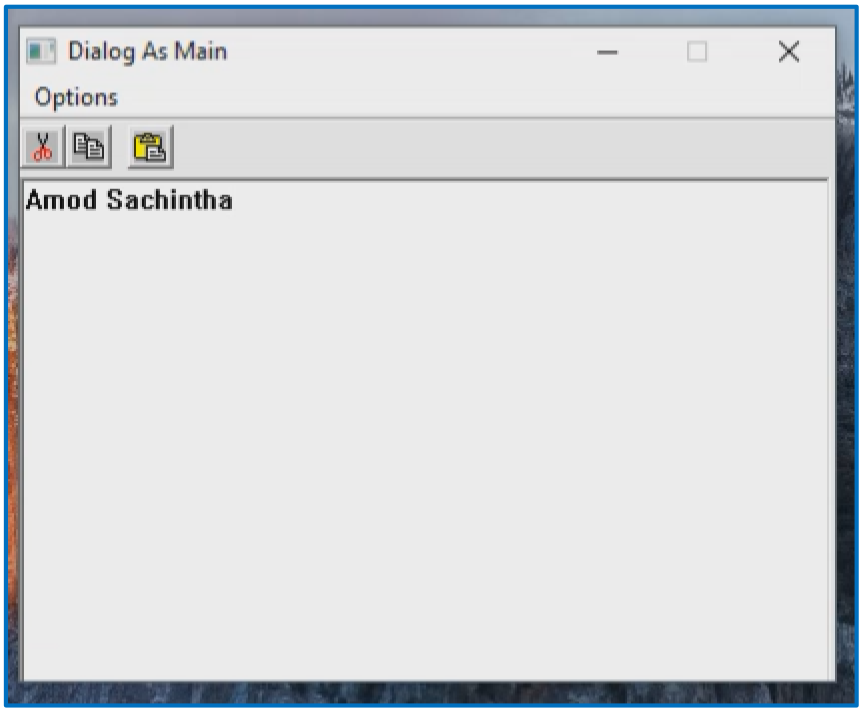
# Reversing with OllyDbg

OFFENSIVE HACKING AND TACTICAL STRATEGY

KVA Sachintha | **IT-16158528** | Reverse Engineering

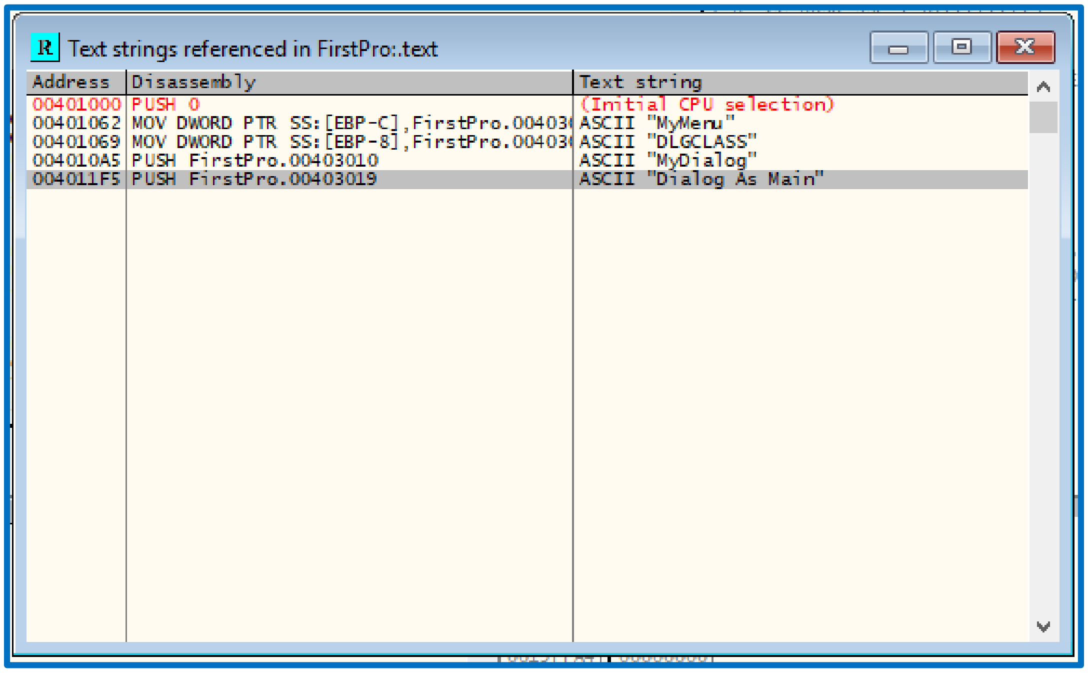
May 1, 2019

## Tutorial 1 – Modifying Popup Title (VIDEO)



Amod Sachintha

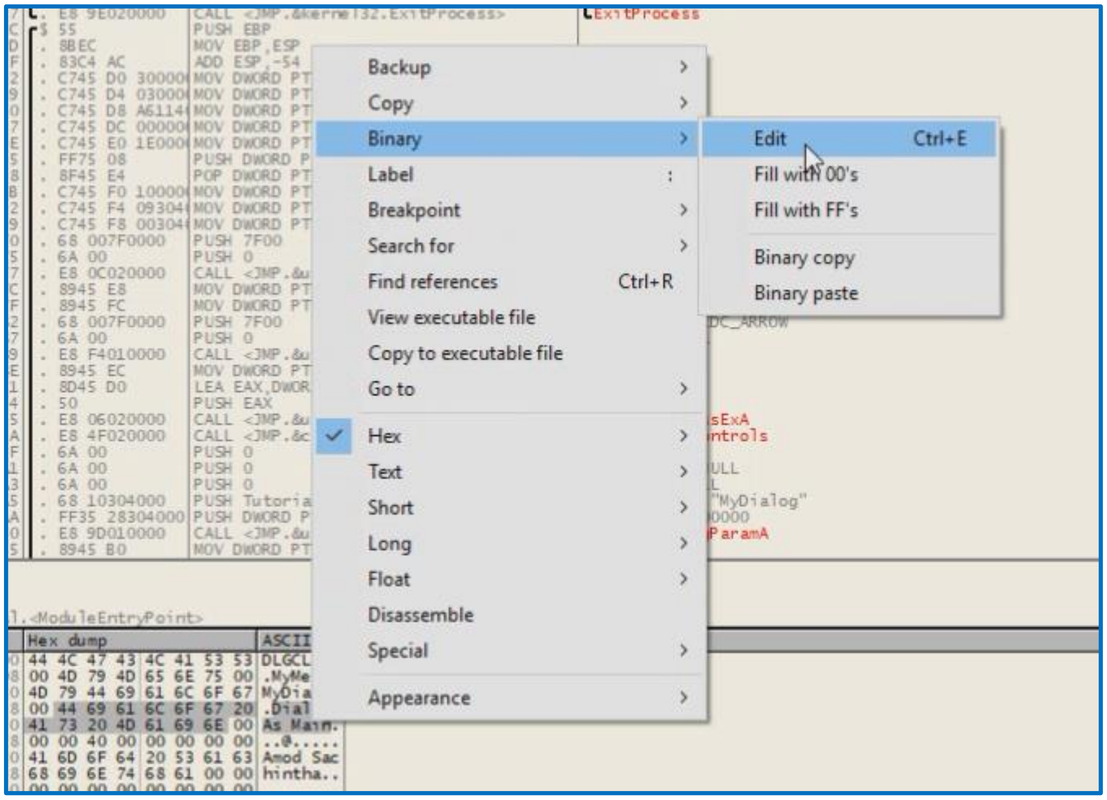
I've used OllyDbg to rename the popup dialog box's name. This can be simply achieved by searching for strings used within the program.



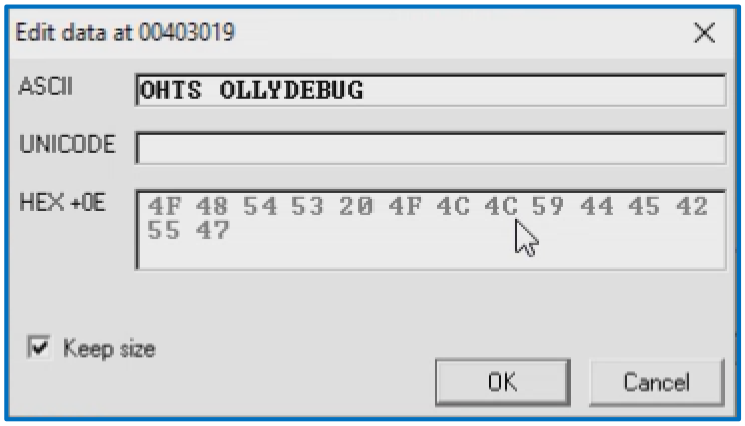
```
Address   Disassembly                               Text string
00401000  PUSH 0                                    (Initial CPU selection)
00401062  MOV DWORD PTR SS:[EBP-C],FirstPro.00403(  ASCII "MyMenu"
00401069  MOV DWORD PTR SS:[EBP-8],FirstPro.00403(  ASCII "DLGCLASS"
004010A5  PUSH FirstPro.00403010                    ASCII "MyDialog"
004011F5  PUSH FirstPro.00403019                    ASCII "Dialog As Main"
```

The search for referenced strings were done by right-clicking on the debugger window with the module **FirstPro** loaded. The module "**FirstPro**" is the application we're trying to modify.

After the search was done, the referenced text "Dialog as Main" was found to be the Title of the popup dialog.


```
 E8 9E020000      CALL <JMP.&kernel32.ExitProcess>      ExitProcess
$ 55              PUSH EBP
. 8BEC            MOV EBP,ESP
. 83C4 AC         ADD ESP,-54
. C745 D0 30000(  MOV DWORD PT
. C745 D4 03000(  MOV DWORD PT
. C745 D8 A6114(  MOV DWORD PT
. C745 DC 00000(  MOV DWORD PT
. C745 E0 1E000(  MOV DWORD PT
. FF75 08         PUSH DWORD P
. 8F45 E4         POP DWORD PT
. C745 F0 10000(  MOV DWORD PT
. C745 F4 09304(  MOV DWORD PT
. C745 F8 00304(  MOV DWORD PT
. 68 007F0000     PUSH 7F00
. 6A 00           PUSH 0
. E8 0C020000     CALL <JMP.&u
. 8945 E8         MOV DWORD PT
. 8945 FC         MOV DWORD PT
. 68 007F0000     PUSH 7F00
. 6A 00           PUSH 0
. E8 F4010000     CALL <JMP.&u
. 8945 EC         MOV DWORD PT
. 8D45 D0         LEA EAX,DWOR
. 50              PUSH EAX
. E8 06020000     CALL <JMP.&u
. E8 4F020000     CALL <JMP.&c
. 6A 00           PUSH 0
. 6A 00           PUSH 0
. 6A 00           PUSH 0
. 68 10304000     PUSH Tutoria                          "MyDialog"
. FF35 28304000   PUSH DWORD P
. E8 9D010000     CALL <JMP.&u                          ParamA
. 8945 B0         MOV DWORD PT

1.<ModuleEntryPoint>

Hex dump                  ASCII
44 4C 47 43 4C 41 53 53   DLGCL
00 4D 79 4D 65 6E 75 00   .MyMe
4D 79 44 69 61 6C 6F 67   MyDia
00 44 69 61 6C 6F 67 20   .Dial
41 73 20 4D 61 69 6E 00   As Main.
00 00 40 00 00 00 00 00   ..@.....
41 6D 6F 64 20 53 61 63   Amod Sac
68 69 6E 74 68 61 00 00   hintha..
```

The selected text was then edited.


Edit data at 00403019

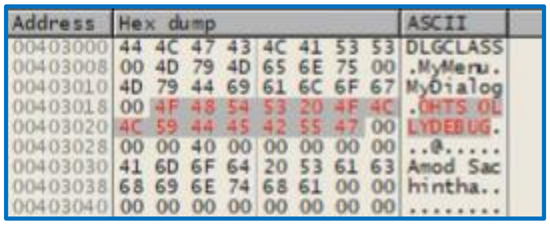
ASCII: OHTS OLLYDEBUG

UNICODE:

HEX +0E: 4F 48 54 53 20 4F 4C 4C 59 44 45 42 55 47

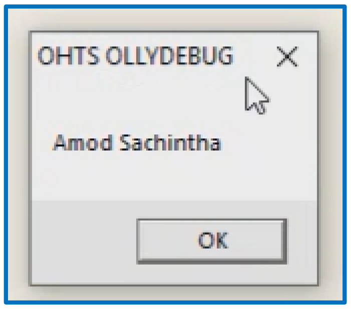
[✓] Keep size

The red colored text in the ASCII strings shows that it was recently modified.



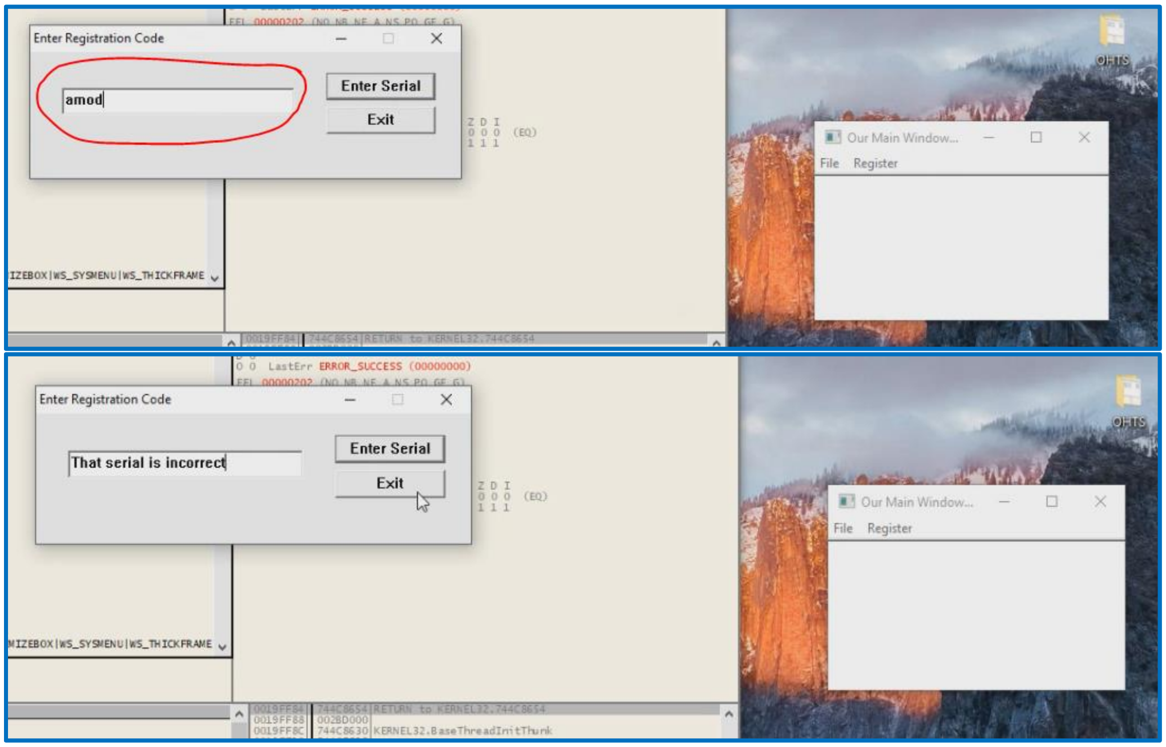
```
Address   Hex dump                 ASCII
00403000  44 4C 47 43 4C 41 53 53  DLGCLASS
00403008  00 4D 79 4D 65 6E 75 00  .MyMenu.
00403010  4D 79 44 69 61 6C 6F 67  MyDialog
00403018  00 4F 48 54 53 20 4F 4C  .OHTS OL
00403020  4C 59 44 45 42 55 47 00  LYDEBUG.
00403028  00 00 40 00 00 00 00 00  ..@.....
00403030  41 6D 6F 64 20 53 61 63  Amod Sac
00403038  68 69 6E 74 68 61 00 00  hintha..
00403040  00 00 00 00 00 00 00 00  ........
```

After this modification was done, the program was run. The popup dialog has the modified text!



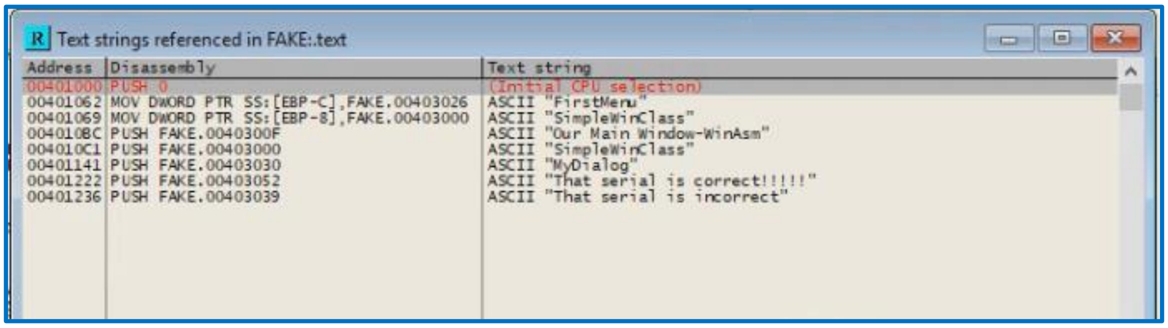
```
OHTS OLLYDEBUG

Amod Sachintha
```

## Tutorial 2 – Bypassing Serial (VIDEO)



This executable asks for a serial in the registration dialog which I've used OllyDbg to circumvent. The process is as follows.

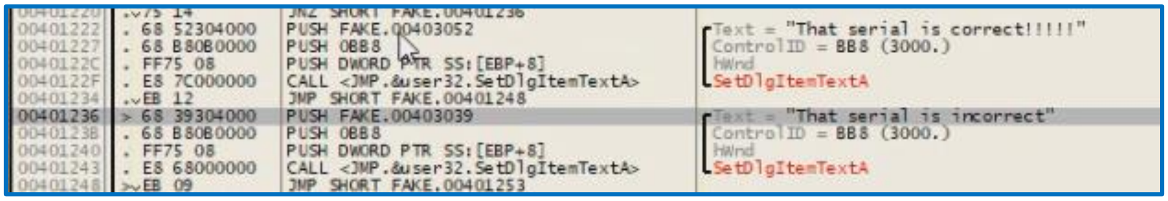
Firstly, running the program with a wrong serial gave me a string namely "That serial is incorrect" which I was able to search and find within the application module.



Text strings referenced in FAKE:.text

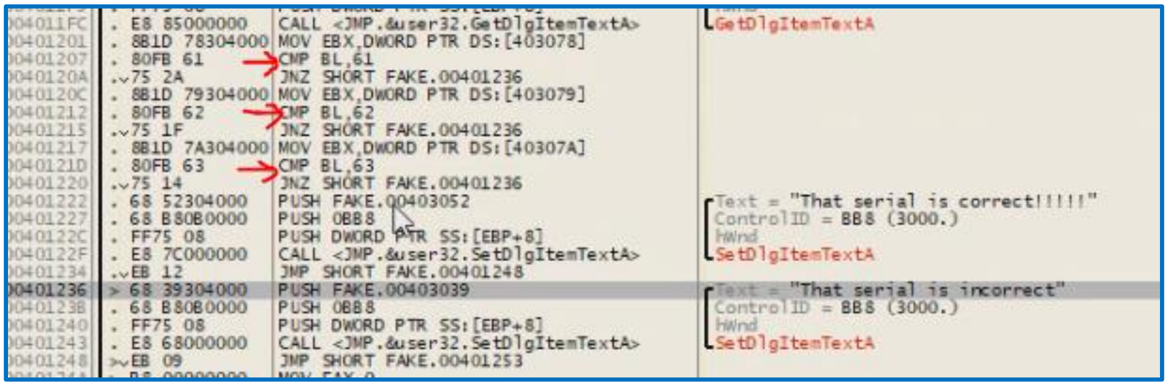
```
Address  Disassembly                              Text string
00401000 PUSH 0                                   (Initial CPU selection)
00401062 MOV DWORD PTR SS:[EBP-C],FAKE.00403026   ASCII "FirstMenu"
00401069 MOV DWORD PTR SS:[EBP-8],FAKE.00403000   ASCII "SimpleWinClass"
004010BC PUSH FAKE.0040300F                       ASCII "Our Main Window-WinAsm"
004010C1 PUSH FAKE.00403000                       ASCII "SimpleWinClass"
00401141 PUSH FAKE.00403030                       ASCII "MyDialog"
00401222 PUSH FAKE.00403052                       ASCII "That serial is correct!!!!!"
00401236 PUSH FAKE.00403039                       ASCII "That serial is incorrect"
```

Double clicking on the relevant string moved me to the corresponding address on the dissembler code.



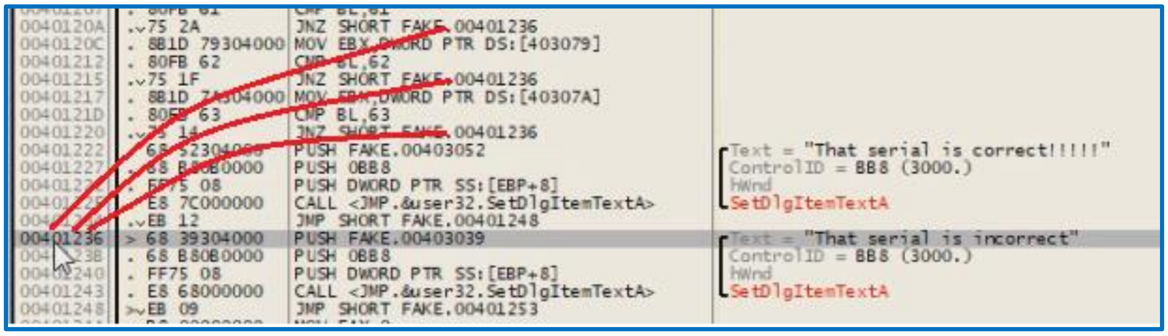
```
00401220  .v75 14         JNZ SHORT FAKE.00401236
00401222  . 68 52304000   PUSH FAKE.00403052                      ┌Text = "That serial is correct!!!!!"
00401227  . 68 B80B0000   PUSH 0BB8                               │ControlID = BB8 (3000.)
0040122C  . FF75 08       PUSH DWORD PTR SS:[EBP+8]               │hWnd
0040122F  . E8 7C000000   CALL <JMP.&user32.SetDlgItemTextA>      └SetDlgItemTextA
00401234  .vEB 12         JMP SHORT FAKE.00401248
00401236  > 68 39304000   PUSH FAKE.00403039                      ┌Text = "That serial is incorrect"
0040123B  . 68 B80B0000   PUSH 0BB8                               │ControlID = BB8 (3000.)
00401240  . FF75 08       PUSH DWORD PTR SS:[EBP+8]               │hWnd
00401243  . E8 68000000   CALL <JMP.&user32.SetDlgItemTextA>      └SetDlgItemTextA
00401248  >vEB 09         JMP SHORT FAKE.00401253
```

Since the serial was being matched to validate it, I looked for Comparisons (CMP) on the Assembly code. This revealed multiple CMP operators within the vicinity of the strings that were searched.



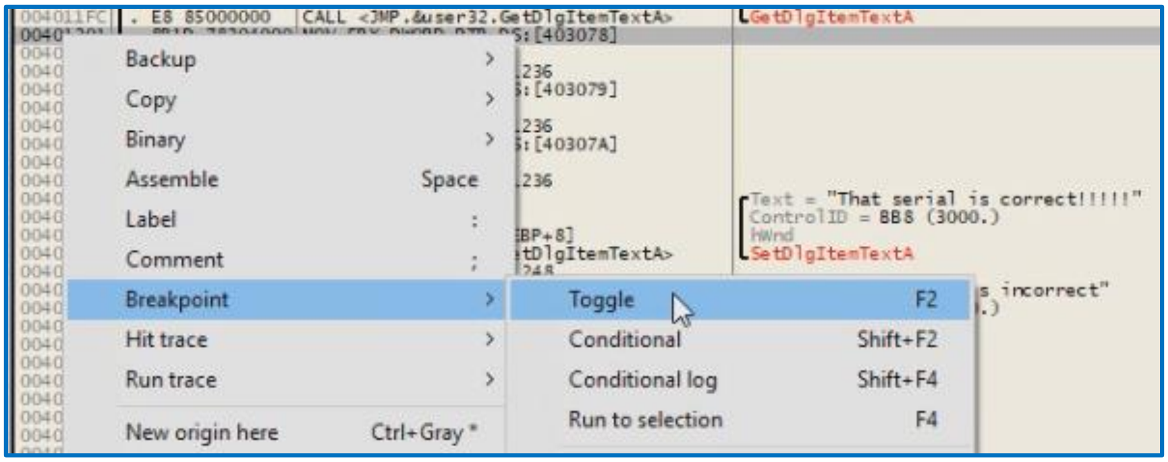
```
004011FC  . E8 85000000   CALL <JMP.&user32.GetDlgItemTextA>      └GetDlgItemTextA
00401201  . 8B1D 78304000 MOV EBX,DWORD PTR DS:[403078]
00401207  . 80FB 61       CMP BL,61
0040120A  .v75 2A         JNZ SHORT FAKE.00401236
0040120C  . 8B1D 79304000 MOV EBX,DWORD PTR DS:[403079]
00401212  . 80FB 62       CMP BL,62
00401215  .v75 1F         JNZ SHORT FAKE.00401236
00401217  . 8B1D 7A304000 MOV EBX,DWORD PTR DS:[40307A]
0040121D  . 80FB 63       CMP BL,63
00401220  .v75 14         JNZ SHORT FAKE.00401236
00401222  . 68 52304000   PUSH FAKE.00403052                      ┌Text = "That serial is correct!!!!!"
00401227  . 68 B80B0000   PUSH 0BB8                               │ControlID = BB8 (3000.)
0040122C  . FF75 08       PUSH DWORD PTR SS:[EBP+8]               │hWnd
0040122F  . E8 7C000000   CALL <JMP.&user32.SetDlgItemTextA>      └SetDlgItemTextA
00401234  .vEB 12         JMP SHORT FAKE.00401248
00401236  > 68 39304000   PUSH FAKE.00403039                      ┌Text = "That serial is incorrect"
0040123B  . 68 B80B0000   PUSH 0BB8                               │ControlID = BB8 (3000.)
00401240  . FF75 08       PUSH DWORD PTR SS:[EBP+8]               │hWnd
00401243  . E8 68000000   CALL <JMP.&user32.SetDlgItemTextA>      └SetDlgItemTextA
00401248  >vEB 09         JMP SHORT FAKE.00401253
```

Looking closely into the Jump (JNZ) call, the address it jumps to is **`0x00401236`**, which corresponds to the block of code that says “That serial is incorrect”.



```
0040120A  .v75 2A         JNZ SHORT FAKE.00401236
0040120C  . 8B1D 79304000 MOV EBX,DWORD PTR DS:[403079]
00401212  . 80FB 62       CMP BL,62
00401215  .v75 1F         JNZ SHORT FAKE.00401236
00401217  . 8B1D 7A304000 MOV EBX,DWORD PTR DS:[40307A]
0040121D  . 80FB 63       CMP BL,63
00401220  .v75 14         JNZ SHORT FAKE.00401236
00401222  . 68 52304000   PUSH FAKE.00403052                      ┌Text = "That serial is correct!!!!!"
00401227  . 68 B80B0000   PUSH 0BB8                               │ControlID = BB8 (3000.)
0040122C  . FF75 08       PUSH DWORD PTR SS:[EBP+8]               │hWnd
0040122F  . E8 7C000000   CALL <JMP.&user32.SetDlgItemTextA>      └SetDlgItemTextA
00401234  .vEB 12         JMP SHORT FAKE.00401248
00401236  > 68 39304000   PUSH FAKE.00403039                      ┌Text = "That serial is incorrect"
0040123B  . 68 B80B0000   PUSH 0BB8                               │ControlID = BB8 (3000.)
00401240  . FF75 08       PUSH DWORD PTR SS:[EBP+8]               │hWnd
00401243  . E8 68000000   CALL <JMP.&user32.SetDlgItemTextA>      └SetDlgItemTextA
00401248  >vEB 09         JMP SHORT FAKE.00401253
```

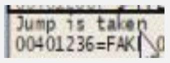
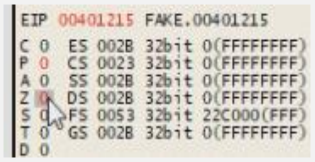
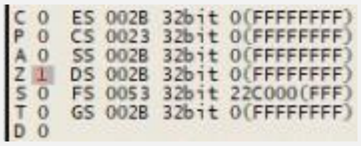
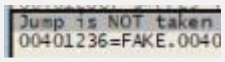
Since after each comparison there's a Jump (JNZ) instruction to the "incorrect block" of code, I toggled a breakpoint well before the comparisons happen. The motive for this was to manually step into each instruction and block all Jumps (JNZ) to the "incorrect block" of code.



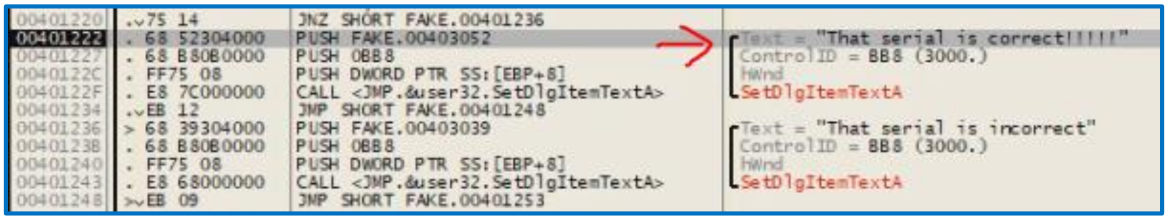
```
004011FC  . E8 85000000   CALL <JMP.&user32.GetDlgItemTextA>   LGetDlgItemTextA

                                                              [Text = "That serial is correct!!!!!"
                                                              |ControlID = BB8 (3000.)
                                                              |hWnd
                                                              LSetDlgItemTextA
```

After the breakpoint was set, the program was executed. After entering some arbitrary string in the registration dialog, the breakpoint was triggered and execution was paused.

From that step onward, I manually stepped over instructions. Once a JNZ instruction was reached, I manually averted the Jump into the call by altering the "Z" register in the application preventing the Jump.

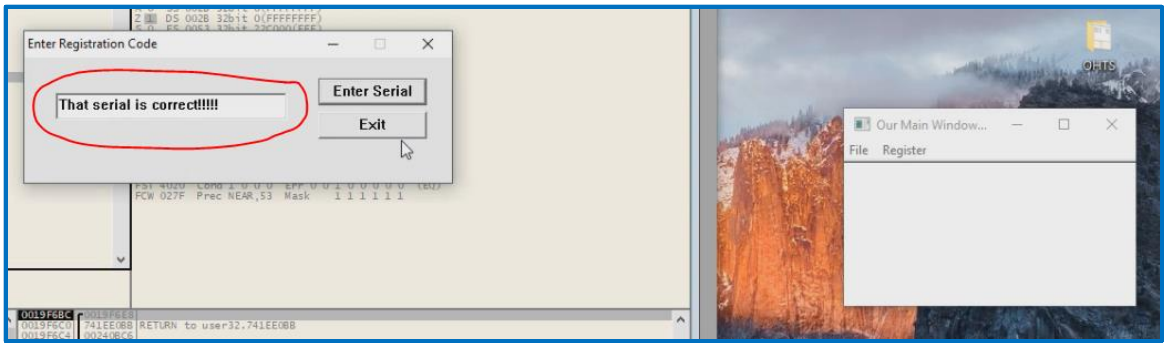
| Jump Taken | "Z" register is set to jump | Toggle Z register to "1", this prevents the jump | Jump is prevented |
|---|---|---|---|
|  `Jump is taken`<br>`00401236=FAK` |  `EIP 00401215 FAKE.00401215`<br>`C 0  ES 002B 32bit 0(FFFFFFFF)`<br>`P 0  CS 0023 32bit 0(FFFFFFFF)`<br>`A 0  SS 002B 32bit 0(FFFFFFFF)`<br>`Z 0  DS 002B 32bit 0(FFFFFFFF)`<br>`S 0  FS 0053 32bit 22C000(FFF)`<br>`T 0  GS 002B 32bit 0(FFFFFFFF)`<br>`D 0` |  `C 0  ES 002B 32bit 0(FFFFFFFF)`<br>`P 0  CS 0023 32bit 0(FFFFFFFF)`<br>`A 0  SS 002B 32bit 0(FFFFFFFF)`<br>`Z 1  DS 002B 32bit 0(FFFFFFFF)`<br>`S 0  FS 0053 32bit 22C000(FFF)`<br>`T 0  GS 002B 32bit 0(FFFFFFFF)`<br>`D 0` |  `Jump is NOT taken`<br>`00401236=FAKE.0040` |

After repeating this process until all Jumps to the incorrect block was prevented, the application was set to run normally.



```
00401220  .v75 14         JNZ SHORT FAKE.00401236
00401222  . 68 52304000   PUSH FAKE.00403052                    [Text = "That serial is correct!!!!!"
00401227  . 68 B80B0000   PUSH 0BB8                             |ControlID = BB8 (3000.)
0040122C  . FF75 08       PUSH DWORD PTR SS:[EBP+8]             |hWnd
0040122F  . E8 7C000000   CALL <JMP.&user32.SetDlgItemTextA>    LSetDlgItemTextA
00401234  .vEB 12         JMP SHORT FAKE.00401248
00401236  > 68 39304000   PUSH FAKE.00403039                    [Text = "That serial is incorrect"
0040123B  . 68 B80B0000   PUSH 0BB8                             |ControlID = BB8 (3000.)
00401240  . FF75 08       PUSH DWORD PTR SS:[EBP+8]             |hWnd
00401243  . E8 68000000   CALL <JMP.&user32.SetDlgItemTextA>    LSetDlgItemTextA
00401248  >vEB 09         JMP SHORT FAKE.00401253
```

It then showed that the serial was correct. The Application is now CRACKED and bypassed!


Enter Registration Code
That serial is correct!!!!!
Enter Serial
Exit
Our Main Window...
File
Register