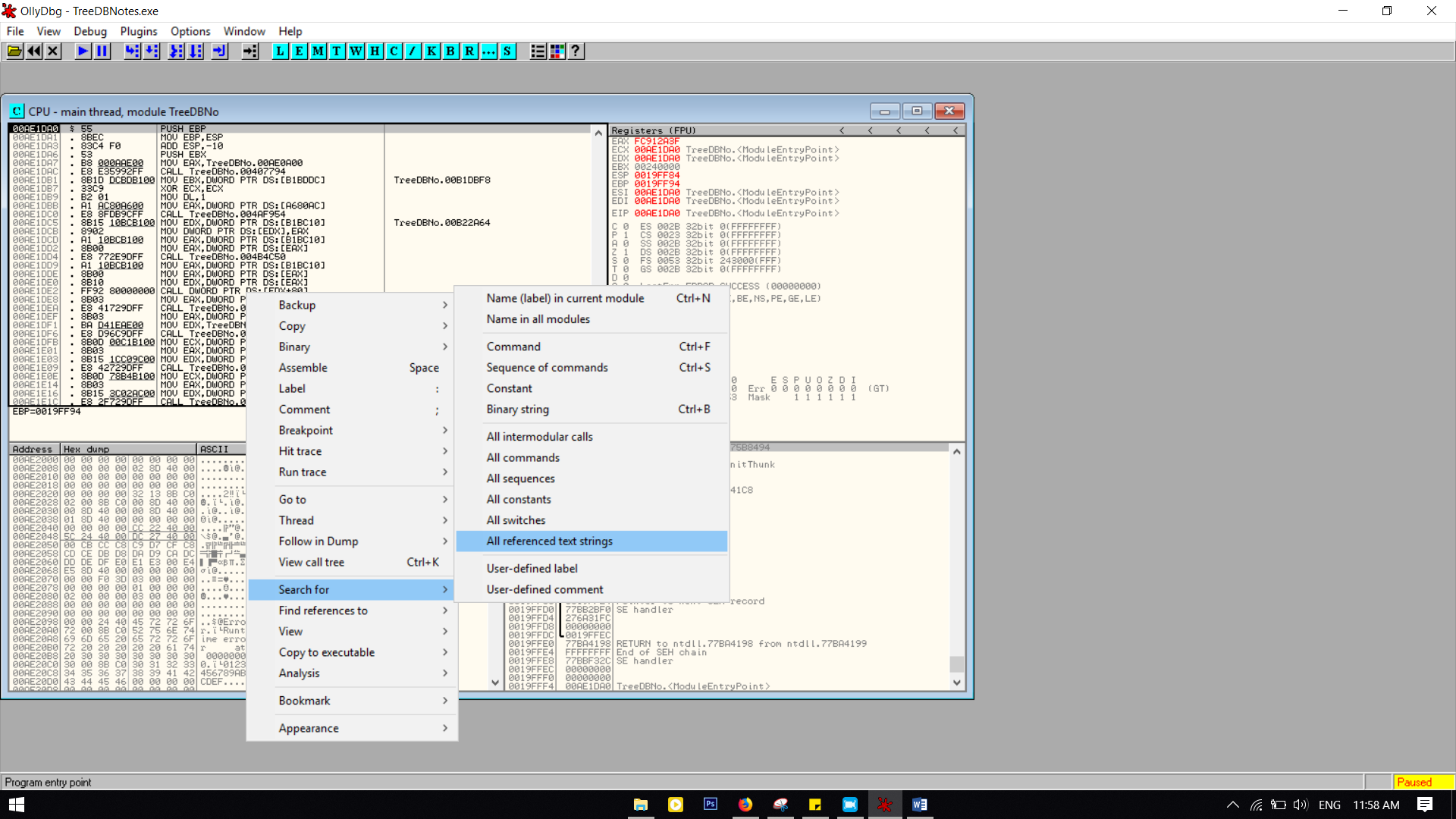


## 2.2 Procedure to get the registered version of TreeDBNotes

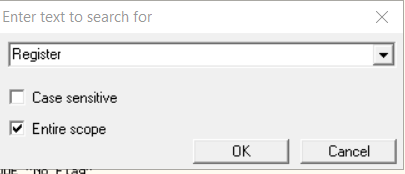
Run the program once to get it to the “Entry Point”

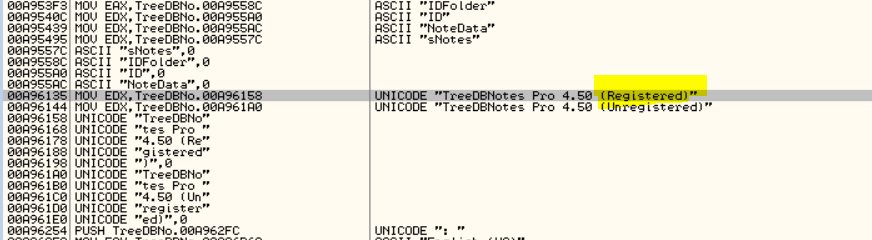


Then Right Click > Search for > All referenced text strings, wjich will open up a window with all identifiable text strings in the program.



As we identified important words before, we can now search the strings for those, to get a lead in the right direction by searching for “register” in the referenced texts.



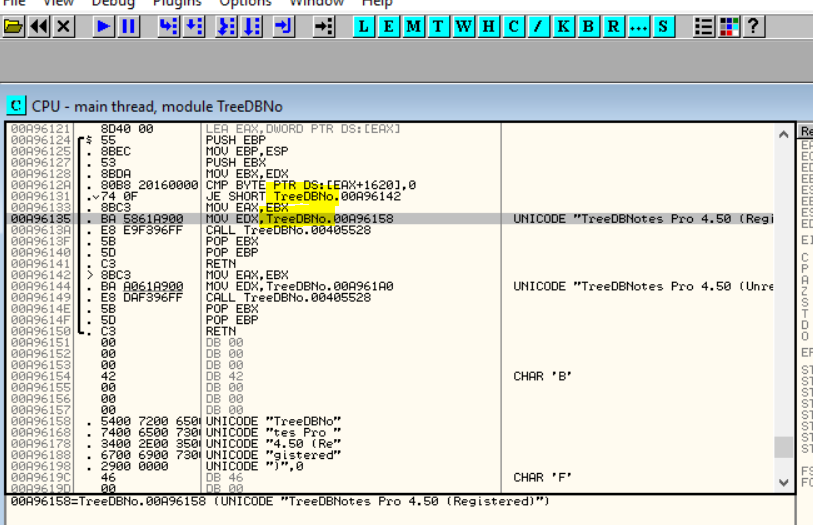
When we keep searching, we will stumble upon a lead in the right direction. 

When you double click on it, you get directed to the location where it resides in the main thread.

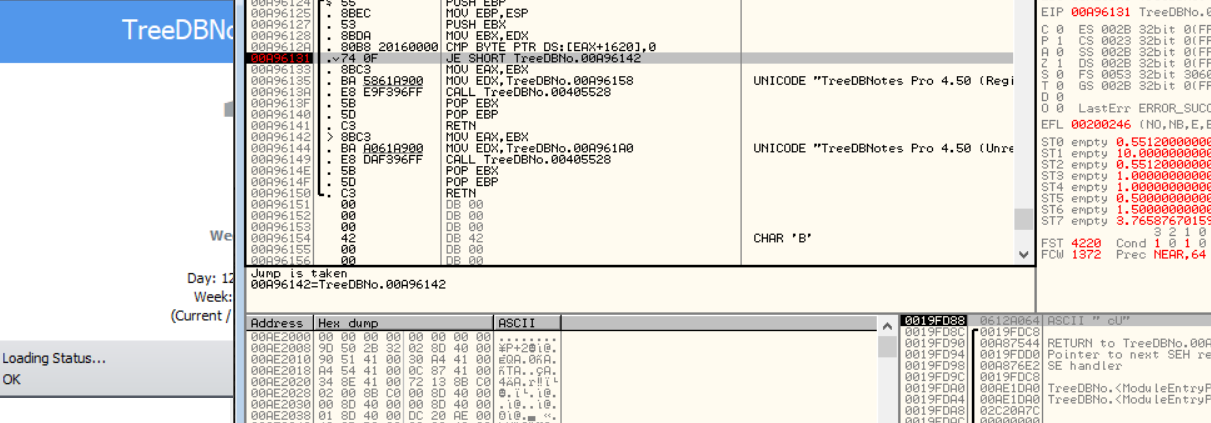
When examining closely, we can identify a conditional jump above the address we found, which has the same method “TreeDBNo”. This conditional jump takes the program back to the bad condition “unregistered”.

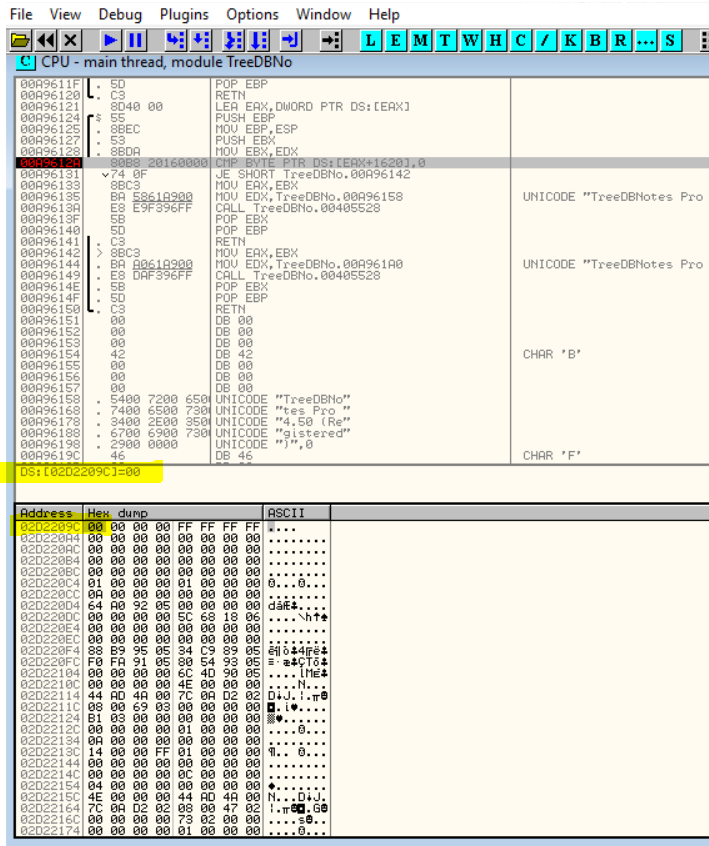
Thus add a breakpoint to 00A96131 which will pause the program when Olly reaches it, so we can stop it from jumping to the bad condition.

Right clck on the line >> Breakpoint >> Toggle or press F1

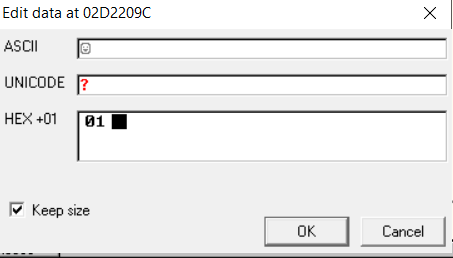


After adding the Breakpoint we can run the program again. Then we will notice that the program pause at the loading window before it shows the “unregistered” word which is probably the instance it checks if the user is registered.

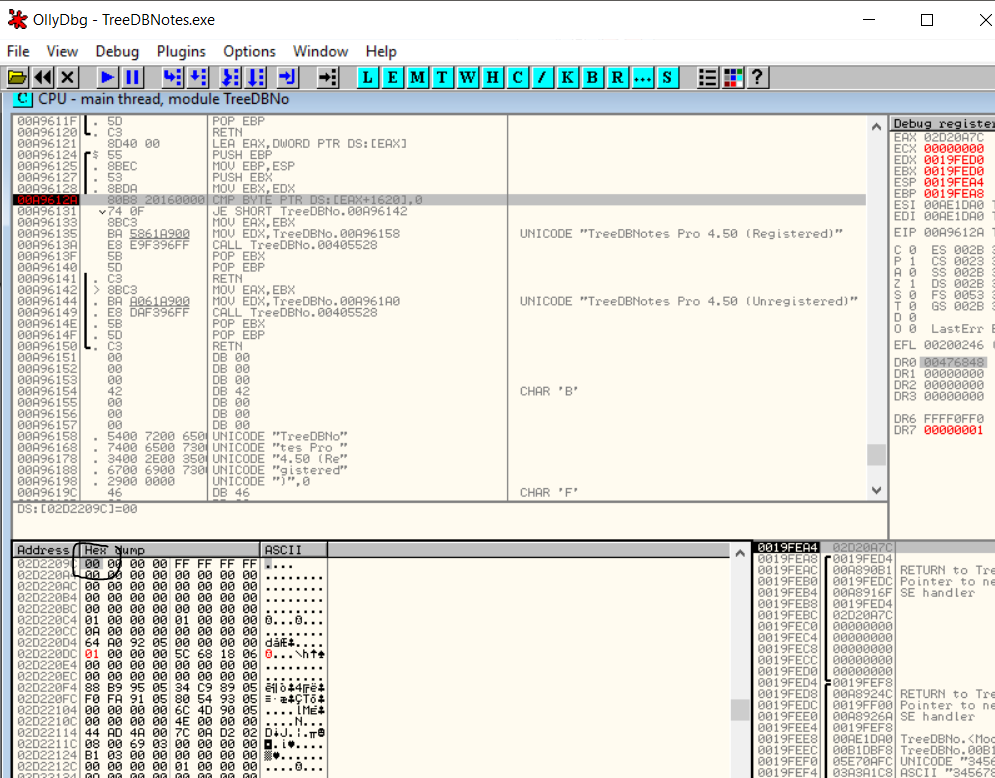
 We can notice that a compare right above the jump line. That should be where the program compares the entered registration code to the right code to determine if the user is registered or not. The variable it compares against is a global variable which is hinted from “DS” and is in the memory address [EAX+1620]



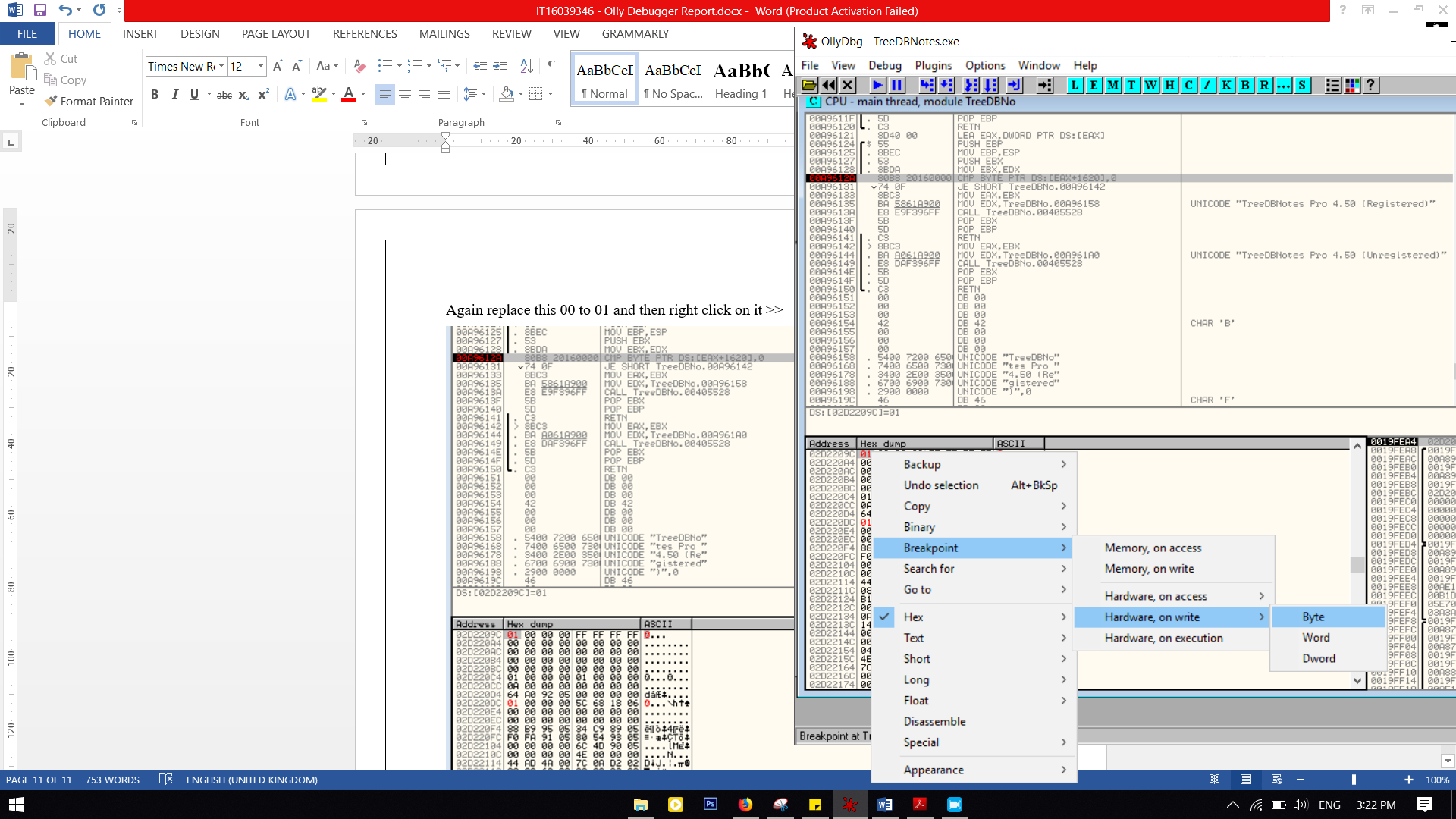
The variable resides in 02D2209. Find the memory by right clicking on the line >> Follow in Dump >> Memory address. As it is a 00 when the user is “unregistered”, anything else besides 00 will make the user “registered”. Thus let us try changing the 00 into 01 by right clicking on 00 >> Bianry >> Edit. Then type in 01 as bellow.



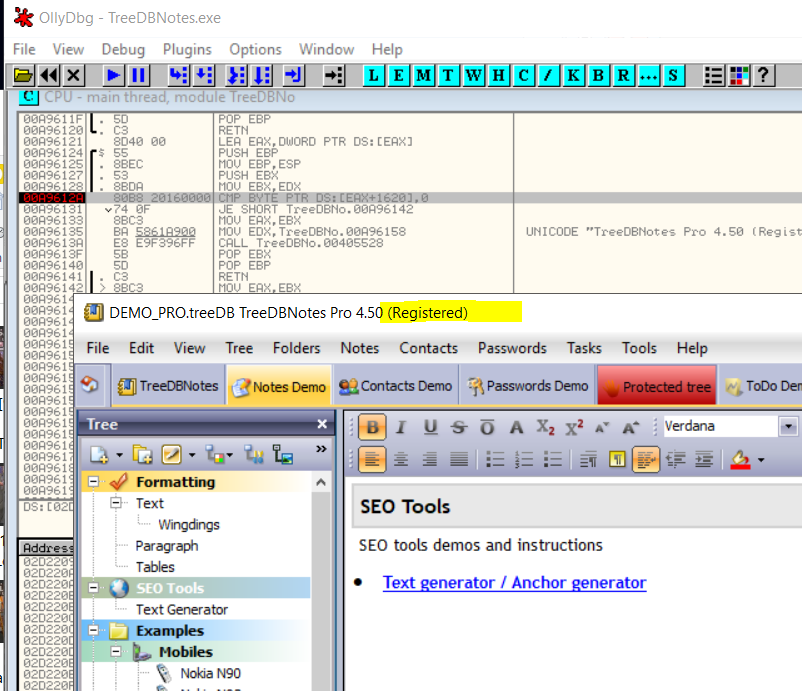
Then we can run the program and check what happens. Unfortunately, the 01 gets replaced again with 00 which means that the program has another instance where it checks if the user is registered. And if not, it resets the global variable to 00 again.

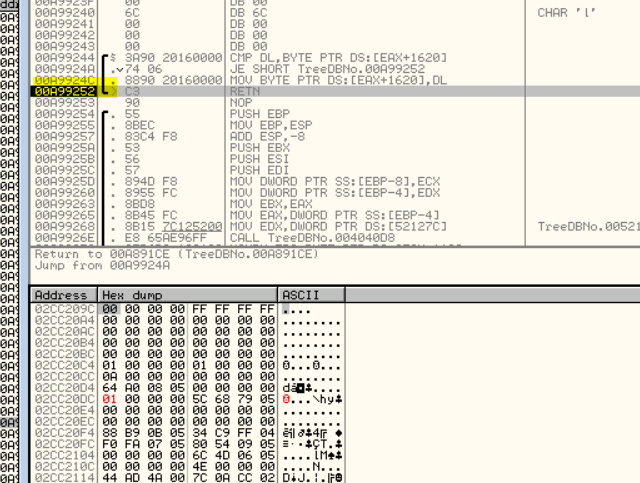


Again replace this 00 to 01 and then right click on it >> Breakpoint >> Hardware, on write >> Byte. This is done so that Olly will pause the execution before it replaces the 01 again with a 00. We are trying to catch the second instance which checks for registration.



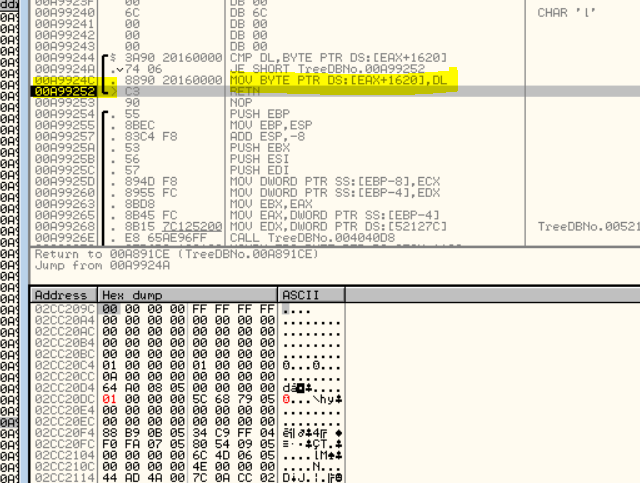
Now when you run the program, it says “registered” as we have stopped execution before the byte turns back into 00. But we still have to figure the second instance, to make it permanent.



When we play the program again, it gets paused at a different memory location. This is because the hardwear breakpoint worked as it should be. 

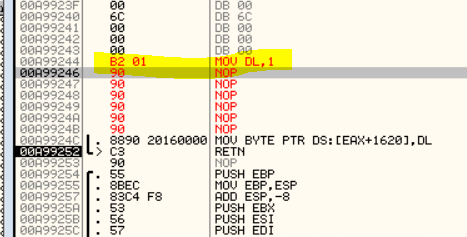
The CMP hints that the program compares the EAX+1620 memory address again to check if registered. Thus, this is the second routine where the 01 gets converted back to 00 making it an unregistered product.

What we can do is to change the replacing mechanism so it replaces a 01 rather than 00.



To do that, we should edit the compare and jump instructions. Right click on compare and jump instructions >>Binary >> Fill with NOPs. Which we will then re-write over the MOV and Return statements.

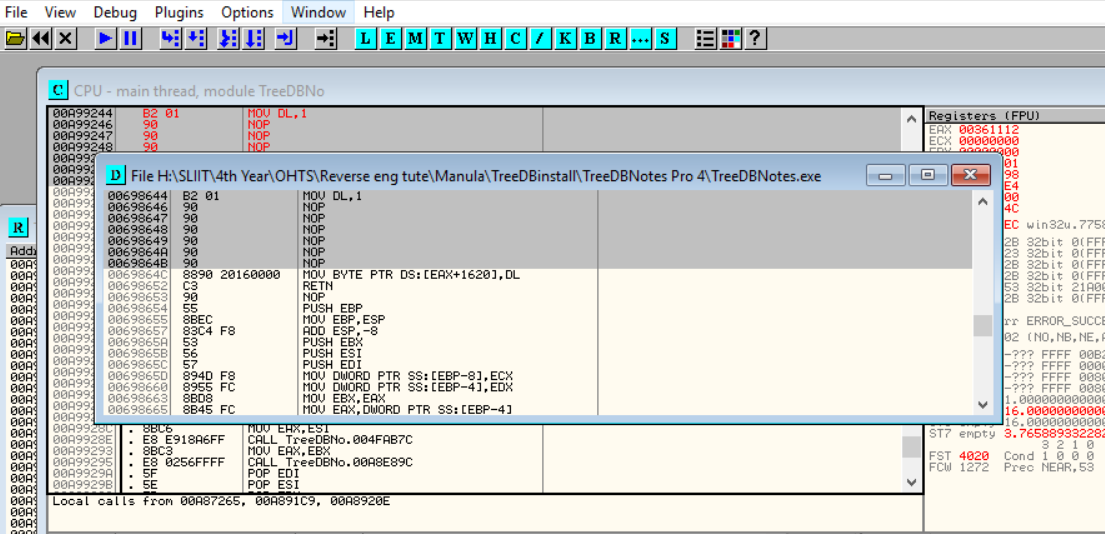
Double click on the first NOP and assemble it with “MOV DL,1” so it will add a 1 when it’s called. Now it will look like in the bellow image.



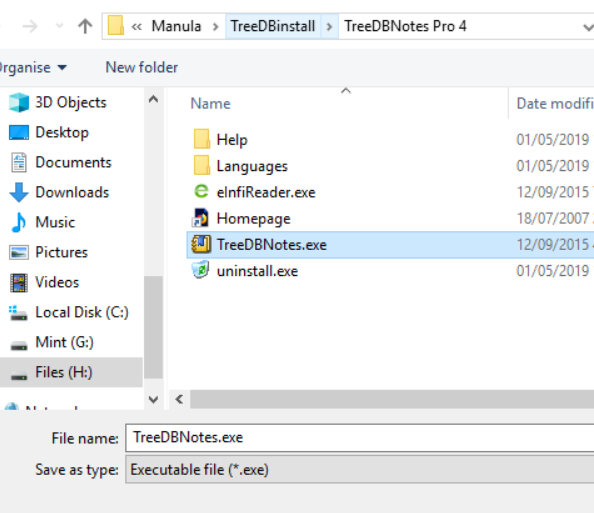
Now all we have to do is to play the application so it will save the 01 as the global variable which in return makes it think that the program is registered.

Now we have to copy all changes to the original executable.

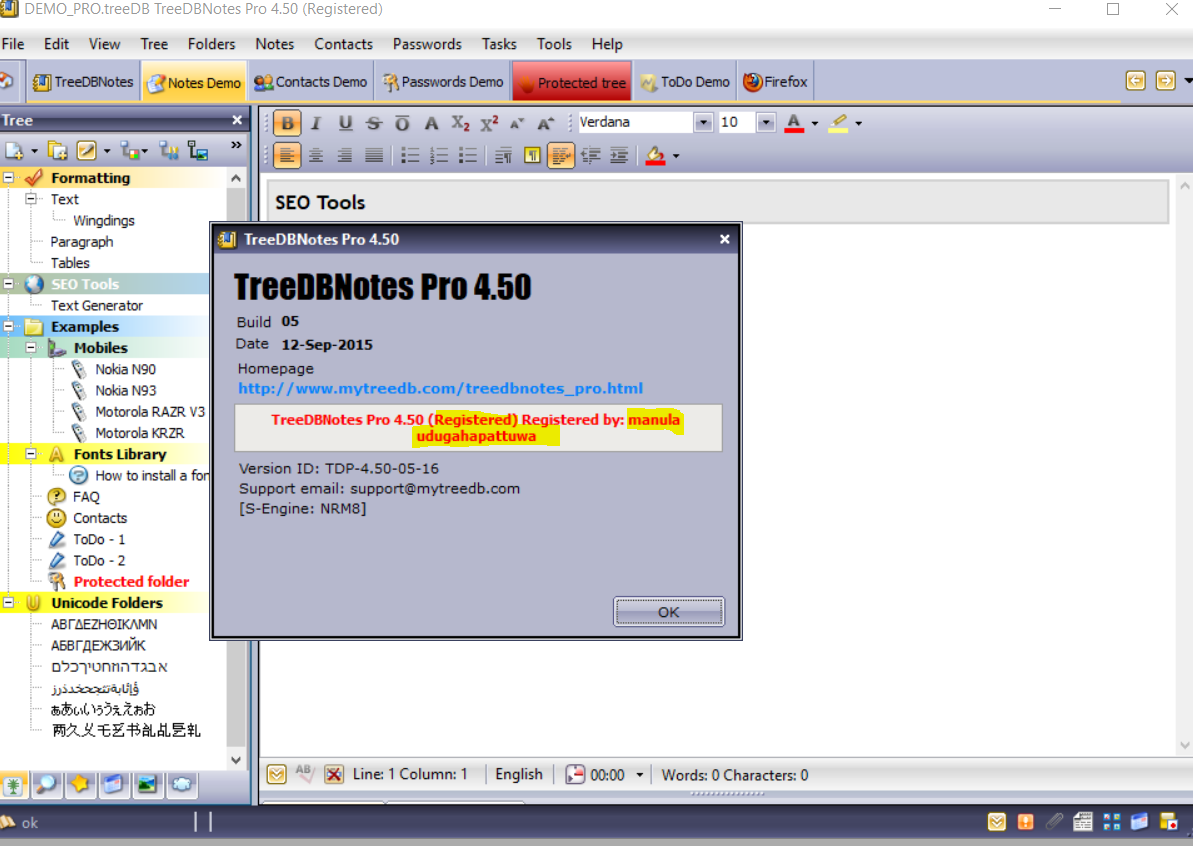
Right click>> copy to executable>> all modifications.



Right Click on the window appeared >> save file >> replace the executable of TreeDBNotes Pro 4 >> Save.



Now close Olly, run the application and check About, to find that the product is registered in your name!!



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*