



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.

Please find the **walkthrough video** of this report at:

<https://drive.google.com/drive/folders/1bgSnKAd7piSbJBQVnZ4nid2XzzDnsyLY?usp=sharing>

Contents

1. What is Olly Debugger	3
2. Cracking an unregistered software with OllyDbg	3
2.1 The software to crack	3
2.2 Procedure to get the registered version of TreeDBNotes	6

1. What is Olly Debugger

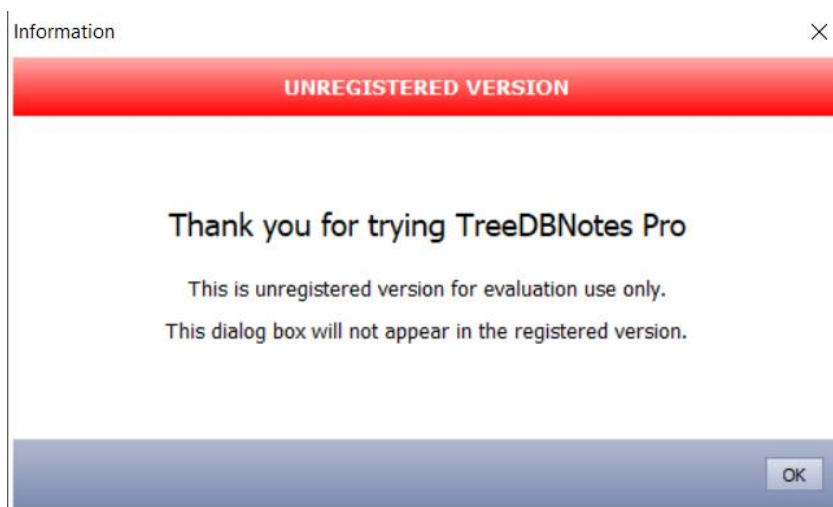


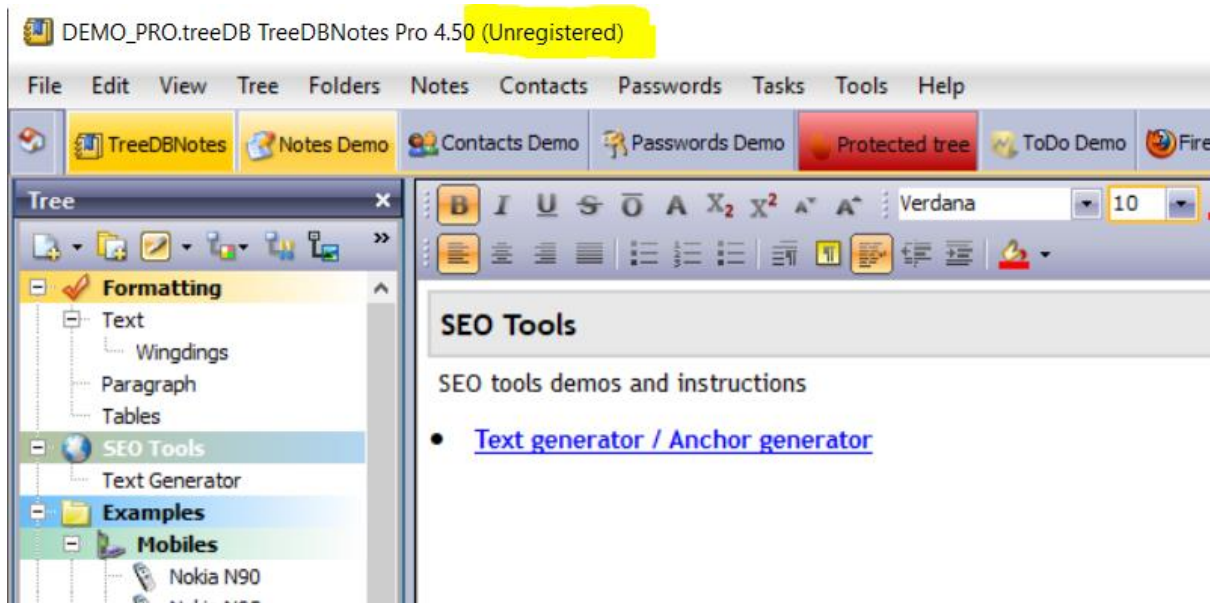
OllyDbg is a debugger for 32 Bit assembler level analysis, equipped with several tools and capabilities to carryout reverse engineering techniques on executable files. Targeting binary code analysis in Olly is extremely useful when the source code of an application is not available. It can identify procedures, 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

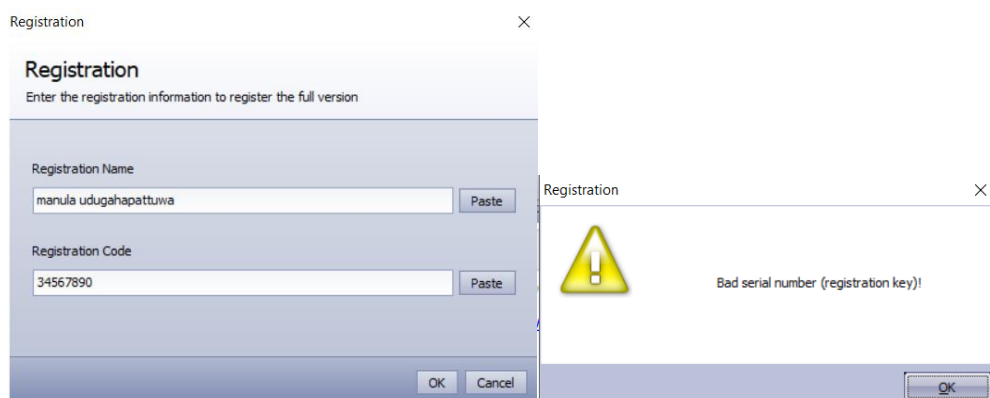
2.1 The software to crack

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.

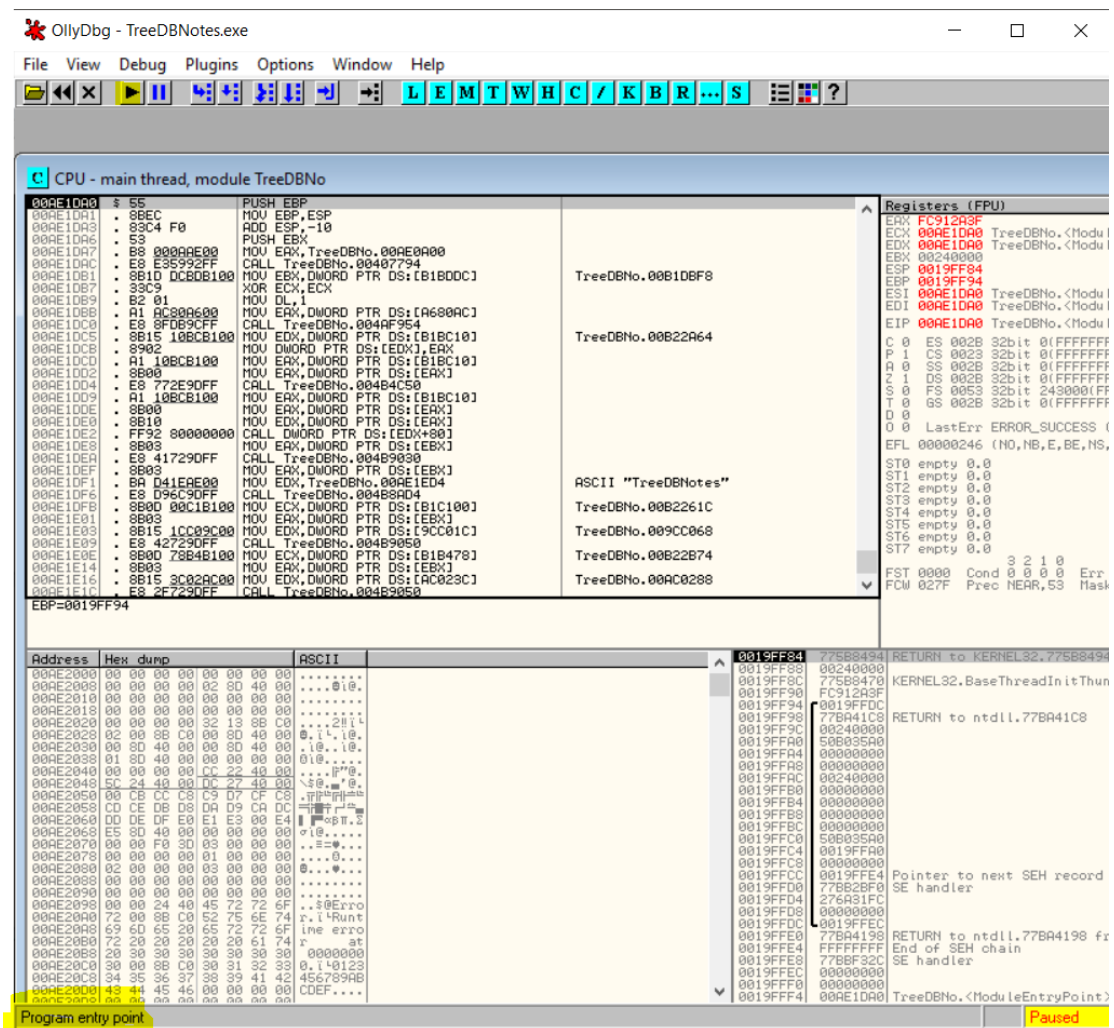


Then load the “TreeDBNotes.exe” file onto OllyDbg to analyse the code and figure out a way to activate/register the program using reverse engineering the registration process so that we have full access. By simply drag and dropping the exe file over the Olly window will open in it up.



2.2 Procedure to get the registered version of TreeDBNotes

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



OllyDbg - TreeDBNotes.exe

File View Debug Plugins Options Window Help

CPU - main thread, module TreeDBNo

Address	Hex dump	ASCII
004E1DA0	55	
004E1DA1	8BEC	
004E1DA3	83C4 F0	
004E1DA6	53	
004E1DA7	8B 00000000	
004E1DAC	E8 E35992FF	
004E1DB1	8B10 DC000100	
004E1DB7	33C9	
004E1DB9	B2 01	
004E1DBB	A1 AC000000	
004E1DC0	E8 8F0B9CFF	
004E1DC5	8B15 10BC0100	
004E1DCB	8902	
004E1DCD	A1 10BC0100	
004E1DD2	8B00	
004E1DD4	E8 772E90FF	
004E1DD9	A1 10BC0100	
004E1DDE	8B00	
004E1DE0	8B10	
004E1DE2	FF92 80000000	
004E1DE8	8B00	
004E1DEA	E8 417290FF	
004E1DEF	8B03	
004E1DF1	BA D41E0E00	
004E1DF6	E8 427290FF	
004E1DFB	8B00 00C1B100	
004E1E01	8B03	
004E1E03	8B15 1CC03C00	
004E1E09	E8 427290FF	
004E1E0E	8B00 78B4B100	
004E1E14	8B03	
004E1E16	8B15 3C020C00	
004E1E1C	E8 2F7290FF	

EBP=0019FF94

Registers (FPU)

Register	Value
EAX	0019FF94
ECX	004E1DA0
EDX	004E1DA0
ESP	00240000
EBP	0019FF94
ESI	004E1DA0
EDI	004E1DA0
EIP	004E1DA0
C 0	ES 002B 32bit 0(FFFFFFFF)
P 1	CS 002B 32bit 0(FFFFFFFF)
N 0	SS 002B 32bit 0(FFFFFFFF)
Z 1	DS 002B 32bit 0(FFFFFFFF)
S 0	FS 0053 32bit 243000(FF)
T 0	GS 002B 32bit 0(FFFFFFFF)
D 0	
0 0	LastErr ERROR_SUCCESS (
EFL	00000246 (NO,NB,E,BE,NS,
ST0	empty 0.0
ST1	empty 0.0
ST2	empty 0.0
ST3	empty 0.0
ST4	empty 0.0
ST5	empty 0.0
ST6	empty 0.0
ST7	empty 0.0
FST	0000 Cond 0 0 0 0 Err
FCW	027F Prec NEAR,S3 Mask

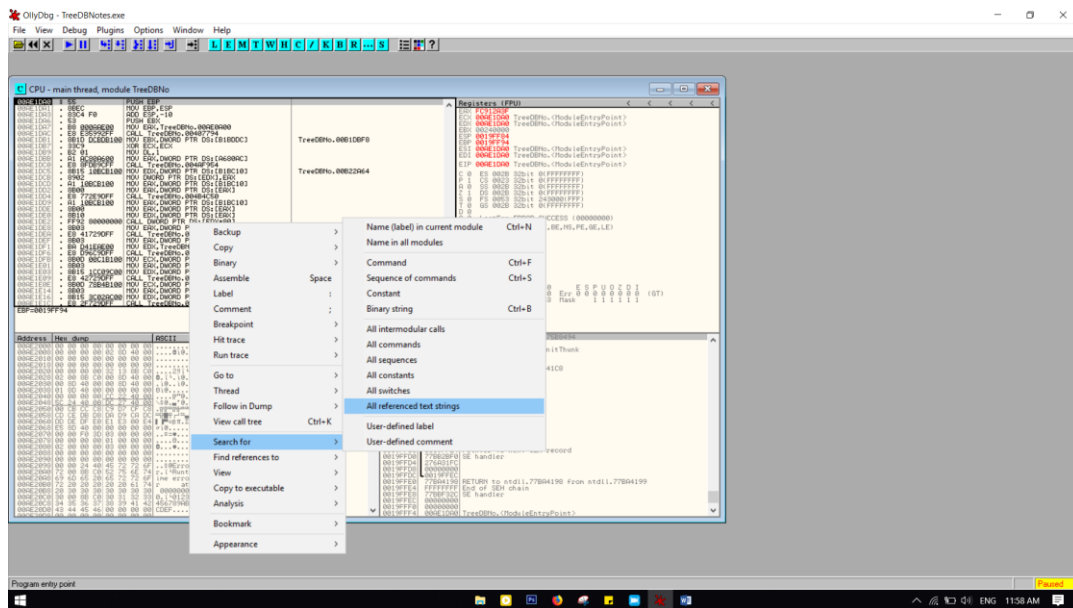
Address Hex dump ASCII

004E2000 00 00 00 00 00 00 00 00
004E2008 00 00 00 00 02 80 40 00010.
004E2010 00 00 00 00 00 00 00 00
004E2018 00 00 00 00 00 00 00 00
004E2020 00 00 00 00 32 13 58 C0 ...211.
004E2028 02 00 00 C0 00 00 40 00 0.1.10.
004E2030 00 80 40 00 00 80 40 00 010.10.
004E2038 01 80 40 00 00 00 00 00 010.....
004E2040 00 00 00 CC 22 40 00P0.
004E2048 5C 24 40 00 CC 22 40 00 ...S0.9.
004E2050 00 CE CC C5 C9 D7 CF C8 ...
004E2058 CD CE D8 D8 DA D9 CA DC ...
004E2060 DD DE DF E0 E1 E3 00 E4 ...
004E2068 E5 8D 40 00 00 00 00 00 010.....
004E2070 00 00 F0 3D 03 00 00 00 ...E...
004E2078 00 00 00 00 01 00 00 00 ...0...
004E2080 02 00 00 00 03 00 00 00 0...0...
004E2088 00 00 00 00 00 00 00 00
004E2090 00 00 00 00 00 00 00 00
004E2098 00 00 24 40 45 72 6F ...0Erro
004E20A0 72 00 8B C0 52 75 6E 74 r.i'Runt
004E20A8 69 6D 65 20 65 72 6F line erro
004E20B0 72 20 20 20 20 61 74 ... at
004E20B8 20 30 30 30 30 30 30 30 00000000
004E20C0 30 00 8B C0 30 31 32 33 0.140123
004E20C8 34 35 36 37 38 39 41 42 456789AB
004E20D0 4B 44 45 46 00 00 00 CDEF....

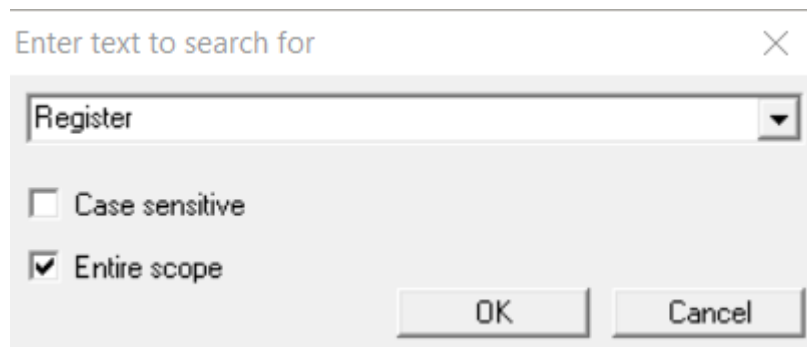
Program entry point

Paused

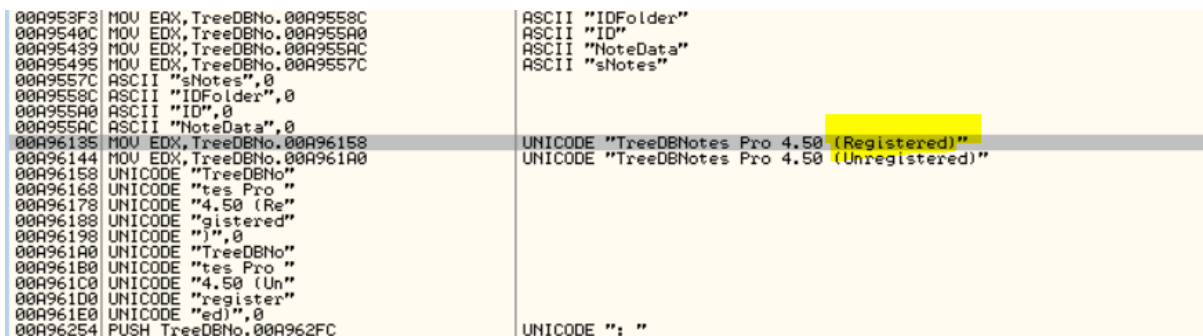
Then Right Click > Search for > All referenced text strings, which will open up a window with all identifiable text strings in the program.



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.

TreeDBNo

00A96124 8BEC PUSH ESP
00A96125 53 MOV EBP, ESP
00A96127 53 PUSH EBX
00A96128 8BD8 MOV EBX, EDI
00A96129 8B68 CMP BYTE PTR DS:[EAX+1620], 0
00A96131 74 0F JE SHORT TreeDBNo.00A96142
00A96133 8BC3 MOV EAX, EBX
00A96135 BA 50610900 MOV EDI, TreeDBNo.00A96158
00A9613A E9 E9F396FF CALL TreeDBNo.00405528
00A9613F 5B POP EBX
00A96140 5D POP EBP
00A96141 C3 RETN
00A96142 8BC3 MOV EAX, EBX
00A96144 BA 00610900 MOV EDI, TreeDBNo.00A961A0
00A96149 E9 E9F396FF CALL TreeDBNo.00405528
00A9614E 5B POP EBX
00A9614F 5D POP EBP
00A96150 C3 RETN
00A96151 00 DB 00
00A96152 00 DB 00
00A96153 00 DB 00
00A96154 42 DB 42
00A96155 00 DB 00
00A96156 00 DB 00

Unicode "TreeDBNotes Pro 4.50 (Regi
Unicode "TreeDBNotes Pro 4.50 (Unre
CHAR 'B'

EIP 00A96131 TreeDBNo.6
C 0 ES 002B 32bit 01FF
P 1 CS 002B 32bit 01FF
H 0 SS 002B 32bit 01FF
Z 1 DS 002B 32bit 01FF
S 0 FS 0053 32bit 3066
T 0 GS 002B 32bit 01FF
0 0
0 0 LastErr ERROR_SUCC
EFL 00200246 (NO, NB, E, E
ST0 empty 0.551200000000
ST1 empty 10.000000000000
ST2 empty 0.551200000000
ST3 empty 1.000000000000
ST4 empty 1.000000000000
ST5 empty 0.500000000000
ST6 empty 1.500000000000
ST7 empty 3.76587670155
FST 4220 Cond 1 0 1 0
FCW 1372 Prec NEAR, 64

Jump is taken
00A96142=TreeDBNo.00A96142

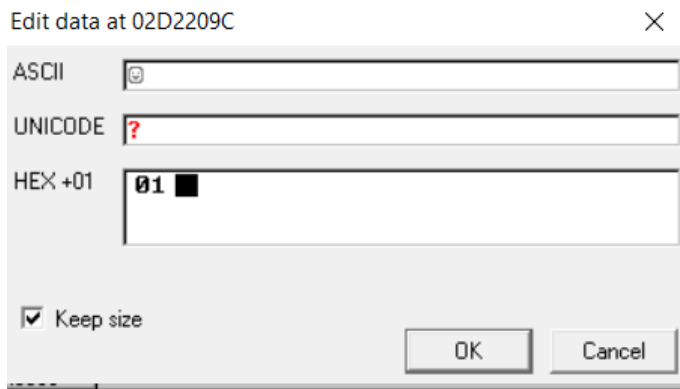
Day: 12
Week:
(Current /

Loading Status...
OK

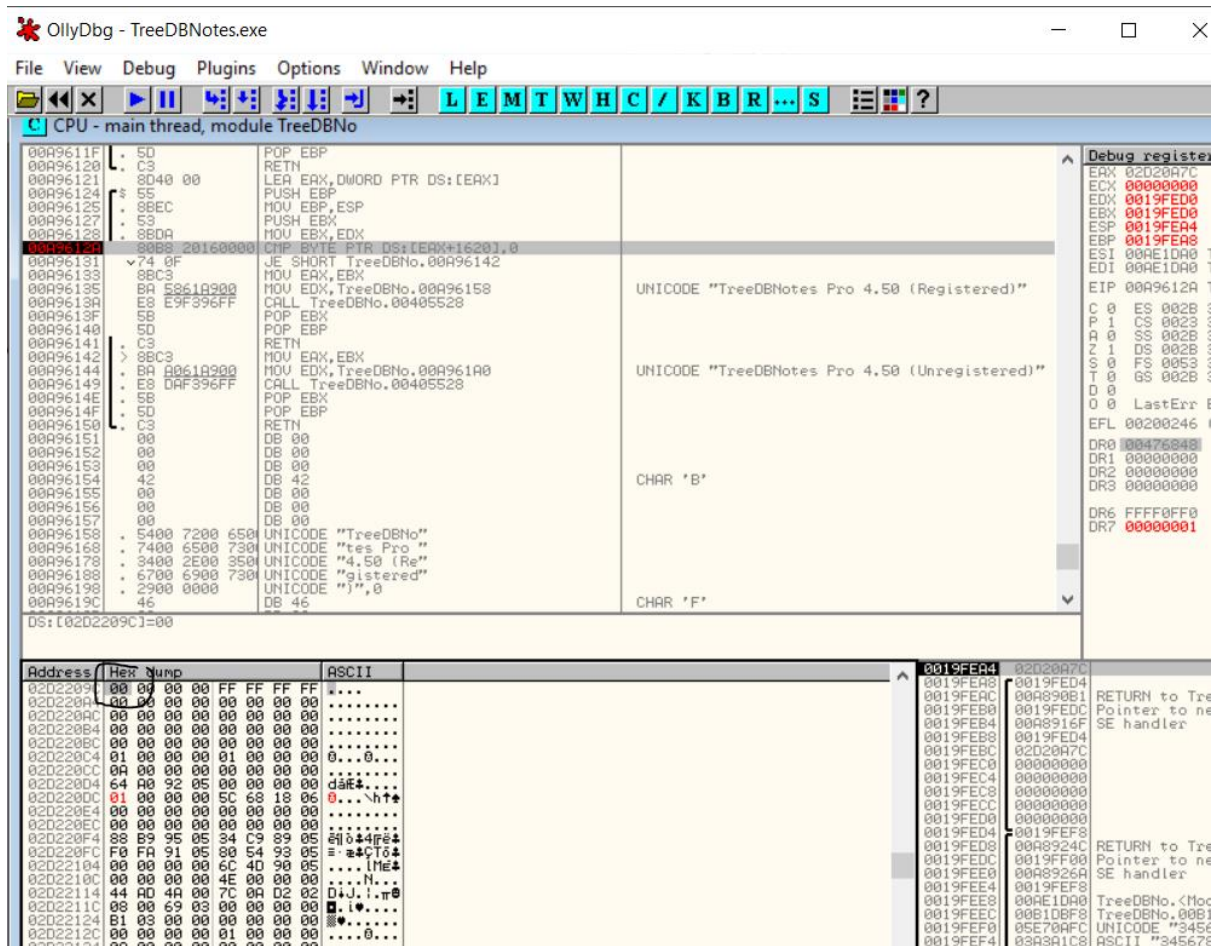
Address	Hex dump	ASCII
00A9E2000	00 00 00 00 00 00 00 00
00A9E2008	90 50 2B 32 02 80 40 00	0P+2010
00A9E2010	90 51 41 00 30 A4 41 00	0DA, 0nA,
00A9E2018	04 54 41 00 0C 87 41 00	0TA, 0nA,
00A9E2020	34 8E 41 00 72 13 8B C0	4AA, rfil
00A9E2028	02 00 0B C0 00 8D 40 00	0, i, i0,
00A9E2030	00 8D 40 00 00 8D 40 00	..i0, i0,
00A9E2038	01 8D 40 00 DC 20 AE 00	010, ..

0019FD88 06120064 ASCII "cU"
0019FDC8 0019FDC8
0019FD90 00A87544 RETURN to TreeDBNo.00A
0019FD94 0019FDD0 Pointer to next SEH re
0019FD98 00A876E2 SE handler
0019FDC8 0019FDC8
0019FDA0 00A81D00 TreeDBNo.<ModuleEntryF
0019FDA8 00A81D00 TreeDBNo.<ModuleEntryF
0019FDB0 02C20A7C
0019FDB8 00A8A000

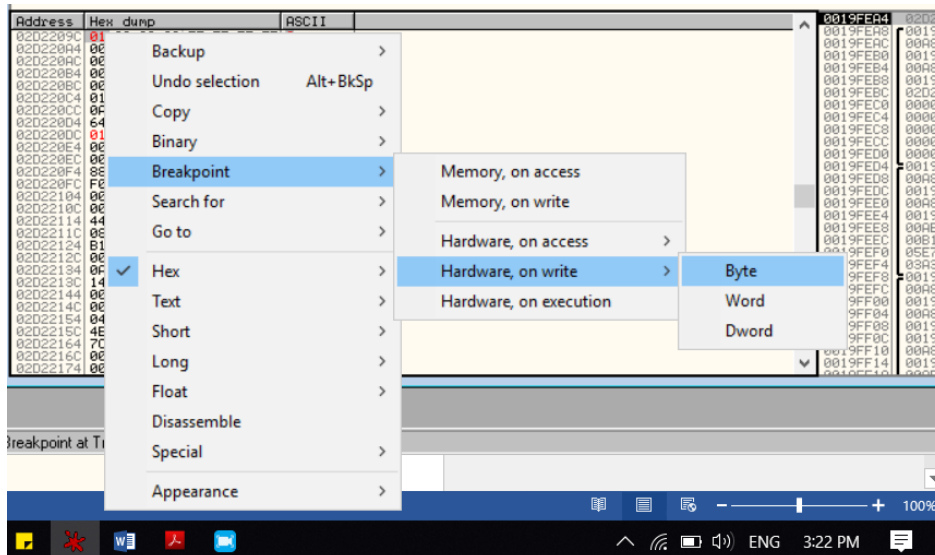
We can notice that a compare right above the jump line. That should be where the program compares the entered registration code to the right code to determine if the user is registered or not. The variable it compares against is a global variable which is hinted from “DS” and is in the memory address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



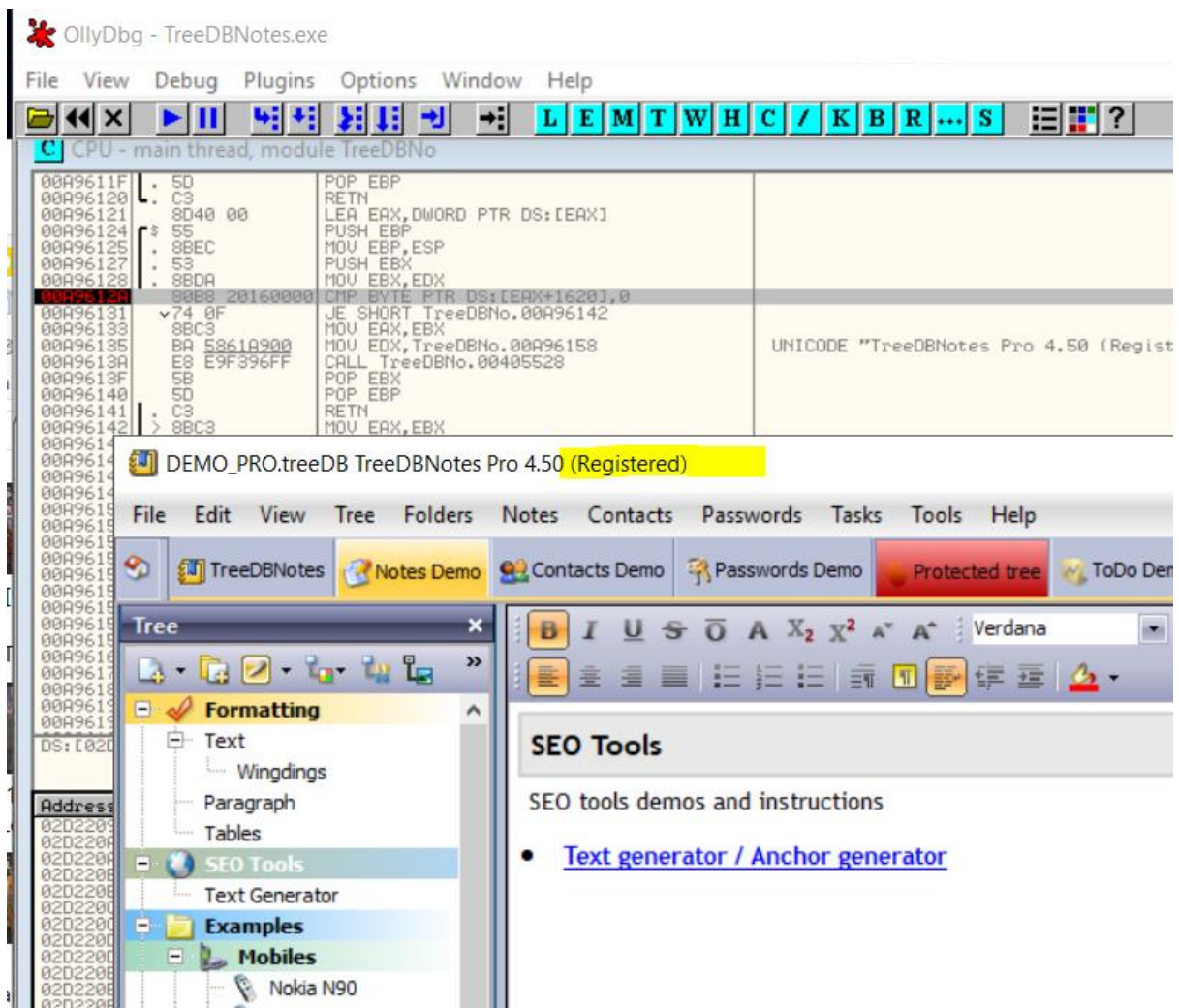
Then we can run the program and check what happens. Unfortunately, the 01 gets replaced again with 00 which means that the program has another instance where it checks if the user is registered. And if not, it resets the global variable to 00 again.



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 hardware breakpoint worked as it should be.

00A99240	6C	DB 6C	CHAR 'l'
00A99241	00	DB 00	
00A99242	00	DB 00	
00A99243	00	DB 00	
00A99244	3A90 20160000	CMP DL, BYTE PTR DS:[EAX+1620]	
00A9924A	74 06	JE SHORT TreeDBNo.00A99252	
00A9924C	8890 20160000	MOV BYTE PTR DS:[EAX+1620], DL	
00A99252	C3	RETN	
00A99253	90	NOP	
00A99254	55	PUSH EBP	
00A99255	8BEC	MOV EBP, ESP	
00A99257	83C4 F8	ADD ESP, -8	
00A9925A	53	PUSH EBX	
00A9925B	56	PUSH ESI	
00A9925C	57	PUSH EDI	
00A9925D	8940 F8	MOV DWORD PTR SS:[EBP-8], ECX	
00A99260	8955 FC	MOV DWORD PTR SS:[EBP-4], EDX	
00A99263	8BD8	MOV EBX, EAX	
00A99265	8B45 FC	MOV EAX, DWORD PTR SS:[EBP-4]	
00A99268	8B15 7C125200	MOV EDX, DWORD PTR DS:[52127C]	TreeDBNo.00521
00A9926E	E8 65AE96FF	CALL TreeDBNo.004040D8	
Return to 00A891CE (TreeDBNo.00A891CE)			
Jump from 00A9924A			
Address	Hex dump	ASCII	
02CC209C	00 00 00 00 FF FF FF FF	
02CC20A4	00 00 00 00 00 00 00 00	
02CC20AC	00 00 00 00 00 00 00 00	
02CC20B4	00 00 00 00 00 00 00 00	
02CC20BC	00 00 00 00 00 00 00 00	
02CC20C4	01 00 00 00 01 00 00 00	0...0...	
02CC20CC	0A 00 00 00 00 00 00 00	
02CC20D4	64 A0 08 05 00 00 00 00	d4A00805	
02CC20DC	01 00 00 00 5C 68 79 05	0... \hy	
02CC20E4	00 00 00 00 00 00 00 00	
02CC20EC	00 00 00 00 00 00 00 00	
02CC20F4	88 B9 0B 05 34 C9 FF 04	88B90B0534C9FF04	
02CC20FC	F0 FA 07 05 80 54 09 05	F0FA070580540905	
02CC2104	00 00 00 00 6C 4D 06 05lM	
02CC210C	00 00 00 00 4E 00 00 00N...	
02CC2114	44 AD 4A 00 7C 0A CC 02	D4AD4A007C0ACC02	

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.

00A99244	3A90 20160000	CMP DL, BYTE PTR DS:[EAX+1620]	
00A9924A	74 06	JE SHORT TreeDBNo.00A99252	
00A9924C	8890 20160000	MOV BYTE PTR DS:[EAX+1620], DL	
00A99252	C3	RETN	
00A99253	90	NOP	

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.

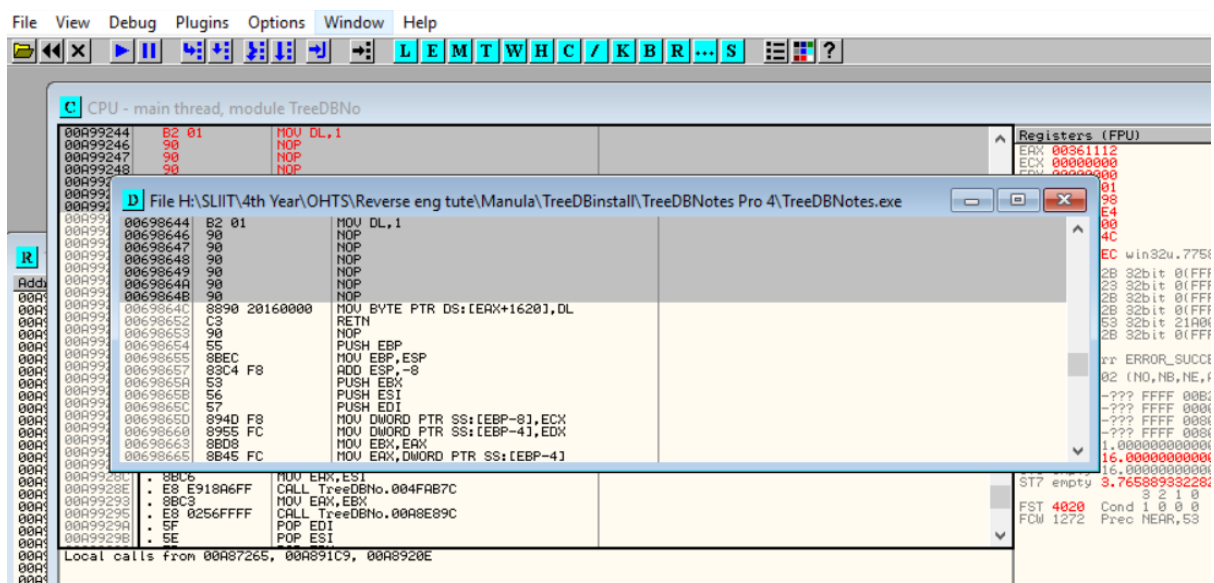
Double click on the first NOP and assemble it with “MOV DL,1” so it will add a 1 when it’s called. Now it will look like in the bellow image.

00A9923F	00	DB 00
00A99240	6C	DB 6C
00A99241	00	DB 00
00A99242	00	DB 00
00A99243	00	DB 00
00A99244	B2 01	MOV DL,1
00A99246	90	NOP
00A99247	90	NOP
00A99248	90	NOP
00A99249	90	NOP
00A9924A	90	NOP
00A9924B	90	NOP
00A9924C	8890 20160000	MOV BYTE PTR DS:[EAX+1620],DL
00A99252	C3	RETN
00A99253	90	NOP
00A99254	55	PUSH EBP
00A99255	8BEC	MOV EBP,ESP
00A99257	83C4 F8	ADD ESP,-8
00A9925A	53	PUSH EBX
00A9925B	56	PUSH ESI
00A9925C	57	PUSH EDI

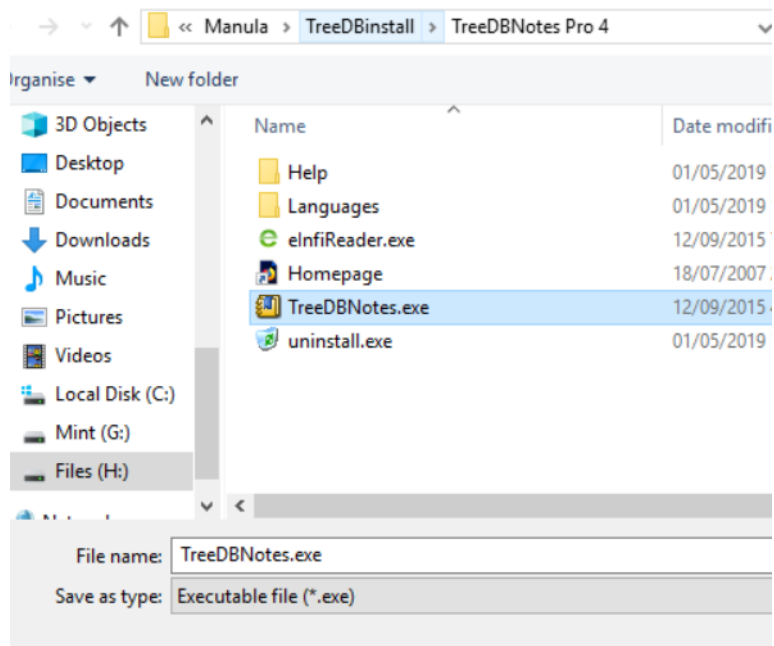
Now all we have to do is to play the application so it will save the 01 as the global variable which in return makes it think that the program is registered.

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!!

