The difficulty contrasts of serial fishing

potassium of ARTeam

Version 1.0 – 12th March 2006

Keywords

serial fishing, lucky, easy

IMPORTANT NOTE (READ THIS):

In contrast with other groups/individuals on the scene, neither the author of this document nor the organization named ARTeam distributes patches, cracked binaries or serial/activation codes. Besides contributing to the team with your own personal knowledge, this is the main criterion for remaining a member of ARTeam. Software developers; (yes, I'm quite sure you are reading these tutorials as well) please do not see this article as a threat to your organization or as a loss of income, but as a possibility for you to better yourselves and your products.

Yours truly, potassium / ARTeam 2006

1. Abstract

Serial fishing is another aspect of the art of reverse engineering. The main goal of serial fishing might be to obtain a valid serial or gain knowledge of how a specific program generates its registration key from different variables like phone numbers, e-mail, name or information collected from hardware (eg. the hard-drive serial number). This tutorial contains two examples, one, which is extremely easy to fish and one that requires a bit more patience (just a little bit though). Even though the two examples are quite easy to fish, they clearly illustrate the variance of difficulty between different applications. Sometimes it's easy; sometimes it's not that easy and occasionally it's real nasty. ^__^

2. The first, ridiculously easy target

Ok.. Applications from Alive Media (www.alivemedia.net) are all extremely easy to fish for serials, so be very alert! This will all be over in a few seconds of debugging. The first example in this tutorial covers the program Alive DVD Ripper v1.3.2.8, which will be available mirrored on our site (http://root.accessroot.com/tools/alvdrip1328.rar). Okay, lets get started! Grab your cup of coffee, smokes, favourite blend of scotch or what ever makes you feel comfortable and launch Olly, open and run the target application. Then you will see this splash screen:

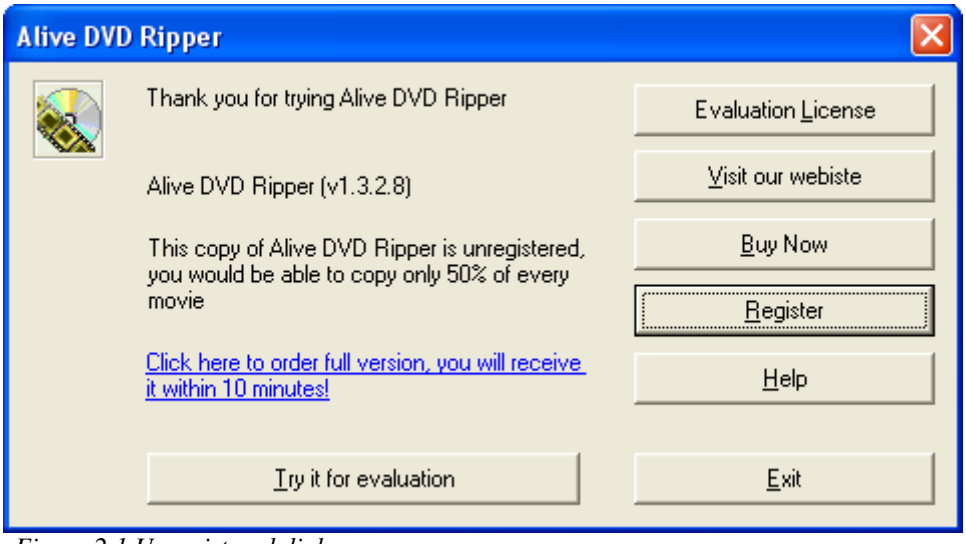

Figure 2.1 Unregistered dialog

Next step is easy, press the 'Register' button and this buddy comes along.

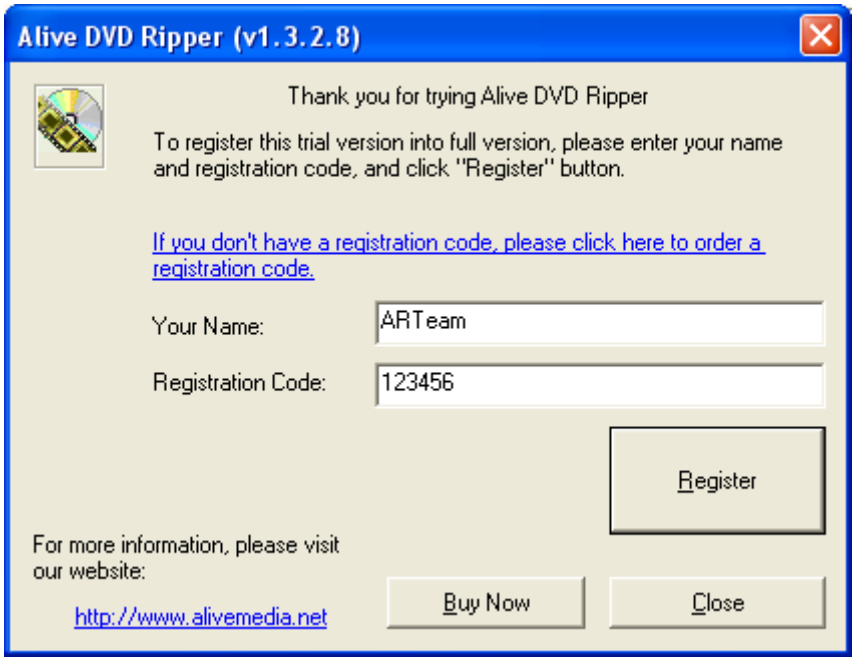

Figure 2.2 Registering dialog

As you can see in figure 2.2, I have entered some nice letters and figures in the registration dialog. Now, put a breakpoint on MessageBoxA. Press the 'Register' button and.. Wham-O!

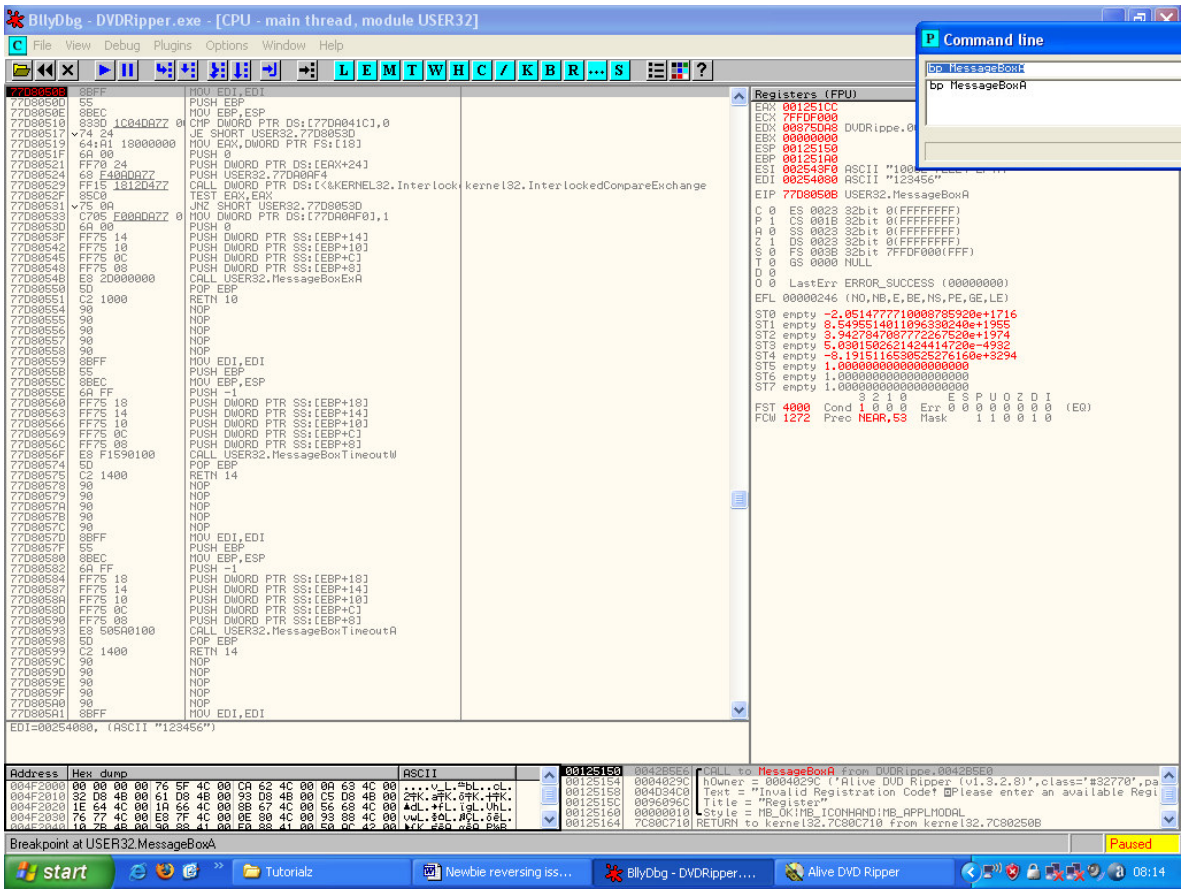

Figure 2.3 Olly and Alive DVD Ripper

What's that in ESI and EDI registers?! Woow, indeed it is our real serial (intentionally blocked by the command line window) in ESI and our input serial in EDI!

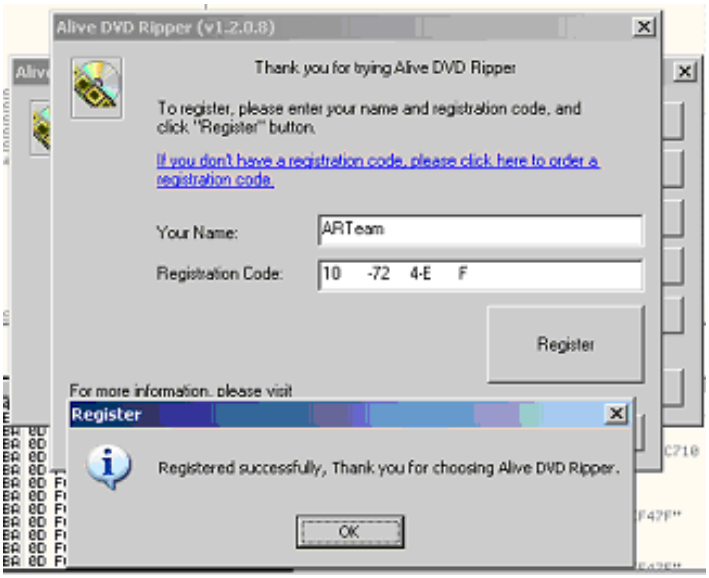

Figure 2.4 Registered!

Just copy and paste into the registration dialog and you are good to go. Couldn't be any easier.

Ok, that was way too easy!! Lets try something different.

3. The second target

The second example is serial fishing the application MemoriesOnTV v2.2.1 from www.codejam.com, which is packed with UPX. One option could be to unpack it and patch it and another option is to fish for a serial. Since this tutorial concentrates on serial fishing that is exactly what we will do. Get the application here : http://root.accessroot.com/tools/motv221.rar and lets go!

Fire up Olly and start the application by pressing F9. Now, press the register button.
And this dialog pops up. Enter a valid e-mail and a any serial.

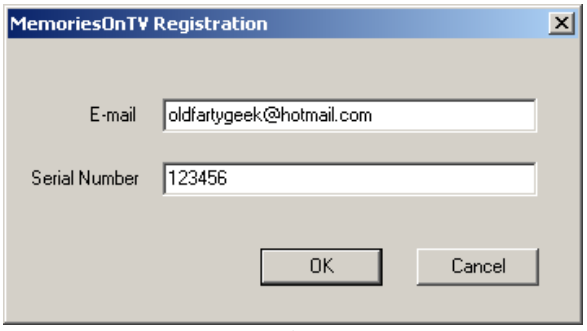

Figure 3.1 Registration box

Before you press the 'OK' button, be sure to put a breakpoint on MessageBoxA.
Olly will now break, press ctrl+F9 and this messagebox will pop up.

Figure 3.2 Invalid serial nag

Sureprise!!? No.. not really ;) Now, press the 'OK' button and land with Olly here:

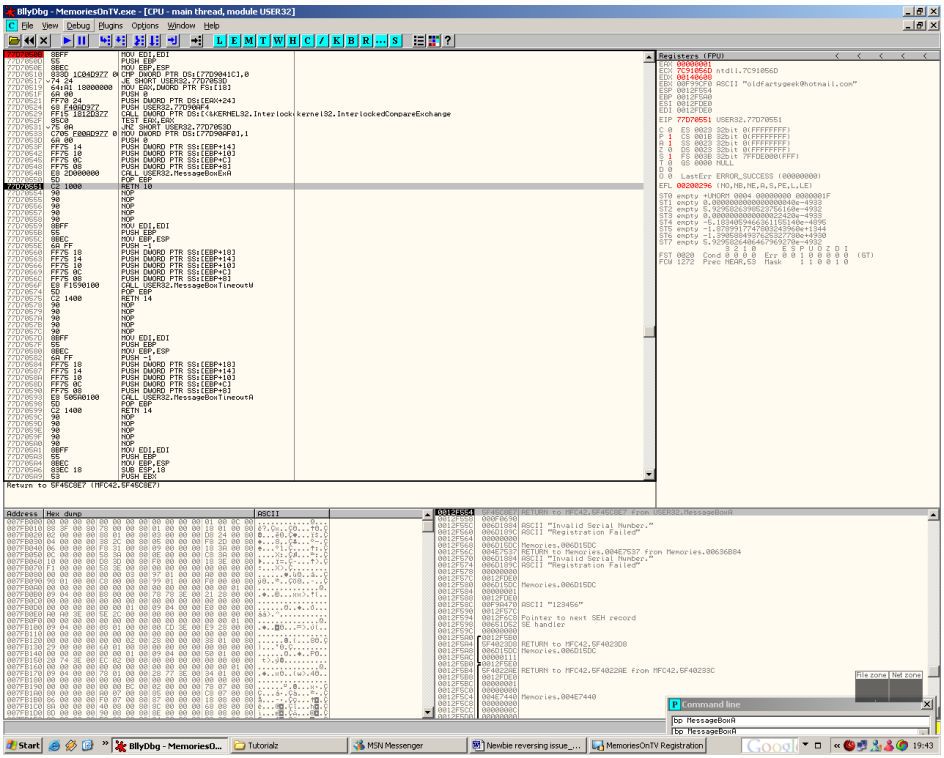

Figure 3.3 MemoriesOnTV and Olly

Execute that RETN and you will land in the mfc42.dll very close to another ret. Execute that RETN as well and you are back in the main application, here:

```
004E751C   E8 15F61400      CALL Memories.00636B36                  JMP to OFFSET MFC42.#4853_?OnOK@CDialog@@MAEXXZ
004E7521 ~EB 14             JMP SHORT Memories.004E7537
004E7523   8B4D E8           MOV ECX,DWORD PTR SS:[EBP-18]
004E7526   6A 00             PUSH 0
004E7528   68 9C186D00       PUSH Memories.006D189C                  ASCII "Registration Failed"
004E752D   68 84186D00       PUSH Memories.006D1884                  ASCII "Invalid Serial Number."
004E7532   E8 4DF61400       CALL Memories.00636B84                  JMP to OFFSET MFC42.#4224_?MessageBoxA@CWnd@@QAEHPBD0I
004E7537   C745 FC FFFFFFFF MOV DWORD PTR SS:[EBP-4],-1
004E753E   8D4D EC           LEA ECX,DWORD PTR SS:[EBP-14]
004E7541   E8 9CEF1400       CALL Memories.006364E2                  JMP to OFFSET MFC42.#800_??1CString@@QAE@XZ
004E7546   8B7D DC           MOV EDI,DWORD PTR SS:[EBP-24]
004E7549   8B75 E0           MOV ESI,DWORD PTR SS:[EBP-20]
004E754C   8B5D E4           MOV EBX,DWORD PTR SS:[EBP-1C]
004E754F   8B4D F4           MOV ECX,DWORD PTR SS:[EBP-C]
004E7552   64:890D 00000000  MOV DWORD PTR FS:[0],ECX
004E7559   8BE5              MOV ESP,EBP
004E755B   5D                POP EBP
004E755C   C3                RETN
```

Figure 3.4 Returned Invalid serial entered

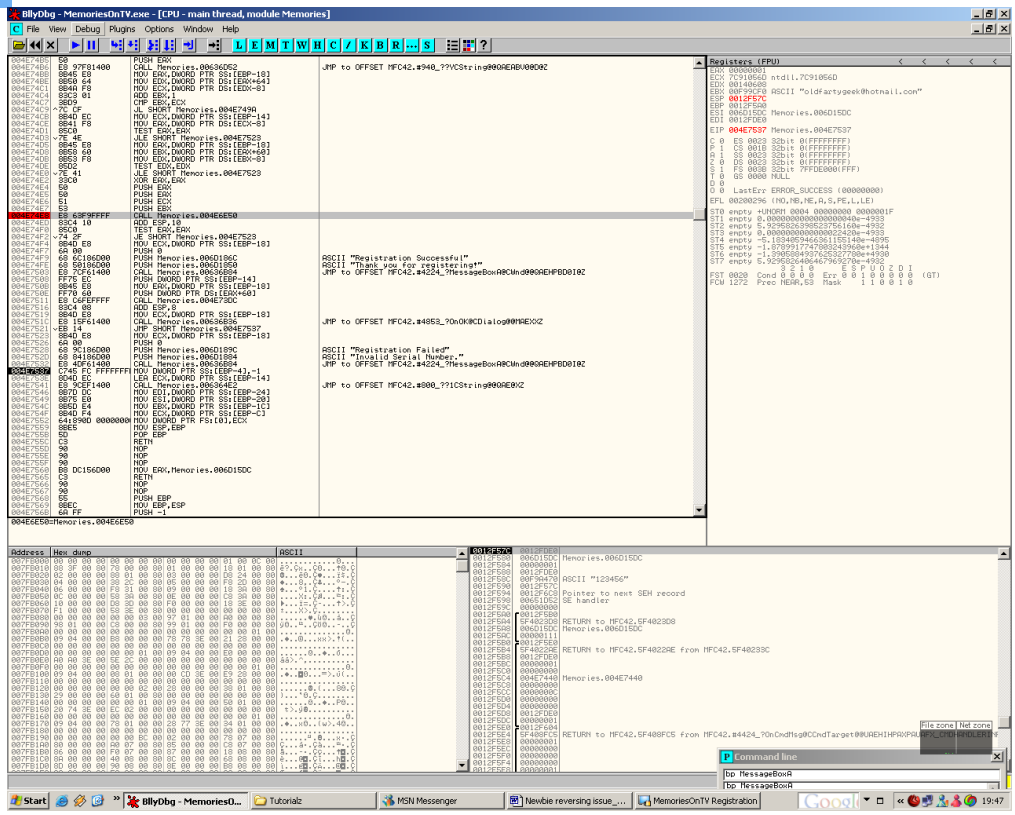

Figure 3.5 Locating the call to serial calculation and comparison

Scroll up a bit and you will see that call to 004E6E50 (Figure 3.5). Put breakpoint on that call and press F9 to try to register again! When Olly breaks at the call to 004E6E50 press F7 to step into the call. Then you will end up here:

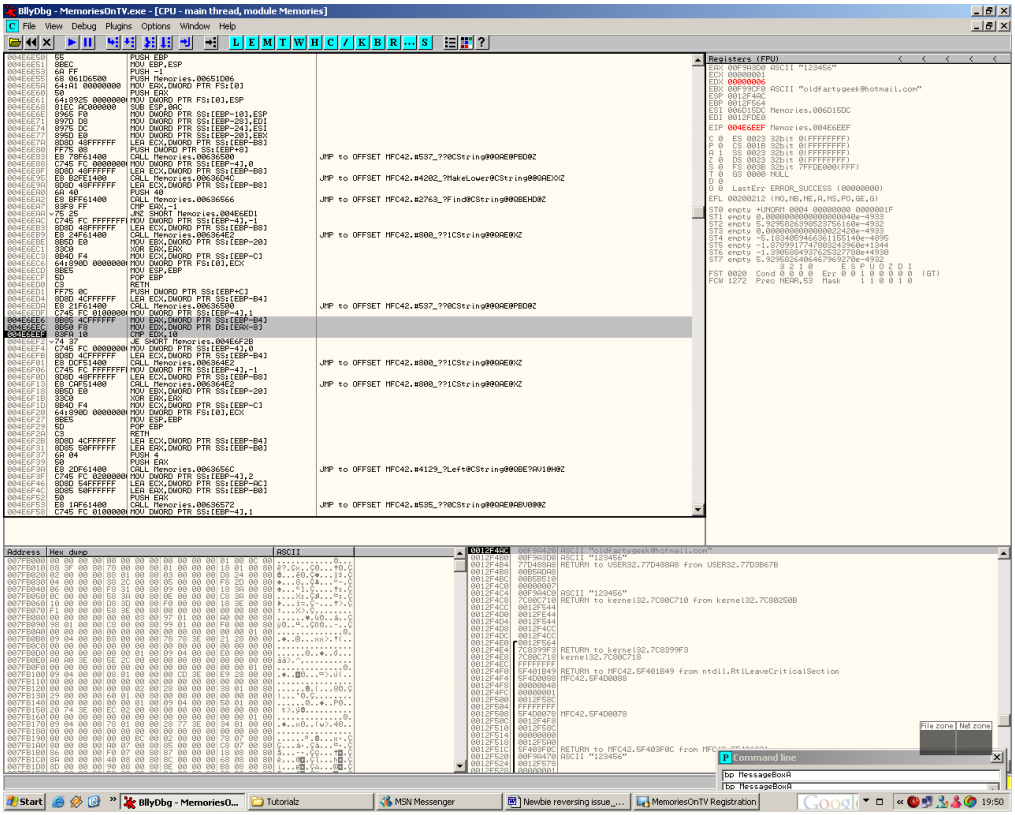

Figure 3.6 Serial length comparison

If you head on and trace to the gray-marked section our fake serial is stored in EAX and the length of it in EDX. Then, the bouncer (JE) says –"Ey pal, take a hike, circle the block a couple of times and come back later" and won't let EDX in because a length of 16 chars is required to enter. :) So, please stretch Mr. EDX a bit and make him the length of 16 chars. Now, try again. This time the bouncer will be satisfied and let us in. Then we get to this place:

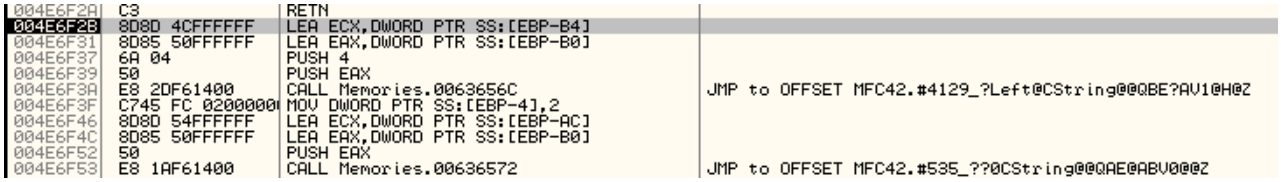

Figure 3.7 Serial length is OK

Now our real serial will be calculated and compared to what we've entered. Scroll on down through all the arithmetic calculations till you reach this place:

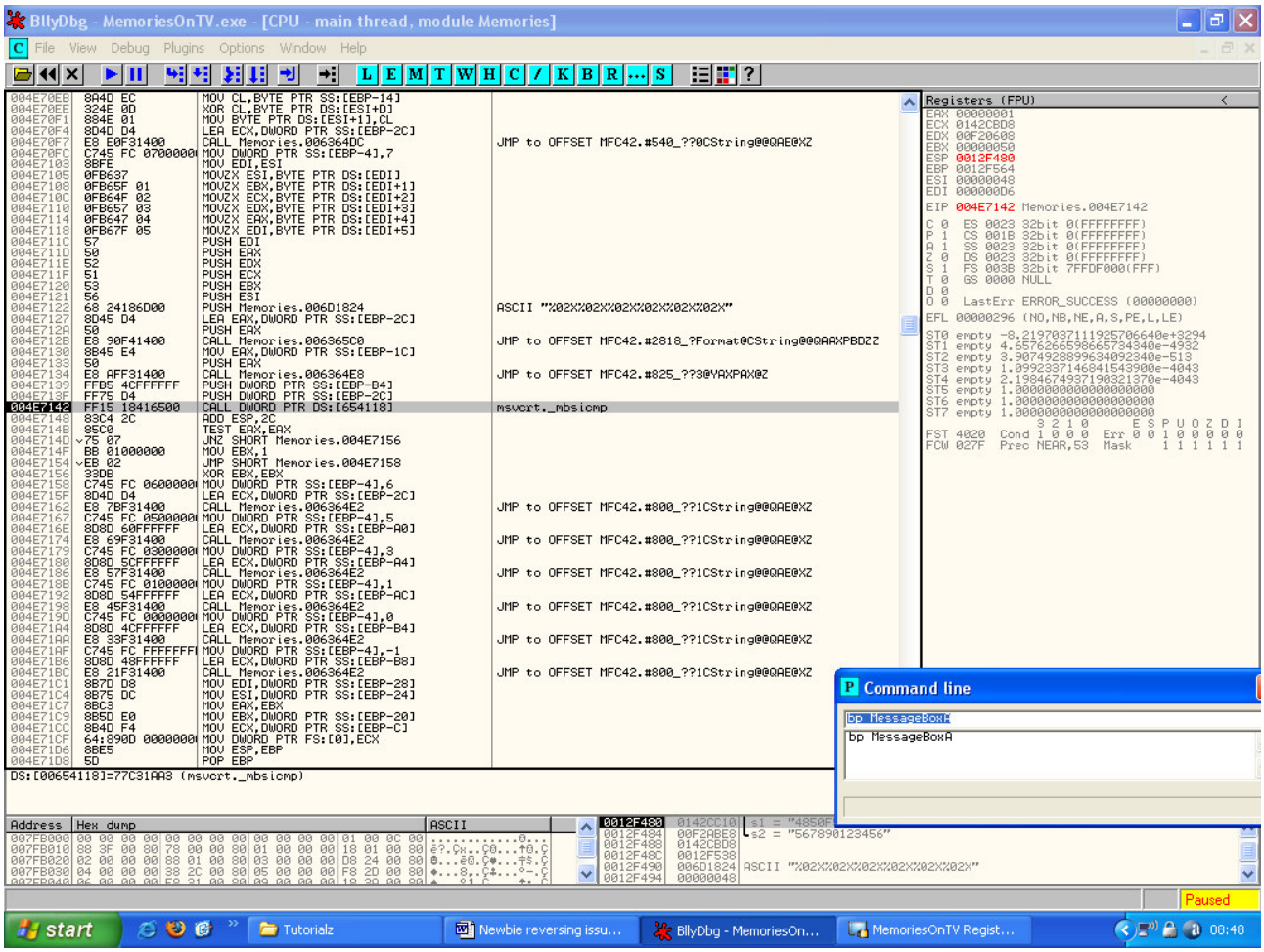

Figure 3.8 Real and fake serial comparison

Now both serials, real and fake, are located in the stack pane window of figure 3.8 (yes, the real serial is intentionally blocked). But, hey!! Where did my first 4 chars go? Well they have been cut by the serial generation routine. In fact all we need to do is add the first 4 chars that we entered in our fake serial to the real serial (most upper serial in stack pane) and we're good to go.

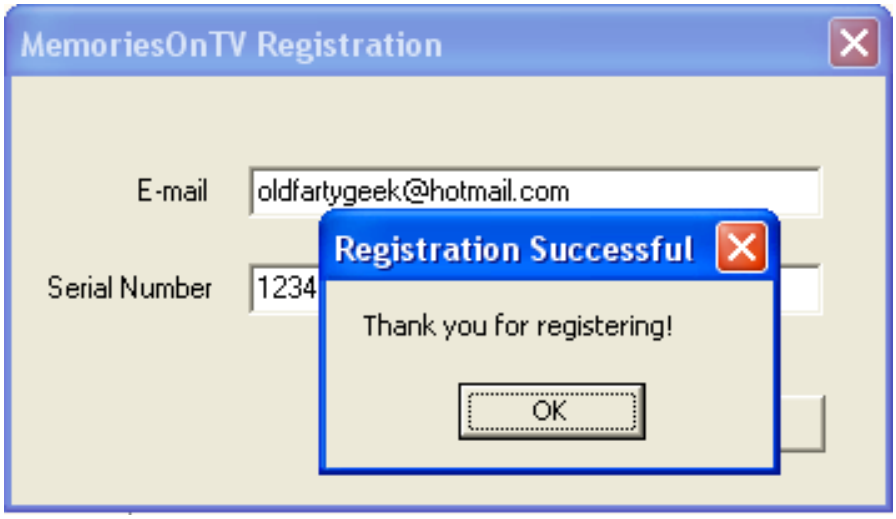

Figure 3.9 Yay! Registered!

Tataaa! *trumpets playing* , we're done, mission complete and all that.

4. Conclusions

Sometimes you're lucky, sometimes you're not so lucky, but this is where knowledge and patience comes into the picture. :) In the first example it was a piece of cake to find the correct serial since it is not cleared out from the stack after the "is-serial-correct-check". The difficulty in fishing serials also greatly depends on what language (asm, C, Delphi, etc) the application was constructed in, which external dependencies it has (mfc42, VB or other) and how much effort the programmers put in to confuse potential reverse engineers. In the second example you would need a little more patience and it also needs a bit more knowledge of program structure, compared to the first example.

5. Greetings

To the whole ARTeam, friend-teams out there and of course you who are reading this.