Bypassing a Server check by XOR06

Intro

Welcome again, nice to see you back! In this tutorial we will be cracking a registration from a 'real' program which uses the registry, and an abnormal server check as protection. Before I come to the server check, I'll show you how to patch the serial checking routine.

I assume that you know Olly and how to change flags etc- just basic knowledge. I have a lot of people helping with this one so I would like to thanks my Friends in TPoDT (especially DMichael, one of my Mentors), and Nwokiller for the great support. Do I have to say; "*This is only for learning so please: if you do like it, pay for it!*"?

Remember the best way to learn is to try to Patch the program yourself first, if you don't succeed (or if you would like to see another approach) then read this tutorial. Have Fun!

Toolz

Olly

Crackers Tools (if you like)

Target

Internet Download Manager 6.11

You can find the target at http://www.internetdownloadmanager.com/download.html

Index

Intro 1

Toolz 1

Target 1

Investigate The Target 2

Finding the Serial check Patch 5

Applying the Serial patch 10

Going Online 11

Finding the First Server Check 11

Finding the Second Server Check 13

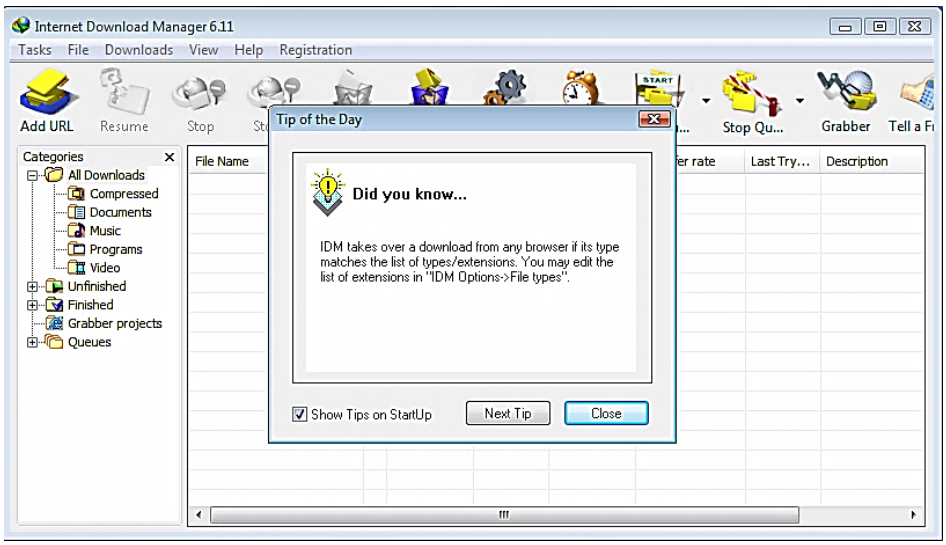
The Final Patch 16

Final Notes 16

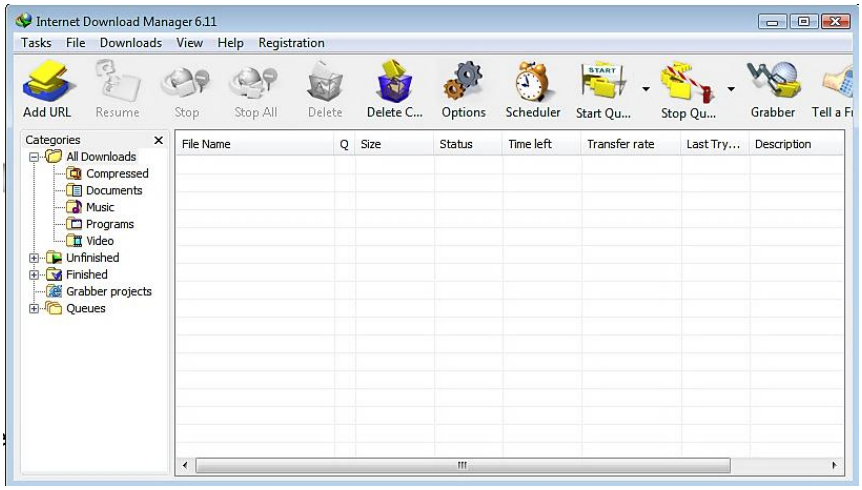
Thanks to: 16

Investigate the Target

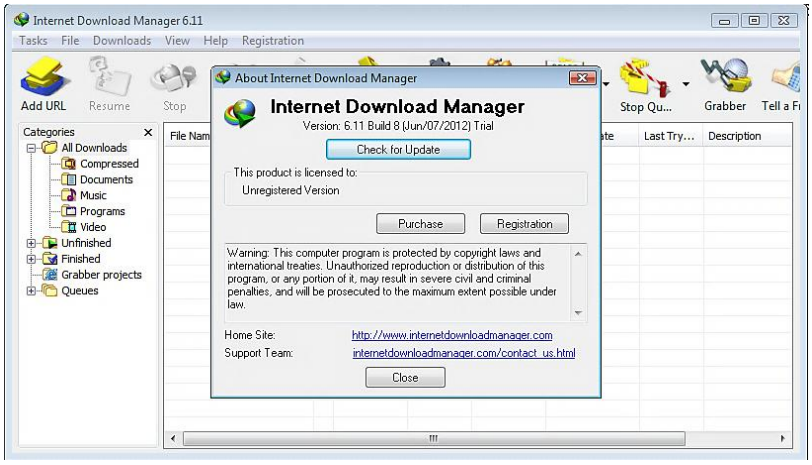
As this is an Internet download program start by disconnecting your internet. (This is always a good idea when starting cracking a target). Okay, then let's run the target. First time you start the program you will see this:



As you can see, no nag, it just starts by showing "Tip of the day". Nice. Now uncheck the option "show Tips on StartUp" so we don't have to deal with this every time we start. Close the dialog "tip of the day" and now you are at the main dialog.

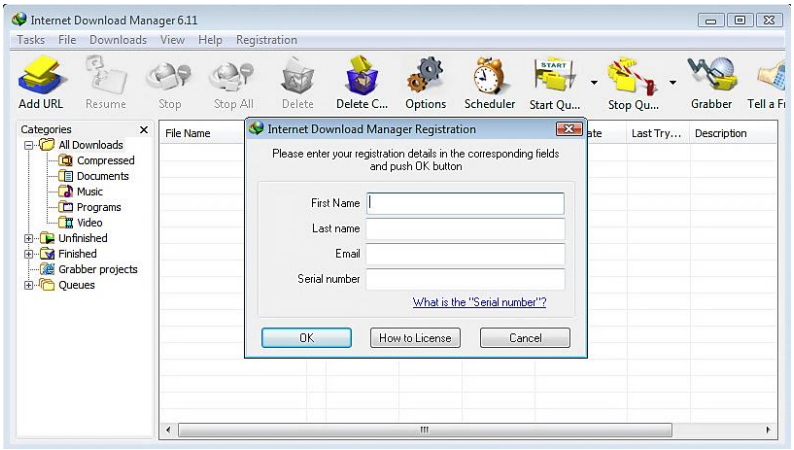


I think it's obvious, that we need to go to the menu "Registration", but before we do that let's take a look in the menu "Help". Choose the "About IDM" dialog. Now you see this:

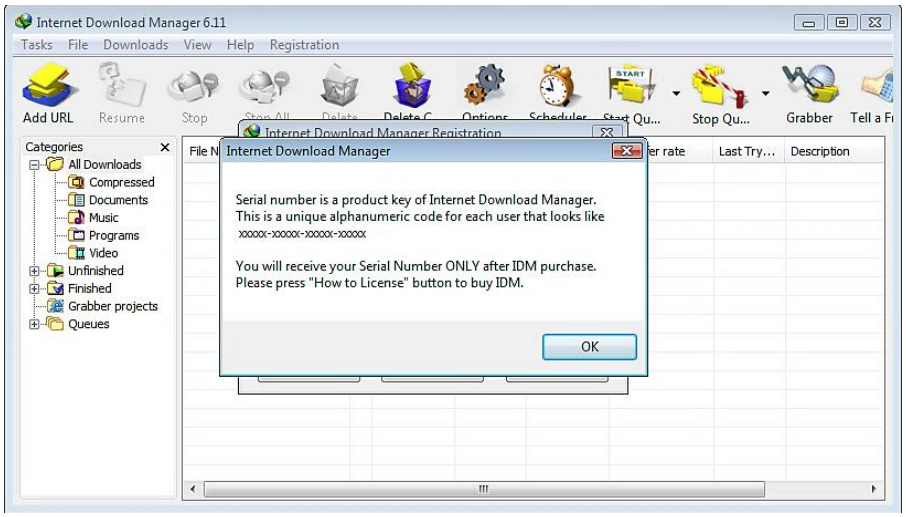


Ok- “Unregistered Version” and “Trial”; I think we can assume that they would be changed, if we’re registered right. (You could start your point of attack there, but I’ll always attack from the Registration dialog.)

Close the About dialog and choose the “Registration” menu and click “Registration”

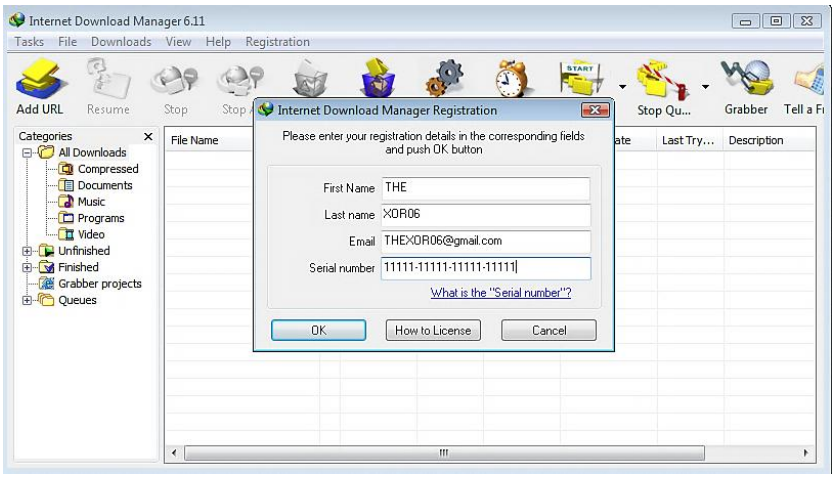


Before you start to enter your info, try to click on “what is the “Serial number” maybe it shows something exciting. Now this pops up:

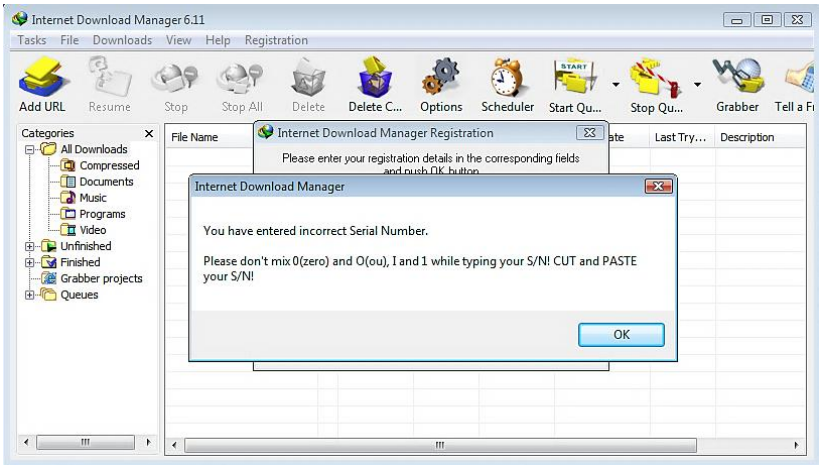


Well Well, look at that- it show us the format of the Serial! Without debugging we now know that the serial looks like this “XXXXX-XXXXX-XXXXX-XXXXX”

Return to the Registration dialog, enter your info and a dummy Serial in the format you just found



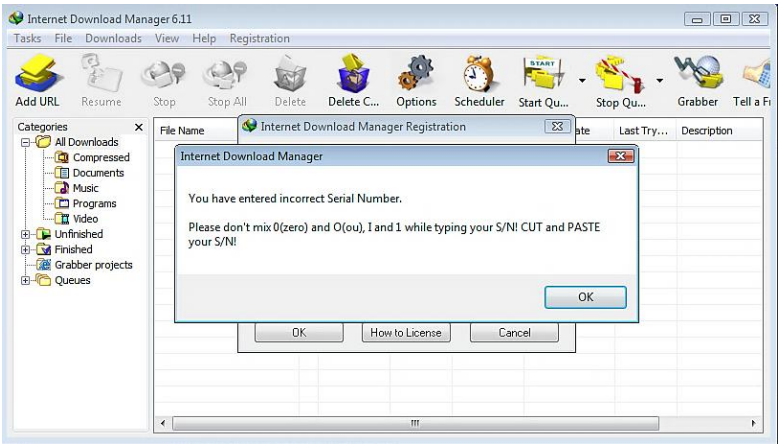
Hit the ok button and, yeah you guessed it, the bad boy pops up.



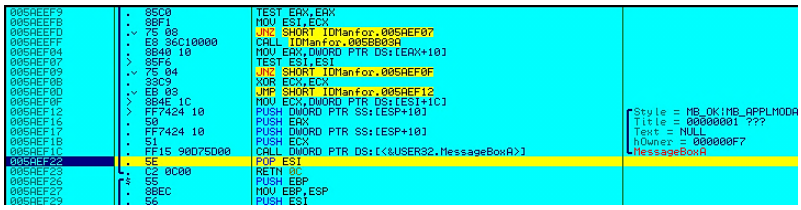
When you press the “Ok” button, you will return and the info is still in the box. Okay, that is all we need to know now, let’s find the patch. Close the target and check that the icon is gone in the system tray.

Finding the Serial check Patch

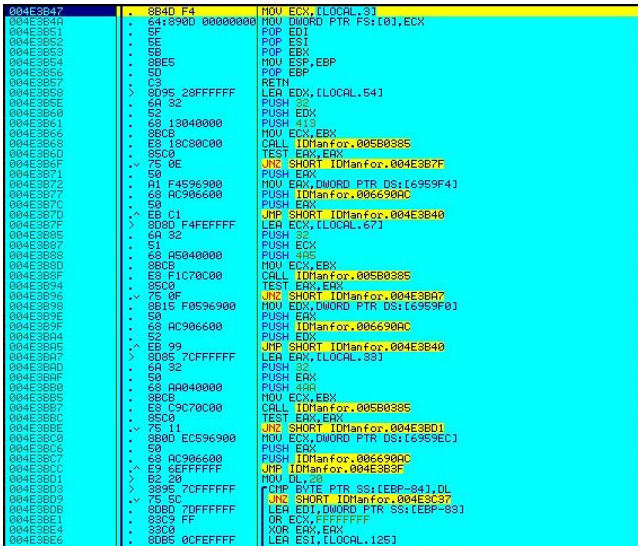
Fire up Olly and load the target. If you try to use the old search-for-the-string trick, you will see that it won't work on this target- it just gets you to a call table. So let's us do something else. Run the target in Olly, go to the "Registration" dialog and enter the Dummy info again (remember the correct format on the serial) and press "Ok". You should now see this again:



Okay now there are many ways to go from here but the easiest to explain is this; press "Pause" (or F12) and chose "Execute till user-code" (or ALT+F9). Now press the "Ok" button and BAM, Olly breaks here:



As you can see, this is right after the MessageboxA, which just called our bad boy. So now step (F8) down and over the RETN 0C command, so we can return to where this is called from. Then we land here:



In a large routine, could this have something to do with the "check-serial-routine"?. Well the bad boy is called from here, so let's scroll up to the top of the routine and place a BP here at 4E39F0:

```
004E39DE    90                NOP
004E39DF    90                NOP
004E39E0  . B8 18CD5F00       MOV EAX,IDManfor.005FCD18
004E39E5  . C3                RETN
004E39E6    90                NOP
004E39E7    90                NOP
004E39E8    90                NOP
004E39E9    90                NOP
004E39EA    90                NOP
004E39EB    90                NOP
004E39EC    90                NOP
004E39ED    90                NOP
004E39EE    90                NOP
004E39EF    90                NOP
004E39F0  r. 55               PUSH EBP
004E39F1  . 8BEC              MOV EBP,ESP
004E39F3  . 6A FF             PUSH -1
004E39F5  . 68 153C5D00       PUSH IDManfor.005D3C15
004E39FA  . 64:A1 00000000    MOV EAX,DWORD PTR FS:[0]
004E3A00  . 50                PUSH EAX
004E3A01  . 64:8925 00000000  MOV DWORD PTR FS:[0],ESP
004E3A08  . 81EC 88030000     SUB ESP,388
004E3A0E  . 53                PUSH EBX
004E3A0F  . 56                PUSH ESI
004E3A10  . 57                PUSH EDI
004E3A11  . 8D85 C0FEFFFF     LEA EAX,[LOCAL.80]
004E3A17  . 8965 F0           MOV [LOCAL.4],ESP
004E3A1A  . 6A 32             PUSH 32
004E3A1C  . 8BD9              MOV EBX,ECX
004E3A1E  . 50                PUSH EAX
004E3A1F  . 68 B0040000       PUSH 4B0
004E3A24  . 895D C8           MOV [LOCAL.14],EBX
004E3A27  . C605 2C516900 32  MOV BYTE PTR DS:[69512C],32
004E3A2E  . C605 2D516900 59  MOV BYTE PTR DS:[69512D],59
004E3A35  . C605 2E516900 4F  MOV BYTE PTR DS:[69512E],4F
004E3A3C  . C605 2F516900 50  MOV BYTE PTR DS:[69512F],50
004E3A43  . C605 30516900 42  MOV BYTE PTR DS:[695130],42
004E3A4A  . C605 31516900 33  MOV BYTE PTR DS:[695131],33
004E3A51  . C605 32516900 41  MOV BYTE PTR DS:[695132],41
004E3A58  . C605 33516900 51  MOV BYTE PTR DS:[695133],51
004E3A5F  . C605 34516900 43  MOV BYTE PTR DS:[695134],43
004E3A66  . C605 35516900 56  MOV BYTE PTR DS:[695135],56
004E3A6D  . C605 36516900 55  MOV BYTE PTR DS:[695136],55
004E3A74  . C605 37516900 58  MOV BYTE PTR DS:[695137],58
004E3A7B  . C605 38516900 4D  MOV BYTE PTR DS:[695138],4D
004E3A82  . C605 39516900 4E  MOV BYTE PTR DS:[695139],4E
004E3A89  . C605 3A516900 52  MOV BYTE PTR DS:[69513A],52
004E3A90  . C605 3B516900 53  MOV BYTE PTR DS:[69513B],53
004E3A97  . C605 3C516900 39  MOV BYTE PTR DS:[69513C],39
004E3A9E  . C605 3D516900 37  MOV BYTE PTR DS:[69513D],37
004E3AA5  . C605 3E516900 57  MOV BYTE PTR DS:[69513E],57
004E3AAC  . C605 3F516900 45  MOV BYTE PTR DS:[69513F],45
004E3AB3  . C605 40516900 30  MOV BYTE PTR DS:[695140],30
004E3ABA  . C605 41516900 49  MOV BYTE PTR DS:[695141],49
```

Let Olly run again (F9) and you will be back in the “registration” dialog, and it is still filled with your dummy info. Press “Ok” again and you will see that we break at our newly set bp.
Start stepping down through the code, all the way through all the MOV commands (if we should have made a keygen for this we should have stop here in the MOV commands to see what it does, but that is not the purpose of this tutorial). After all the MOV commands, you end here:

```
004E3B07  . C605 4C516900 31  MOV BYTE PTR DS:[69514C],31
004E3B0E  . C605 4D516900 36  MOV BYTE PTR DS:[69514D],36
004E3B15  . C605 4E516900 35  MOV BYTE PTR DS:[69514E],35
004E3B1C  . C605 4F516900 54  MOV BYTE PTR DS:[69514F],54
004E3B23  . C745 FC 00000000  MOV [LOCAL.1],0
004E3B2A  . E8 56C80C00       CALL IDManfor.005B0385
004E3B2F  . 85C0              TEST EAX,EAX
004E3B31  .v 75 25            JNZ SHORT IDManfor.004E3B58
004E3B33  . 8B0D F8596900     MOV ECX,DWORD PTR DS:[6959F8]
004E3B39  . 50                PUSH EAX
004E3B3A  . 68 AC906600       PUSH IDManfor.006690AC
004E3B3F  > 51                PUSH ECX
004E3B40  > 8BCB              MOV ECX,EBX
004E3B42  > E8 ADB30C00       CALL IDManfor.005AEEF4
004E3B47  . 8B4D F4           MOV ECX,[LOCAL.3]
004E3B4A  . 64:890D 00000000  MOV DWORD PTR FS:[0],ECX
004E3B51  . 5F                POP EDI
004E3B52  . 5E                POP ESI
004E3B53  . 5B                POP EBX
004E3B54  . 8BE5              MOV ESP,EBP
004E3B56  . 5D                POP EBP
004E3B57  . C3                RETN
004E3B58  > 8D95 28FFFFFF     LEA EDX,[LOCAL.54]
004E3B5E  . 6A 32             PUSH 32
004E3B60  . 52                PUSH EDX
004E3B61  . 68 13040000       PUSH 413
004E3B66  . 8BCB              MOV ECX,EBX
004E3B68  . E8 18C80C00       CALL IDManfor.005B0385
004E3B6D  . 85C0              TEST EAX,EAX
004E3B6F  .v 75 0E            JNZ SHORT IDManfor.004E3B7F
004E3B71  . 50                PUSH EAX
004E3B72  . A1 F4596900       MOV EAX,DWORD PTR DS:[6959F4]
004E3B77  . 68 AC906600       PUSH IDManfor.006690AC
004E3B7C  . 50                PUSH EAX
004E3B7D  .^ EB C1            JMP SHORT IDManfor.004E3B40
004E3B7F  > 8D8D F4FEFFFF     LEA ECX,[LOCAL.67]
004E3B85  . 6A 32             PUSH 32
004E3B87  . 51                PUSH ECX
004E3B88  . 68 A5040000       PUSH 4A5
004E3B8D  . 8BCB              MOV ECX,EBX
004E3B8F  . E8 F1C70C00       CALL IDManfor.005B0385
004E3B94  . 85C0              TEST EAX,EAX
004E3B96  .v 75 0F            JNZ SHORT IDManfor.004E3BA7
004E3B98  . 8B15 F0596900     MOV EDX,DWORD PTR DS:[6959F0]
004E3B9E  . 50                PUSH EAX
004E3B9F  . 68 AC906600       PUSH IDManfor.006690AC
004E3BA4  . 52                PUSH EDX
004E3BA5  .^ EB 99            JMP SHORT IDManfor.004E3B40
004E3BA7  > 8D85 7CFFFFFF     LEA EAX,[LOCAL.33]
```

A test command, and look in EAX the length of our First Name (in hexdecimal: if you like, you can convert it to decimal with Crackers’ tool). So this tests if we have entered some info in the “First Name” field, and if we have not it will not jump. But we have, so it jumps, and after some push command you will see at 4E3B6D another TEST command and yes, look in EAX: the length of our “Last name” info and jumps if empty.
I think, you get the point by now. There will be two more checks to see if we put some info in the last “to” field. The test for the “Email” field is the one at 004E3B94 and the test command for Serial number Field is 004E3BBC. Step all the way down to that test and let it jump as it did the 3 other times. Now you should be here:

```
004E3BBC  .  85C0          TEST EAX,EAX
004E3BBE  .v 75 11         JNZ SHORT IDManfor.004E3BD1
004E3BC0  .  8B0D EC596900 MOV ECX,DWORD PTR DS:[6959EC]
004E3BC6  .  50            PUSH EAX
004E3BC7  .  68 AC906600   PUSH IDManfor.006690AC                ASCII "Internet Download Manager"
004E3BCC  .^ E9 6FFFFFFF   JMP IDManfor.004E3B3F
004E3BD1  > ↳B2 20         MOV DL,20
004E3BD3  >  3895 7CFFFFFF CMP BYTE PTR SS:[EBP-84],DL
004E3BD9  .v 75 5C         JNZ SHORT IDManfor.004E3C37
004E3BDB  .  8DBD 7DFFFFFF LEA EDI,DWORD PTR SS:[EBP-83]
004E3BE1  .  83C9 FF       OR ECX,FFFFFFFF
004E3BE4  .  33C0          XOR EAX,EAX
004E3BE6  .  8DB5 0CFEFFFF LEA ESI,[LOCAL.125]
004E3BEC  .  F2:AE         REPNE SCAS BYTE PTR ES:[EDI]
004E3BEE  .  F7D1          NOT ECX                               USER32.75DFF69D
004E3BF0  .  2BF9          SUB EDI,ECX                           USER32.75DFF69D
004E3BF2  .  8975 DC       MOV [LOCAL.9],ESI                     IDManfor.005FCD18
004E3BF5  .  8BC1          MOV EAX,ECX                           USER32.75DFF69D
004E3BF7  .  8BF7          MOV ESI,EDI
004E3BF9  .  8B7D DC       MOV EDI,[LOCAL.9]                     USER32.75E090D0
004E3BFC  .  C1E9 02       SHR ECX,2
004E3BFF  .  F3:A5         REP MOVS DWORD PTR ES:[EDI],DWORD PTR DS:[ESI]
004E3C01  .  8BC8          MOV ECX,EAX
004E3C03  .  33C0          XOR EAX,EAX
004E3C05  .  83E1 03       AND ECX,3
004E3C08  .  F3:A4         REP MOVS BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]
004E3C0A  .  8DBD 0CFEFFFF LEA EDI,[LOCAL.125]
004E3C10  .  83C9 FF       OR ECX,FFFFFFFF
004E3C13  .  F2:AE         REPNE SCAS BYTE PTR ES:[EDI]
004E3C15  .  F7D1          NOT ECX                               USER32.75DFF69D
004E3C17  .  8DB5 7CFFFFFF LEA ESI,[LOCAL.33]
004E3C1D  .  2BF9          SUB EDI,ECX                           USER32.75DFF69D
004E3C1F  .  8BC1          MOV EAX,ECX                           USER32.75DFF69D
004E3C21  .  8975 DC       MOV [LOCAL.9],ESI                     IDManfor.005FCD18
004E3C24  .  8BF7          MOV ESI,EDI
004E3C26  .  8B7D DC       MOV EDI,[LOCAL.9]                     USER32.75E090D0
004E3C29  .  C1E9 02       SHR ECX,2
004E3C2C  .  F3:A5         REP MOVS DWORD PTR ES:[EDI],DWORD PTR DS:[ESI]
004E3C2E  .  8BC8          MOV ECX,EAX
004E3C30  .  83E1 03       AND ECX,3
004E3C33  .  F3:A4         REP MOVS BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]
004E3C35  .^ EB 9C         JMP SHORT IDManfor.004E3BD3
004E3C37  >  8DBD 7CFFFFFF LEA EDI,[LOCAL.33]
004E3C3D  .  83C9 FF       OR ECX,FFFFFFFF
```

Internet Dow
Please enter
First
Last

Ok, now we enter a smaller routine. As you can see here it moves 20 into DL and the next line compares it, so step down to the compare and take a look in the information window in Olly:

```
DL=20 (' ')
Stack SS:[0012A348]=31 ('1')
Jump from 004E3C35
```

You may say, “Hey, it compares our serial’s first letter with ‘nothing’, and if it is not equal then it jumps ”, but that is not correct. Olly shows you what DL is and it’s “ “, but that is because 20 is “Space” in Ascii. So what this routine does is to check if you, by accident, has press the “Spacekey” before the code. (This usually only happens when copy/pasting the serial code). If you step one time, you will also see that we jump because we have not pressed the Spacekey before our code. Ok, step about six times and you should now be at then JNZ on 004E3C47

```
004E3C42  .  F2:AE         REPNE SCAS BYTE PTR ES:[EDI]
004E3C44  .  F7D1          NOT ECX
004E3C46  .  49            DEC ECX
004E3C47  .v 75 12         JNZ SHORT IDManfor.004E3C5B
004E3C49  >  8B0D E8596900 MOV ECX,DWORD PTR DS:[6959E8]
004E3C4F  .  6A 00         PUSH 0
004E3C51  .  68 AC906600   PUSH IDManfor.006690AC
004E3C56  .^ E9 E4FEFFFF   JMP IDManfor.004E3B3F
004E3C5B  > ↳8DBD 7CFFFFFF LEA EDI,[LOCAL.33]
004E3C61  .  83C9 FF       OR ECX,FFFFFFFF
004E3C64  .  33C0          XOR EAX,EAX
```

Here if we do nothing it jumps, but is that right? If you click the next line and look into the information window, you’ll see the string to the bad boy and if you click on the jump on 004E3C56 and follow the jump (hit enter), you’ll see that it goes up to one of the places we have tried to avoid earlier. So just let Olly jump and end right before a small routine again:

```
004E3C5B  > ↳8DBD 7CFFFFFF LEA EDI,[LOCAL.33]
004E3C61  .  83C9 FF       OR ECX,FFFFFFFF
004E3C64  .  33C0          XOR EAX,EAX
004E3C66  .  F2:AE         REPNE SCAS BYTE PTR ES:[EDI]
004E3C68  .  F7D1          NOT ECX
004E3C6A  .  49            DEC ECX
004E3C6B  .  38940D 7BFFFFFF CMP BYTE PTR SS:[EBP+ECX-85],DL
004E3C72  .v 75 19         JNZ SHORT IDManfor.004E3C8D
004E3C74  .  8DBD 7CFFFFFF LEA EDI,[LOCAL.33]
004E3C7A  .  83C9 FF       OR ECX,FFFFFFFF
004E3C7D  .  33C0          XOR EAX,EAX
004E3C7F  .  F2:AE         REPNE SCAS BYTE PTR ES:[EDI]
004E3C81  .  F7D1          NOT ECX
004E3C83  .  49            DEC ECX
004E3C84  .  88840D 7BFFFFFF MOV BYTE PTR SS:[EBP+ECX-85],AL
004E3C8B  .^ EB CE         JMP SHORT IDManfor.004E3C5B
004E3C8D  >  8D95 7CFFFFFF LEA EDX,[LOCAL.33]
004E3C93  .  52            PUSH EDX
```

If you step down to 004E3C6B and look in the information window can you see what it does?

```
004E3C64  .  33C0            XOR EAX,EAX
004E3C66  .  F2:AE           REPNE SCAS BYTE PTR ES:[EDI]
004E3C68  .  F7D1            NOT ECX
004E3C6A  .  49              DEC ECX
004E3C6B  .  38940D 7BFFFFFF CMP BYTE PTR SS:[EBP+ECX-85],DL
004E3C72  .v 75 19           JNZ SHORT IDManfor.004E3C8D
004E3C74  .  8DBD 7CFFFFFF   LEA EDI,[LOCAL.33]
004E3C7A  .  83C9 FF         OR ECX,FFFFFFFF
004E3C7D  .  33C0            XOR EAX,EAX
004E3C7F  .  F2:AE           REPNE SCAS BYTE PTR ES:[EDI]
004E3C81  .  F7D1            NOT ECX
004E3C83  .  49              DEC ECX
004E3C84  .  88840D 7BFFFFFF MOV BYTE PTR SS:[EBP+ECX-85],AL
004E3C8B  .^ EB CE           JMP SHORT IDManfor.004E3C5B
004E3C8D  >  8D95 7CFFFFFF   LEA EDX,[LOCAL.33]
004E3C93  .  52              PUSH EDX
004E3C94  .  E8 923C0B00     CALL IDManfor.0059792B
004E3C99  .  8DBD 7CFFFFFF   LEA EDI,[LOCAL.33]
004E3C9F  .  83C9 FF         OR ECX,FFFFFFFF
004E3CA2  .  33C0            XOR EAX,EAX
004E3CA4  .  83C4 04         ADD ESP,4
004E3CA7  .  F2:AE           REPNE SCAS BYTE PTR ES:[EDI]
004E3CA9  .  F7D1            NOT ECX
004E3CAB  .  49              DEC ECX
004E3CAC  .  83F9 17         CMP ECX,17
004E3CAF  .^ 75 98           JNZ SHORT IDManfor.004E3C49
004E3CB1  .  8A4D 81         MOV CL,BYTE PTR SS:[EBP-7F]
004E3CB4  .  8845 EF         MOV BYTE PTR SS:[EBP-11],AL
004E3CB7  .  B0 2D           MOV AL,2D

DL=20 (' ')
Stack SS:[0012A35D]=31 ('1')
```

Yes, that is correct, it checks if there is a “spacekey”. Press F8 after the serial, and since we have not done that, it jumps on the next line down to 004E3C8D. Now step down to 004ECAF:

```
004E3CAC  .  83F9 17         CMP ECX,17
004E3CAF  .^ 75 98           JNZ SHORT IDManfor.004E3C49
004E3CB1  .  8A4D 81         MOV CL,BYTE PTR SS:[EBP-7F]
004E3CB4  .  8845 EF         MOV BYTE PTR SS:[EBP-11],AL
004E3CB7  .  B0 2D           MOV AL,2D
004E3CB9  .  3AC8            CMP CL,AL
004E3CBB  .v 75 0A           JNZ SHORT IDManfor.004E3CC7
004E3CBD  .  3845 87         CMP BYTE PTR SS:[EBP-79],AL
004E3CC0  .v 75 05           JNZ SHORT IDManfor.004E3CC7
004E3CC2  .  3845 8D         CMP BYTE PTR SS:[EBP-73],AL
004E3CC5  .v 74 04           JE SHORT IDManfor.004E3CCB
004E3CC7  >  C645 EF 01      MOV BYTE PTR SS:[EBP-11],1
004E3CCB  >  8D85 7CFFFFFF   LEA EAX,[LOCAL.33]
004E3CD1  .  6A 05           PUSH 5
004E3CD3  .  8D4D DC         LEA ECX,[LOCAL.9]
004E3CD6  .  50              PUSH EAX
004E3CD7  .  51              PUSH ECX
```

Take a look in ECX. There you now have the length of our serial (hex 17= 23 dec.). Here the target checks to see if the Serial we put in has the right length and jumps if not. As we already know the format of the serial and the length, Olly will not jump. Step down to CMP CL,AL and look in the information window:

```
004E3CB9  .  3AC8            CMP CL,AL
004E3CBB  .v 75 0A           JNZ SHORT IDManfor.004E3CC7
004E3CBD  .  3845 87         CMP BYTE PTR SS:[EBP-79],AL
004E3CC0  .v 75 05           JNZ SHORT IDManfor.004E3CC7
004E3CC2  .  3845 8D         CMP BYTE PTR SS:[EBP-73],AL
004E3CC5  .v 74 04           JE SHORT IDManfor.004E3CCB
004E3CC7  >  C645 EF 01      MOV BYTE PTR SS:[EBP-11],1
004E3CCB  >  8D85 7CFFFFFF   LEA EAX,[LOCAL.33]
004E3CD1  .  6A 05           PUSH 5
004E3CD3  .  8D4D DC         LEA ECX,[LOCAL.9]
004E3CD6  .  50              PUSH EAX
004E3CD7  .  51              PUSH ECX
004E3CD8  .  E8 23120B00     CALL IDManfor.00594F00
004E3CDD  .  8D55 82         LEA EDX,DWORD PTR SS:[EBP-7E]
004E3CE0  .  6A 05           PUSH 5
004E3CE2  .  8D45 CC         LEA EAX,[LOCAL.13]
004E3CE5  .  52              PUSH EDX
004E3CE6  .  50              PUSH EAX
004E3CE7  .  E8 14120B00     CALL IDManfor.00594F00
004E3CEC  .  8D4D 88         LEA ECX,[LOCAL.30]
004E3CEF  .  6A 05           PUSH 5
004E3CF1  .  8D55 D4         LEA EDX,[LOCAL.11]
004E3CF4  .  51              PUSH ECX
004E3CF5  .  52              PUSH EDX
004E3CF6  .  E8 05120B00     CALL IDManfor.00594F00
004E3CFB  .  8D45 8E         LEA EAX,DWORD PTR SS:[EBP-72]
004E3CFE  .  6A 05           PUSH 5
004E3D00  .  8D4D B0         LEA ECX,[LOCAL.20]
004E3D03  .  50              PUSH EAX
004E3D04  .  51              PUSH ECX
004E3D05  .  E8 F6110B00     CALL IDManfor.00594F00
004E3D0A  .  33FF            XOR EDI,EDI
004E3D0C  .  83C4 30         ADD ESP,30
004E3D0F  .  C645 E1 00      MOV BYTE PTR SS:[EBP-1F],0
004E3D13  .  C645 D1 00      MOV BYTE PTR SS:[EBP-2F],0
004E3D17  .  C645 D9 00      MOV BYTE PTR SS:[EBP-27],0
004E3D1B  .  C645 B5 00      MOV BYTE PTR SS:[EBP-4B],0
004E3D1F  .  897D E8         MOV [LOCAL.6],EDI
004E3D22  .  33F6            XOR ESI,ESI
004E3D24  >  83FE 05         CMP ESI,5
004E3D27  .v 7D 32           JGE SHORT IDManfor.004E3D5B
004E3D29  .  8A5435 DC       MOV DL,BYTE PTR SS:[EBP+ESI-24]
004E3D2D  .  83C9 FF         OR ECX,FFFFFFFF
004E3D30  .  33C0            XOR EAX,EAX
004E3D32  >  83F8 24         CMP EAX,24
004E3D35  .v 7D 0A           JGE SHORT IDManfor.004E3D41
004E3D37  .  3890 2C516900   CMP BYTE PTR DS:[EAX+69512C],DL
004E3D3D  .v 75 15           JNZ SHORT IDManfor.004E3D54
004E3D3F  .  8BC8            MOV ECX,EAX
004E3D41  >  83F9 FF         CMP ECX,-1
004E3D44  .v 74 11           JE SHORT IDManfor.004E3D57
004E3D46  .  8D14FF          LEA EDX,DWORD PTR DS:[EDI+EDI*8]
004E3D49  .  03CF            ADD ECX,EDI
004E3D4B  .  46              INC ESI

AL=2D ('-')
CL=2D ('-')
```

This compare command, the CMP on 0043CBD and on 0043CC2, checks if there is the 3 “-“ in our serial as there should be. But again, thanks to our good study of the target, we know we have the 3 “-“, so let’s just step over them. You can step all the way down to 004E3D24, because the code until that just splits our serial up in 4 segments with 5 digits in each. It does that to prepare itself for the now coming serial checking routine.

```
004E3D24  > 83FE 05          CMP ESI,5
004E3D27  .v 7D 32           JGE SHORT IDManfor.004E3D5B
004E3D29  . 8A5435 DC        MOV DL,BYTE PTR SS:[EBP+ESI-24]
004E3D2D  . 83C9 FF          OR ECX,FFFFFFFF
004E3D30  . 33C0             XOR EAX,EAX
004E3D32  > 83F8 24          CMP EAX,24
004E3D35  .v 7D 0A           JGE SHORT IDManfor.004E3D41
004E3D37  . 3890 2C516900    CMP BYTE PTR DS:[EAX+69512C],DL
004E3D3D  .v 75 15           JNZ SHORT IDManfor.004E3D54
004E3D3F  . 8BC8             MOV ECX,EAX
004E3D41  > 83F9 FF          CMP ECX,-1
004E3D44  .v 74 11           JE SHORT IDManfor.004E3D57
004E3D46  . 8D14FF           LEA EDX,DWORD PTR DS:[EDI+EDI*8]
004E3D49  . 03CF             ADD ECX,EDI
004E3D4B  . 46               INC ESI
004E3D4C  . 8D3C91           LEA EDI,DWORD PTR DS:[ECX+EDX*4]
004E3D4F  . 897D E8          MOV [LOCAL.6],EDI
004E3D52  .^ EB D0           JMP SHORT IDManfor.004E3D24
004E3D54  > 40               INC EAX
004E3D55  .^ EB DB           JMP SHORT IDManfor.004E3D32
004E3D57  > C645 EF 01       MOV BYTE PTR SS:[EBP-11],1
004E3D5B  > 33FF             XOR EDI,EDI
004E3D5D  . 33F6             XOR ESI,ESI
004E3D5F  . 897D E4          MOV [LOCAL.7],EDI
004E3D62  > 83FE 05          CMP ESI,5
004E3D65  .v 7D 32           JGE SHORT IDManfor.004E3D99
004E3D67  . 8A5435 CC        MOV DL,BYTE PTR SS:[EBP+ESI-34]
```

Okay, let's talk a little about this. This is what it prepared itself for. If you scroll down you'll see that this routine is used 4 times; one time for each piece of code, and it does the same. It calculates the value for the right code and saves it for a later compare. Nothing will happen here that we need to think about for patching, so let's set a bp after these 4 routines on the first Test right here:

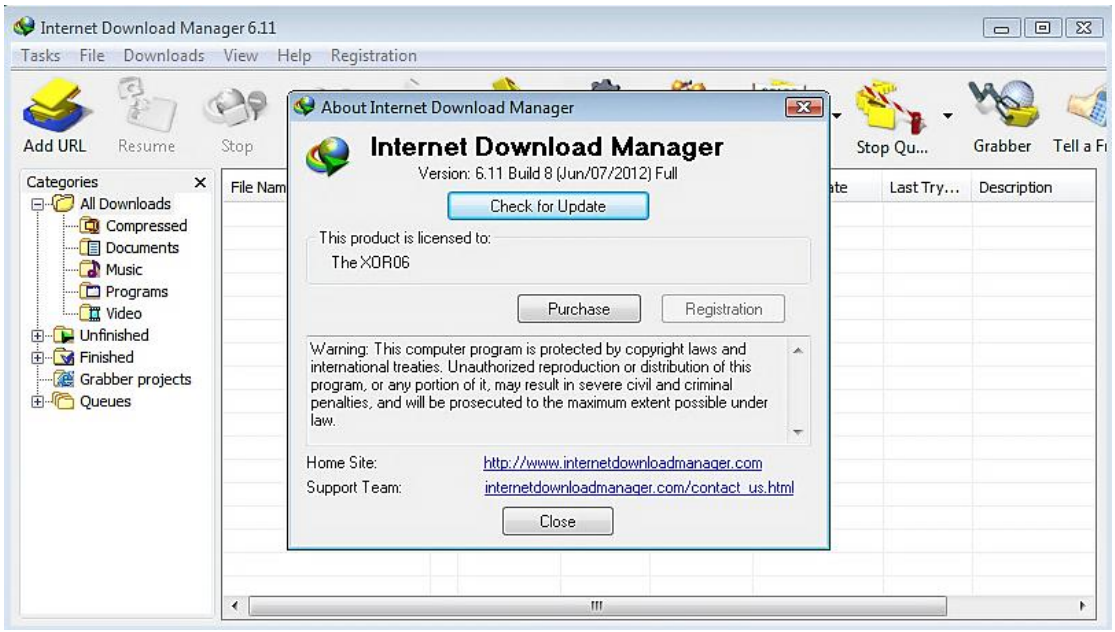
```
004E3E08  . 84C0             TEST AL,AL
004E3E0A  .v 74 16           JE SHORT IDManfor.004E3E22
004E3E0C  > 8B0D E8596900    MOV ECX,DWORD PTR DS:[6959E8]
004E3E12  . 6A 00            PUSH 0
004E3E14  . 68 AC906600      PUSH IDManfor.006690AC
004E3E19  . 51               PUSH ECX
004E3E1A  . 8B4D C8          MOV ECX,[LOCAL.14]
004E3E1D  .^ E9 20FDFFFF     JMP IDManfor.004E3B42
004E3E22  > 8B4D E8          MOV ECX,[LOCAL.6]
004E3E25  . BE 2B000000      MOV ESI,2B
004E3E2A  . 8BC1             MOV EAX,ECX
004E3E2C  . 99               CDQ
004E3E2D  . F7FE             IDIV ESI
004E3E2F  . 85D2             TEST EDX,EDX
004E3E31  .v 75 04           JNZ SHORT IDManfor.004E3E37
004E3E33  . 85C9             TEST ECX,ECX
004E3E35  .v 75 04           JNZ SHORT IDManfor.004E3E3B
004E3E37  > C645 EF 01       MOV BYTE PTR SS:[EBP-11],1
004E3E3B  > 8B4D E4          MOV ECX,[LOCAL.7]
004E3E3E  . BE 17000000      MOV ESI,17
004E3E43  . 8BC1             MOV EAX,ECX
004E3E45  . 99               CDQ
004E3E46  . F7FE             IDIV ESI
004E3E48  . 85D2             TEST EDX,EDX
004E3E4A  .v 75 04           JNZ SHORT IDManfor.004E3E50
004E3E4C  . 85C9             TEST ECX,ECX
004E3E4E  .v 75 04           JNZ SHORT IDManfor.004E3E54
004E3E50  > C645 EF 01       MOV BYTE PTR SS:[EBP-11],1
004E3E54  > 8BC3             MOV EAX,EBX
004E3E56  . B9 11000000      MOV ECX,11
004E3E5B  . 99               CDQ
004E3E5C  . F7F9             IDIV ECX
004E3E5E  . 85D2             TEST EDX,EDX
004E3E60  .v 75 04           JNZ SHORT IDManfor.004E3E66
004E3E62  . 85DB             TEST EBX,EBX
004E3E64  .v 75 04           JNZ SHORT IDManfor.004E3E6A
004E3E66  > C645 EF 01       MOV BYTE PTR SS:[EBP-11],1
004E3E6A  > 8BC7             MOV EAX,EDI
004E3E6C  . B9 35000000      MOV ECX,35
004E3E71  . 99               CDQ
004E3E72  . F7F9             IDIV ECX
004E3E74  . 85D2             TEST EDX,EDX
004E3E76  .v 75 0B           JNZ SHORT IDManfor.004E3E83
004E3E78  . 85FF             TEST EDI,EDI
004E3E7A  .v 74 07           JE SHORT IDManfor.004E3E83
004E3E7C  . 8A45 EF          MOV AL,BYTE PTR SS:[EBP-11]
004E3E7F  . 84C0             TEST AL,AL
004E3E81  .v 74 16           JE SHORT IDManfor.004E3E99
004E3E83  > 8B15 E8596900    MOV EDX,DWORD PTR DS:[6959E8]
004E3E89  . 8B4D C8          MOV ECX,[LOCAL.14]
004E3E8C  . 6A 00            PUSH 0
004E3E8E  . 68 AC906600      PUSH IDManfor.006690AC
004E3E93  . 52               PUSH EDX
004E3E94  .^ E9 A9FCFFFF     JMP IDManfor.004E3B42
```

Now you can step past all of these tests until 004E3E76:

```
004E3E76  .v 75 0B           JNZ SHORT IDManfor.004E3E83
004E3E78  . 85FF             TEST EDI,EDI
004E3E7A  .v 74 07           JE SHORT IDManfor.004E3E83
004E3E7C  . 8A45 EF          MOV AL,BYTE PTR SS:[EBP-11]
004E3E7F  . 84C0             TEST AL,AL
004E3E81  .v 74 16           JE SHORT IDManfor.004E3E99
004E3E83  > 8B15 E8596900    MOV EDX,DWORD PTR DS:[6959E8]
004E3E89  . 8B4D C8          MOV ECX,[LOCAL.14]
004E3E8C  . 6A 00            PUSH 0
004E3E8E  . 68 AC906600      PUSH IDManfor.006690AC                  ASCII "Internet Download Manager"
004E3E93  . 52               PUSH EDX
004E3E94  .^ E9 A9FCFFFF     JMP IDManfor.004E3B42
004E3E99  > 8D45 C4          LEA EAX,[LOCAL.15]
004E3E9C  . 6A 00            PUSH 0                                  pDisposition = NULL
004E3E9E  . 50               PUSH EAX                                pHandle = 0011BEF9
004E3E9F  . 6A 00            PUSH 0                                  pSecurity = NULL
004E3EA1  . 68 3F000F00      PUSH 0F003F                             Access = KEY_ALL_ACCESS
004E3EA6  . 6A 00            PUSH 0                                  Options = REG_OPTION_NON_VOLATILE
004E3EA8  . 6A 00            PUSH 0                                  Class = NULL
004E3EAA  . 6A 00            PUSH 0                                  Reserved = 0
004E3EAC  . 68 68BE6600      PUSH IDManfor.00668E60                  Subkey = "SOFTWARE\\Internet Download Manager"
004E3EB1  . 68 02000080      PUSH 80000002                           hKey = HKEY_LOCAL_MACHINE
004E3EB6  . FF15 14D05D00    CALL DWORD PTR DS:[<&ADVAPI32.RegCreateKeyExA>]  RegCreateKeyExA
004E3EBC  . 85C0             TEST EAX,EAX
004E3EBE  .v 74 09           JE SHORT IDManfor.004E3EC9
```

Let's think a little. Why do I stop here? Well, that's because if this jumps we can't avoid the JMP at 004E3E99. And why would I like to avoid that? First, because it jumps back to a badboy, but also if you look after the JMP you'll see that there is a call to RegCreateKeyExA . That must be the right way, but why should it open the registry if we enter a wrong serial? It only makes good sense to open the registry if it thinks we have entered a correct serial and now is preparing to write it in to the registry.

Place a bp here and change the flags so it does not jump. The next JNZ at 004E3E7A will not jump so just step over that. Then we come to JE on 004E3E81. Here Olly will not jump, but we just found out that we need to avoid the JMP command at 004E3E99. Place a bp and change the flags so we jump.
Press F9 and now the target runs without the Badboy showing!! If you try to choose the Registration menu, you'll see the "Registration" is grayed out. Try to choose the about dialog and you'll see this:



Click "close" and return till Olly and lets us make the serial check patch

Applying the Serial patch

In Olly, go to the first bp (open the bp list and click)

The first JNZ on line 004E3E76 could just be NOPed out, but then we should also change the JE to a JMP. Instead we could just change the "JNZ SHORT 004E3E83" at 004E3E76 to "JMP SHORT 004E3E99", so let's do that and save the changes to file. *(Actually, you can make the target jump to 004E3E99 before it checks the format on the serial- you can play with it if you like.)*

Run your new file and you'll see that it runs fine.

Going Online

Now, when you're sure the program works offline, Connect to the net. We now try to provoke the target to make a server check so we can patch it. Open your browser, download a file, and open the about box again on your program to provoke the target to make a server check.

(If all other attempts fail you can cheat and open the program in Olly. Goto 66C910 in the dump and change the byte from 1 to 0 save it and restart the target).

Finding the First Server Check

You now should have seen this nag

If you try to go offline, you will find out that it now does not work even if you're online. So let's open the target in Olly again. Run it and the Nag appears. Let us use the same trick as on the badboy from the serial check. Pause Olly, Press ALT+F9 (Execute till user-code) and Click Ok and, BAM, Olly breaks here:

```
0043CACA   .  E8 31441700    CALL IDManfor.005B0F00
0043CACF   .^ E9 E4FEFFFF    JMP IDManfor.0043C9B8
0043CAD4   >  8B0D DC596900  MOV ECX,DWORD PTR DS:[6959DC]          Case 3 of switch 0043C996
0043CADA   .  8B35 90D75D00  MOV ESI,DWORD PTR DS:[<&USER32.MessageBoxA>]  USER32.MessageBoxA
0043CAE0   .  68 30000500    PUSH 50030                             Style = MB_OK|MB_ICONEXCLAMATION|MB_APPLMODAL|50000
0043CAE5   .  68 AC906600    PUSH IDManfor.006690AC                 Title = "Internet Download Manager"
0043CAEA   .  51             PUSH ECX                               Text = 0000008E ???
0043CAEB   .  6A 00          PUSH 0                                 hOwner = NULL
0043CAED   .  FFD6           CALL ESI                               MessageBoxA
0043CAEF   .^ E9 C4FEFFFF    JMP IDManfor.0043C9B8
0043CAF4   >  8B15 08536900  MOV EDX,DWORD PTR DS:[695308]          Case 7 of switch 0043C996
```

Olly breaks again right after the MessageBoxA (showing the nag) but how did it come here? Look at address 0043CAD4. Olly shows us that a jump ends here and right before, at line 0043CACF, there is a jump "back in the code", so we can only come to this code by a jump that ends at 0043CAD4. Click on it and look in the information window where it comes from and follow it. You should be here now:

```
0043C92F   .  85F6           TEST ESI,ESI
0043C931   .v 74 40          JE SHORT IDManfor.0043C973
0043C933   .  68 1CD36600    PUSH IDManfor.0066D31C                 ASCII "DllRegisterServer"
0043C938   .  56             PUSH ESI
0043C939   .  FFD7           CALL EDI
0043C93B   .  85C0           TEST EAX,EAX
0043C93D   .v 74 02          JE SHORT IDManfor.0043C941
0043C93F   .  FFD0           CALL EAX
0043C941   >  8A85 93FEFFFF  MOV AL,BYTE PTR SS:[EBP-16D]
0043C947   .  84C0           TEST AL,AL
0043C949   .v 74 09          JE SHORT IDManfor.0043C954
0043C94B   .  A1 184D6900    MOV EAX,DWORD PTR DS:[694D18]
0043C950   .  85C0           TEST EAX,EAX
0043C952   .v 75 18          JNZ SHORT IDManfor.0043C96C
0043C954   >  68 A4D16600    PUSH IDManfor.0066D1A4                 ASCII "ControlMonitoring"
0043C959   .  56             PUSH ESI
0043C95A   .  FFD7           CALL EDI
0043C95C   .  85C0           TEST EAX,EAX
0043C95E   .v 74 0C          JE SHORT IDManfor.0043C96C
0043C960   .  6A 00          PUSH 0
0043C962   .  6A 00          PUSH 0
0043C964   .  6A 01          PUSH 1
0043C966   .  FFD0           CALL EAX
0043C968   .  85C0           TEST EAX,EAX
0043C96A   .v 74 07          JE SHORT IDManfor.0043C973
0043C96C   >  56             PUSH ESI                               hLibModule = 000002A0 (window)
0043C96D   .  FF15 A8D35D00  CALL DWORD PTR DS:[<&KERNEL32.FreeLibrary>]  FreeLibrary
0043C973   >  6A 01          PUSH 1                                 Real = 00000001
0043C975   .  E8 A6090000    CALL IDManfor.0043D320                 IDManfor.0043D320
0043C97A   .  8BF8           MOV EDI,EAX
0043C97C   .  83C4 04        ADD ESP,4
0043C97F   .  897D EC        MOV DWORD PTR SS:[EBP-14],EDI
0043C982   .  8B45 EC        MOV EAX,DWORD PTR SS:[EBP-14]
0043C985   .  85C0           TEST EAX,EAX
0043C987   .v 74 0A          JE SHORT IDManfor.0043C993
0043C989   .  C705 2C4D6900 01 MOV DWORD PTR DS:[694D2C],1
0043C993   >  8B45 EC        MOV EAX,DWORD PTR SS:[EBP-14]
0043C996   .  83F8 07        CMP EAX,7                              Switch (cases 0..7)
0043C999   .v 77 17          JA SHORT IDManfor.0043C9B2
0043C99B   .v FF2485 DCD24300 JMP DWORD PTR DS:[EAX*4+43D2DC]       IDManfor.0043CAD4
0043C9A2   >  8B4D B8        MOV ECX,DWORD PTR SS:[EBP-48]          IDManfor.00694808; Case 0 of switch 0043C996
0043C9A5   .  E8 E61A0000    CALL IDManfor.0043E490
```

The only jump to get past this is that JA on 0043C999 but if you change that It just goes to the registration dialog. Now use your brain; the command on 0043C99B: JMP DWORD PTR DS:[EAX*4+43D2DC]

Where does this JUMP command jump to? It is not a constant address,;it uses EAX to calculate where it should go. So if we changes EAX it should go some place else right?
Then again take a look at the JE on 0043C987. It jumps if EAX (again it is EAX who decides) is Zero, and here is what it jumps over; “MOV DWORD PTR DS:[694D2C],1”
That looks like some kind of a pointer to me, let us take a look. Set a bp on that (j.e. on 0043C987) and restart the target.

(*Do you understand that because both a pointer and the jump to the Nag, as well as the switch, is being decided by EAX? It has awaken my interest. Now I must see if the JE is taken or not because I don’t know if the pointer should be 0 or 1. Thereafter, I’ll trace in to find where it sets EAX but I’ll wait until I know what the pointer should be. Remember, only if EAX is ZERO will it jump past the pointer*).

```
0043C973   .  6A 01            PUSH 1                                   Arg1 = 00000001
0043C975   .  E8 A6090000      CALL IDManfor.0043D320                   IDManfor.0043D320
0043C97A   .  8BF8             MOV EDI,EAX
0043C97C   .  83C4 04          ADD ESP,4
0043C97F   .  897D EC          MOV DWORD PTR SS:[EBP-14],EDI
0043C982   .  8B45 EC          MOV EAX,DWORD PTR SS:[EBP-14]
0043C985   .  85C0             TEST EAX,EAX
0043C987   .v 74 0A            JE SHORT IDManfor.0043C993
0043C989   . | C705 2C4D6900 01 MOV DWORD PTR DS:[694D2C],1
0043C993   > 8B45 EC           MOV EAX,DWORD PTR SS:[EBP-14]
0043C996   .  83F8 07          CMP EAX,7                                Switch (cases 0..7)
0043C999   .v 77 17            JA SHORT IDManfor.0043C9B2
0043C99B   .v FF2485 DCD24300  JMP DWORD PTR DS:[EAX*4+43D2DC]           IDManfor.0043CAD4
```

Well, it does NOT jump, so we can assume that the pointer should NOT be 1. Okay, then we need this to jump and we know it only does that if EAX is zero, so let’s go into the call at 0043C975 where it sets EAX. Look after what EAX is now. It is 3. Now we know what to look for in the call. Set a bp on that call and restart. *(You cannot check it by changing the flags because that will only change the pointer, not the destination address for the jump.)*
Olly breaks at the Call. Press F7 (step in) and you get here

```
0043D320   $  55               PUSH EBP
0043D321   .  8BEC             MOV EBP,ESP
0043D323   .  6A FF            PUSH -1
0043D325   .  68 1B645C00      PUSH IDManfor.005C641B                   SE handler installation
0043D32A   .  64:A1 00000000   MOV EAX,DWORD PTR FS:[0]
0043D330   .  50               PUSH EAX
0043D331   .  64:8925 00000000 MOV DWORD PTR FS:[0],ESP
0043D338   .  81EC A0040000    SUB ESP,4A0
0043D33E   .  53               PUSH EBX                                 ADVAPI32.75F67618
0043D33F   .  56               PUSH ESI
0043D340   .  57               PUSH EDI                                 kernel32.GetProcAddress
0043D341   .  33D2             XOR EDX,EDX                              ntdll.KiFastSystemCallRet
0043D343   .  32DB             XOR BL,BL
0043D345   .  8965 F0          MOV [LOCAL.4],ESP
0043D348   .  8955 B8          MOV [LOCAL.18],EDX                       ntdll.KiFastSystemCallRet
0043D34B   .  885D DE          MOV BYTE PTR SS:[EBP-22],BL
0043D34E   .  885D DF          MOV BYTE PTR SS:[EBP-21],BL
0043D351   .  885D E7          MOV BYTE PTR SS:[EBP-19],BL
0043D354   .  8955 FC          MOV [LOCAL.1],EDX                        ntdll.KiFastSystemCallRet
0043D357   .  C745 B4 6C745E00 MOV [LOCAL.19],IDManfor.005E746C
0043D35E   .  B9 40000000      MOV ECX,40
0043D363   .  33C0             XOR EAX,EAX
0043D365   .  8DBD 80FEFFFF    LEA EDI,[LOCAL.96]
0043D36B   .  8845 80          MOV BYTE PTR SS:[EBP-80],AL
0043D36E   .  F3:AB            REP STOS DWORD PTR ES:[EDI]
0043D370   .  8B0D 74536900    MOV ECX,DWORD PTR DS:[695374]
0043D376   .  8D45 EC          LEA EAX,[LOCAL.5]
0043D379   .  50               PUSH EAX                                 pHandle = 00000001
0043D37A   .  6A 01            PUSH 1                                   Access = KEY_QUERY_VALUE
0043D37C   .  52               PUSH EDX                                 Reserved = 77285CD4
0043D37D   .  51               PUSH ECX                                 Subkey = "┬\x04"
0043D37E   .  68 02000080      PUSH 80000002                            hKey = HKEY_LOCAL_MACHINE
0043D383   .  C645 FC 01       MOV BYTE PTR SS:[EBP-4],1
0043D387   .  FF15 08D05D00    CALL DWORD PTR DS:[<&ADVAPI32.RegOpenKeyExA>]   RegOpenKeyExA
0043D38D   .  85C0             TEST EAX,EAX
0043D38F   .  BF 32000000      MOV EDI,32
0043D394   .v 0F85 A8000000    JNZ IDManfor.0043D442
0043D39A   .  8B0D 80536900    MOV ECX,DWORD PTR DS:[695380]
0043D3A0   .  8B35 04D05D00    MOV ESI,DWORD PTR DS:[<&ADVAPI32.RegQueryValueExA>]   ADVAPI32.RegQueryValueExA
0043D3A6   .  8D55 E8          LEA EDX,[LOCAL.6]
0043D3A9   .  8D85 4CFEFFFF    LEA EAX,[LOCAL.109]
0043D3AF   .  52               PUSH EDX                                 pBufSize = ntdll.KiFastSystemCallRet
0043D3B0   .  8B55 EC          MOV EDX,[LOCAL.5]
0043D3B3   .  50               PUSH EAX                                 Buffer = 00000001
0043D3B4   .  6A 00            PUSH 0                                   pValueType = NULL
0043D3B6   .  6A 00            PUSH 0                                   Reserved = NULL
0043D3B8   .  51               PUSH ECX                                 ValueName = "┬\x04"
0043D3B9   .  52               PUSH EDX                                 hKey = 77285CD4
0043D3BA   .  897D E8          MOV [LOCAL.6],EDI                        kernel32.GetProcAddress
0043D3BD   .  FFD6             CALL ESI                                 RegQueryValueExA
0043D3BF   .  85C0             TEST EAX,EAX
0043D3C1   .v 75 02            JNZ SHORT IDManfor.0043D3C5
0043D3C3   .  B3 01            MOV BL,1
0043D3C5   >  8B15 7C536900    MOV EDX,DWORD PTR DS:[69537C]
```

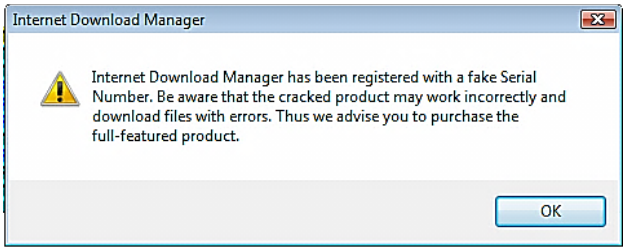
Now scroll down. We are looking for a something like MOV EAX,3 and then a RETN command (because we know EAX was 3 after the call). Keep scrolling- it is a long Routine, until you come here:

```
0043DDAD  .  8D4D CC          LEA ECX,[LOCAL.13]
0043DDB0  .  C645 FC 01       MOV BYTE PTR SS:[EBP-4],1
0043DDB4  .  E8 E7440A00      CALL IDManfor.004E22A0
0043DDB9  .  B8 03000000      MOV EAX,3
0043DDBE  .  8B4D F4          MOV ECX,[LOCAL.3]
0043DDC1  .  64:890D 00000000 MOV DWORD PTR FS:[0],ECX          ADVAPI32.75F679B4
0043DDC8  .  5F               POP EDI                           IDManfor.0043C97A
0043DDC9  .  5E               POP ESI                           IDManfor.0043C97A
0043DDCA  .  5B               POP EBX                           IDManfor.0043C97A
0043DDCB  .  8BE5             MOV ESP,EBP
0043DDCD  .  5D               POP EBP                           IDManfor.0043C97A
0043DDCE  .  C3               RETN
0043DDCF  >  A1 90526900      MOV EAX,DWORD PTR DS:[695290]
0043DDD4  .  8DB5 80FEFFFF    LEA ESI,[LOCAL.96]
0043DDDA  >  8A10             MOV DL,BYTE PTR DS:[EAX]
```

There at 0043DDB9 set a bp and see if Olly break there when you run the program. BAM, Olly breaks

```
0043DDB9  .  B8 03000000      MOV EAX,3
0043DDBE  .  8B4D F4          MOV ECX,[LOCAL.3]
0043DDC1  .  64:890D 00000000 MOV DWORD PTR FS:[0],ECX
0043DDC8  .  5F               POP EDI                           kernel32.GetProcAddress
0043DDC9  .  5E               POP ESI                           kernel32.GetProcAddress
0043DDCA  .  5B               POP EBX                           kernel32.GetProcAddress
0043DDCB  .  8BE5             MOV ESP,EBP
0043DDCD  .  5D               POP EBP                           kernel32.GetProcAddress
0043DDCE  .  C3               RETN
0043DDCF  >  A1 90526900      MOV EAX,DWORD PTR DS:[695290]
```

Now you know EAX should be 0, so let's change the code so it moves 0 into EAX instead of 3. Press space and change the code to MOV EAX, 0. Hit "Enter" (and thereafter press "Cancel"), then press F9. Yes! The target starts and now you can feel joy for about 4 sec. before a new nag pops up:



A new Server check FFS!! Do not panic we will kill this one to

Finding the Second Server Check

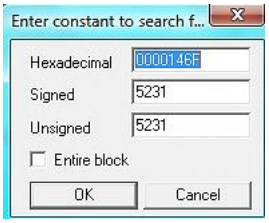
Let's use the same approach as at the first nag. Pause Olly, Press ALT+F9 (Execute till user-code) and Click Ok and BAM Olly breaks here:

```
0044BAC2  .  68 E4DB6600      PUSH IDMan.0066DBE4                ASCII "506938841"
0044BAC7  .  8D55 A0          LEA EDX,DWORD PTR SS:[EBP-60]
0044BACA  .  51               PUSH ECX
0044BACB  .  52               PUSH EDX
0044BACC  .  8D4D 0C          LEA ECX,DWORD PTR SS:[EBP+C]
0044BACF  .  E8 AC25FFFF      CALL IDMan.0043E080
0044BAD4  .  FF15 8CD75D00    CALL DWORD PTR DS:[<&USER32.GetDesktopWindow>]   GetDesktopWindow
0044BADA  .  50               PUSH EAX
0044BADB  .  E8 1D1D1600      CALL IDMan.005AD7FD
0044BAE0  .  8B40 1C          MOV EAX,DWORD PTR DS:[EAX+1C]
0044BAE3  .  8D4D A0          LEA ECX,DWORD PTR SS:[EBP-60]
0044BAE6  .  6A 30            PUSH 30                            Style = MB_OK|MB_ICONEXCLAMATION|MB_APPLMODAL
0044BAE8  .  8D95 30FEFFFF    LEA EDX,DWORD PTR SS:[EBP-1D0]
0044BAEE  .  51               PUSH ECX                           Title = "/@%xFF%x0F"
0044BAEF  .  52               PUSH EDX                           Text = "þ/@%xFF%x0F"
0044BAF0  .  50               PUSH EAX                           hOwner = 00000001
0044BAF1  .  FF15 90D75D00    CALL DWORD PTR DS:[<&USER32.MessageBoxA>]        MessageBoxA
0044BAF7  .  8B4B 1C          MOV ECX,DWORD PTR DS:[EBX+1C]
0044BAFA  .  6A 00            PUSH 0                             lParam = 0
0044BAFC  .  68 70140000      PUSH 1470                          wParam = 1470
0044BB01  .  68 11010000      PUSH 111                           Message = WM_COMMAND
0044BB06  .  51               PUSH ECX                           hWnd = 380095
0044BB07  .  FF15 88D75D00    CALL DWORD PTR DS:[<&USER32.PostMessageA>]       PostMessageA
0044BB0D  .^ E9 BBBDFFFF      JMP IDMan.004478CD
0044BB12  >  8B15 C4016900    MOV EDX,DWORD PTR DS:[6901C4]      IDMan.006901D8; Case 146C of switch 00446A9C
0044BB18  .  8955 EC          MOV DWORD PTR SS:[EBP-14],EDX
0044BB1B  .  8B45 0C          MOV EAX,DWORD PTR SS:[EBP+C]       IDMan.005E746C
0044BB1E  .  8B0D 04536900    MOV ECX,DWORD PTR DS:[695304]
0044BB24  .  50               PUSH EAX
```

You can see we now are in a switch. Scroll up and let us see where this is called from. Keep scrolling and you come to this

```
0044B3A7  >  8B55 0C          MOV EDX,DWORD PTR SS:[EBP+C]          IDMan.005E746C; Case 146B of switch 00446A9C
0044B3AA  .  57               PUSH EDI
0044B3AB  .  6A 02            PUSH 2
0044B3AD  .  52               PUSH EDX
0044B3AE  .  8BCB             MOV ECX,EBX
0044B3B0  .  E8 B80A0100      CALL IDMan.0045BE70
0044B3B5  .^ E9 13C5FFFF      JMP IDMan.004478CD
0044B3BA  >  8B45 0C          MOV EAX,DWORD PTR SS:[EBP+C]          IDMan.005E746C; Case 1485 of switch 00446A9C
0044B3BD  .  57               PUSH EDI
0044B3BE  .  6A 05            PUSH 5
0044B3C0  .  50               PUSH EAX
0044B3C1  .  8BCB             MOV ECX,EBX
0044B3C3  .  E8 A80A0100      CALL IDMan.0045BE70
0044B3C8  .^ E9 00C5FFFF      JMP IDMan.004478CD
0044B3CD  >  C745 0C 6C745E00 MOV DWORD PTR SS:[EBP+C],IDMan.005E746C   Case 146F of switch 00446A9C
0044B3D4  .  B0 F6            MOV AL,0F6
0044B3D6  .  B1 A8            MOV CL,0A8
0044B3D8  .  C645 FC 39       MOV BYTE PTR SS:[EBP-4],39
0044B3DC  .  C685 30FEFFFF A9 MOV BYTE PTR SS:[EBP-1D0],0A9
0044B3E3  .  C685 31FEFFFF 9E MOV BYTE PTR SS:[EBP-1CF],9E
0044B3EA  .  C685 32FEFFFF BC MOV BYTE PTR SS:[EBP-1CE],0BC
0044B3F1  .  C685 33FEFFFF FE MOV BYTE PTR SS:[EBP-1CD],0FE
0044B3F8  .  C685 34FEFFFF 7E MOV BYTE PTR SS:[EBP-1CC],7B
0044B3FF  .  C685 35FEFFFF 71 MOV BYTE PTR SS:[EBP-1CB],71
0044B406  .  C685 36FEFFFF 69 MOV BYTE PTR SS:[EBP-1CA],69
0044B40D  .  C685 37FEFFFF 21 MOV BYTE PTR SS:[EBP-1C9],21
0044B414  .  C685 38FEFFFF F0 MOV BYTE PTR SS:[EBP-1C8],0F0
0044B41B  .  C685 39FEFFFF D9 MOV BYTE PTR SS:[EBP-1C7],0D9
0044B422  .  C685 3AFEFFFF 76 MOV BYTE PTR SS:[EBP-1C6],76
0044B429  .  C685 3BFEFFFF CD MOV BYTE PTR SS:[EBP-1C5],0CD
```

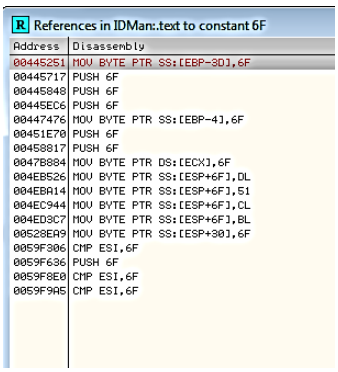
Now make a search for constants for this case (146F) and press “Ok”



You only find this one hit:

```
00451E53  .  C686 01020000 01 MOV BYTE PTR DS:[ESI+201],1
00451E5A  .  8B4D F4          MOV ECX,DWORD PTR SS:[EBP-C]
00451E5D  .  64:890D 00000000 MOV DWORD PTR FS:[0],ECX
00451E64  .  5F               POP EDI                               0012E5A0
00451E65  .  5E               POP ESI                               0012E5A0
00451E66  .  5B               POP EBX                               0012E5A0
00451E67  .  8BE5             MOV ESP,EBP
00451E69  .  5D               POP EBP                               0012E5A0
00451E6A  .  C2 0400          RETN 4
00451E6D  >  8B56 1C          MOV EDX,DWORD PTR DS:[ESI+1C]         Case 6F of switch 00451A03
00451E70  .  6A 6F            PUSH 6F                               TimerID = 6F (111.)
00451E72  .  52               PUSH EDX                              hWnd = 0000006E ('Quick Launch',class='ToolbarWindow32',parent=
00451E73  .  FF15 74D75D00    CALL DWORD PTR DS:[<&USER32.KillTimer>]  KillTimer
00451E79  .  8B46 1C          MOV EAX,DWORD PTR DS:[ESI+1C]
00451E7C  .  6A 00            PUSH 0                                lParam = 0
00451E7E  .  68 6F140000      PUSH 146F                             wParam = 146F
00451E83  .  68 11010000      PUSH 111                              Message = WM_COMMAND
00451E88  .  50               PUSH EAX                              hWnd = 1
00451E89  .  C686 F4010000 00 MOV BYTE PTR DS:[ESI+1F4],0
00451E90  .  FF15 80D75D00    CALL DWORD PTR DS:[<&USER32.PostMessageA>]  PostMessageA
00451E96  .  8B4D F4          MOV ECX,DWORD PTR SS:[EBP-C]
00451E99  .  64:890D 00000000 MOV DWORD PTR FS:[0],ECX
00451EA0  .  5F               POP EDI                               0012E5A0
00451EA1  .  5E               POP ESI                               0012E5A0
00451EA2  .  5B               POP EBX                               0012E5A0
00451EA3  .  8BE5             MOV ESP,EBP
00451EA5  .  5D               POP EBP                               0012E5A0
00451EA6  .  C2 0400          RETN 4
00451EA9  >  8B4E 1C          MOV ECX,DWORD PTR DS:[ESI+1C]         Case 1B of switch 00451A03
00451EAC  .  6A 1B            PUSH 1B                               TimerID = 1B (27.)
```

Hmm, and how did we come here? Look at 00451E6D: “Case 6F switch 00451A03”. So let’s once more search for the constants, but this time when you find the first hit, right-click and chose “find reference to immediate constant”. Look here:



Here we have many hits but we will only look into the “Push” 6F, so set a bp on all the “Push 6F”. Restart the program and run it. (Before running the program again, remember to apply the patch for the first server check). Bam Olly breaks.

```
o_O - main thr3ad, m0dule IDMan
00445E6C  .  50              PUSH EAX
00445E6D  .  E8 DB101500     CALL IDMan.00596F4D
00445E72  .  83C4 08         ADD ESP,8
00445E75  .  E8 E0101500     CALL IDMan.00596F5A
00445E7A  .  8945 EC         MOV [LOCAL.5],EAX
00445E7D  .  DB45 EC         FILD [LOCAL.5]
00445E80  .  DC0D E0845E00   FMUL QWORD PTR DS:[5E84E0]
00445E86  .  E8 0DF01400     CALL IDMan.00594E98
00445E8B  .  8B7D 08         MOV EDI,[ARG.1]
00445E8E  .  BE E8030000     MOV ESI,3E8
00445E93  .  8B1D 78D75D00   MOV EBX,DWORD PTR DS:[<&USER32.SetTimer>]   USER32.SetTimer
00445E99  .  2BF0            SUB ESI,EAX
00445E9B  .  8B4F 1C         MOV ECX,DWORD PTR DS:[EDI+1C]
00445E9E  .  6A 00           PUSH 0                                      Timerproc = NULL
00445EA0  .  8D86 204E0000   LEA EAX,DWORD PTR DS:[ESI+4E20]
00445EA6  .  C787 0C020000 B8 MOV DWORD PTR DS:[EDI+20C],88B8
00445EB0  .  50              PUSH EAX                                    Timeout = 0. ms
00445EB1  .  6A 01           PUSH 1                                      TimerID = 1
00445EB3  .  51              PUSH ECX                                    hWnd = 0012D6AC
00445EB4  .  FFD3            CALL EBX                                    SetTimer
00445EB6  .  8A87 F4010000   MOV AL,BYTE PTR DS:[EDI+1F4]
00445EBC  .  84C0            TEST AL,AL
00445EBE  .v 75 26           JNZ SHORT IDMan.00445EE6
00445EC0  .  8B57 1C         MOV EDX,DWORD PTR DS:[EDI+1C]
00445EC3  .  6A 00           PUSH 0                                      Timerproc = NULL
00445EC5  .  56              PUSH ESI                                    Timeout = 12475. ms
00445EC6  .  6A 6F           PUSH 6F                                     TimerID = 6F (111.)
00445EC8  .  52              PUSH EDX                                    hWnd = 0029060E ('Internet Download Manager 6.11',class='#32770
00445EC9  .  C687 F4010000 01 MOV BYTE PTR DS:[EDI+1F4],1
00445ED0  .  FFD3            CALL EBX                                    SetTimer
```

This could be the badboy . Can you see that SetTimer call? This is where it sets the timer for the bad boy, so we would like to avoid this. There are two SetTimer calls here, so it would be weird if it was the JNZ on 00445EBE between the SetTimer calls. We should change this because we don’t want any timer around the bad boy to be set, so let us scroll up a little.

```
00445E5C  >  895D E0         MOV [LOCAL.8],EBX                           USER32.SetTimer
00445E5F  .v EB 04           JMP SHORT IDMan.00445E65
00445E61  >  85C0            TEST EAX,EAX
00445E63  .v 74 6D           JE SHORT IDMan.00445ED2
00445E65  >  6A 00           PUSH 0
00445E67  .  E8 97021500     CALL IDMan.00596103
00445E6C  .  50              PUSH EAX
00445E6D  .  E8 DB101500     CALL IDMan.00596F4D
00445E72  .  83C4 08         ADD ESP,8
00445E75  .  E8 E0101500     CALL IDMan.00596F5A
00445E7A  .  8945 EC         MOV [LOCAL.5],EAX
00445E7D  .  DB45 EC         FILD [LOCAL.5]
00445E80  .  DC0D E0845E00   FMUL QWORD PTR DS:[5E84E0]
00445E86  .  E8 0DF01400     CALL IDMan.00594E98
00445E8B  .  8B7D 08         MOV EDI,[ARG.1]
00445E8E  .  BE E8030000     MOV ESI,3E8
00445E93  .  8B1D 78D75D00   MOV EBX,DWORD PTR DS:[<&USER32.SetTimer>]   USER32.SetTimer
00445E99  .  2BF0            SUB ESI,EAX
00445E9B  .  8B4F 1C         MOV ECX,DWORD PTR DS:[EDI+1C]
00445E9E  .  6A 00           PUSH 0                                      Timerproc = NULL
00445EA0  .  8D86 204E0000   LEA EAX,DWORD PTR DS:[ESI+4E20]
00445EA6  .  C787 0C020000 B8 MOV DWORD PTR DS:[EDI+20C],88B8
00445EB0  .  50              PUSH EAX                                    Timeout = 0. ms
00445EB1  .  6A 01           PUSH 1                                      TimerID = 1
00445EB3  .  51              PUSH ECX                                    hWnd = 0012D6AC
00445EB4  .  FFD3            CALL EBX                                    SetTimer
00445EB6  .  8A87 F4010000   MOV AL,BYTE PTR DS:[EDI+1F4]
00445EBC  .  84C0            TEST AL,AL
00445EBE  .v 75 26           JNZ SHORT IDMan.00445EE6
00445EC0  .  8B57 1C         MOV EDX,DWORD PTR DS:[EDI+1C]
```

Here at 00455E63 is the first JE we come to, before any of the SetTimer Calls, and if you follow it, you’ll see it jumps to 00445ED2 right after the SetTimer Calls! It must be it.
You could place a bp on this JNZ at 00445E63 and restart Olly, but then you will be disappointed because Olly doesn’t break there. That is because Olly will jump at 00445E5F, right over our JE at 00445E63. *(I was not sure when I ran the target the first time so I just placed a bp oat 00445E5F, 00445E61 and 00445E63 to be sure to break.)* Place a bp oat the JMP 00445E5F and restart Olly

```
00445E5F  .v EB 04           JMP SHORT IDMan.00445E65
00445E61  >  85C0            TEST EAX,EAX
00445E63  .v 74 6D           JE SHORT IDMan.00445ED2
00445E65  >  6A 00           PUSH 0
00445E67  .  E8 97021500     CALL IDMan.00596103
00445E6C  .  50              PUSH EAX
00445E6D  .  E8 DB101500     CALL IDMan.00596F4D
00445E72  .  83C4 08         ADD ESP,8
00445E75  .  E8 E0101500     CALL IDMan.00596F5A
00445E7A  .  8945 EC         MOV [LOCAL.5],EAX
00445E7D  .  DB45 EC         FILD [LOCAL.5]
00445E80  .  DC0D E0845E00   FMUL QWORD PTR DS:[5E84E0]
00445E86  .  E8 0DF01400     CALL IDMan.00594E98
00445E8B  .  8B7D 08         MOV EDI,[ARG.1]
```

The Final Patch

We know it should not jump here, so we could just NOP this JMP at 00445E5F and make the JE at 00445E63 to a JMP. Then the program will always jump over this SetTimer call for the last bad boy.

(There are lots of ways to do it. Again you could also just make the Command "JMP Short 00445E65" to "JMP Short 00445ED2", but it is a little more risky.)

Save the first and second server check patch to the file and you did it!!

So all in all we made these patches:

Serial patch	at: 004E3E76 Change "JNZ SHORT 004E3E83" to "JMP SHORT 004E3E99"
Sever Check patch	at: 0043DDB9 Change " MOV EAX,3" to "MOV EAX,0"
Sever Check Patch 2	at: 00445E5F "JMP SHORT 00445E65" to "JMP SHORT 00445ED2"

Final Notes

Yeah that is it, you can run the program without the nags from the server check and you can still check for updates. Many server check are bypassed by changing the Http:// address they are pointing at, but then you normally can't update. With this approach you can do it all.

For questions and discussion regarding this tutorial and other interesting stuff go to R4ndom's forum:

http://www.thelegendofrandom.com/forum/index.php

Thanks for now..

-XOR06

Thanks to:

Random for showing me the light again
NwoKiller for all his help
DMichael for sharing his wisdom
The **TPoDT** team

Without these guys this wouldn't be possible.