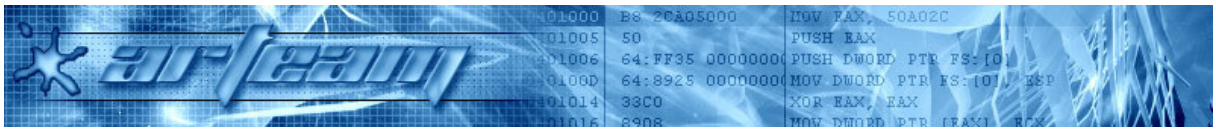

010Editor V2.0

http://www.sweetscape.com/

1. Target analysis

After install it use PeiD to know more about this target (if this is packed or about some crypto signature):

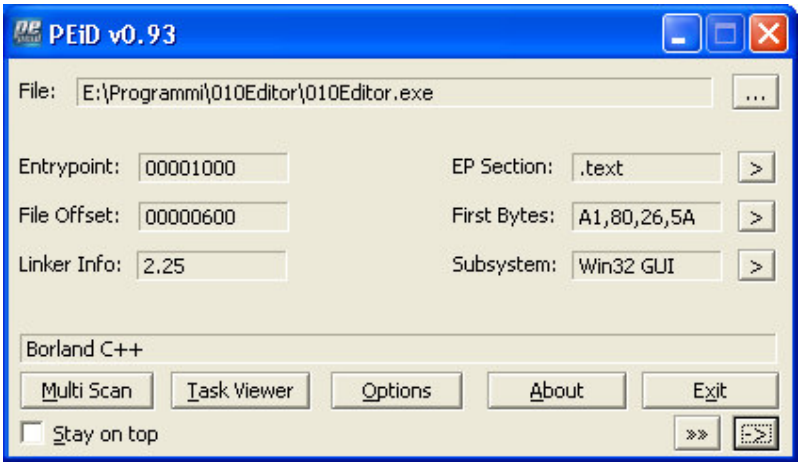

Fig. 1 File scanning about packer / compiler language.

Well target was not packed, another check can be made about some signature:

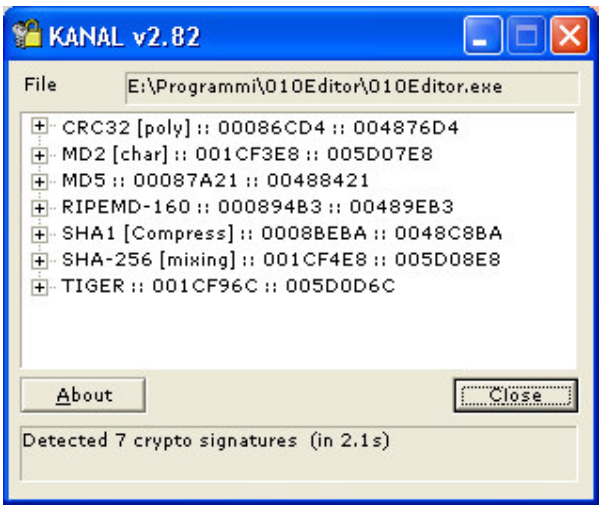

Fig. 2 Crypto signature.

Now is time to step into the code and patch it in order to keep full registration with a fake serial and show some simple searching technique useful to make the cracking stage.

2. Cracking stage

Load target in OllyDbg then right click and select Search for -> All referenced text strings:

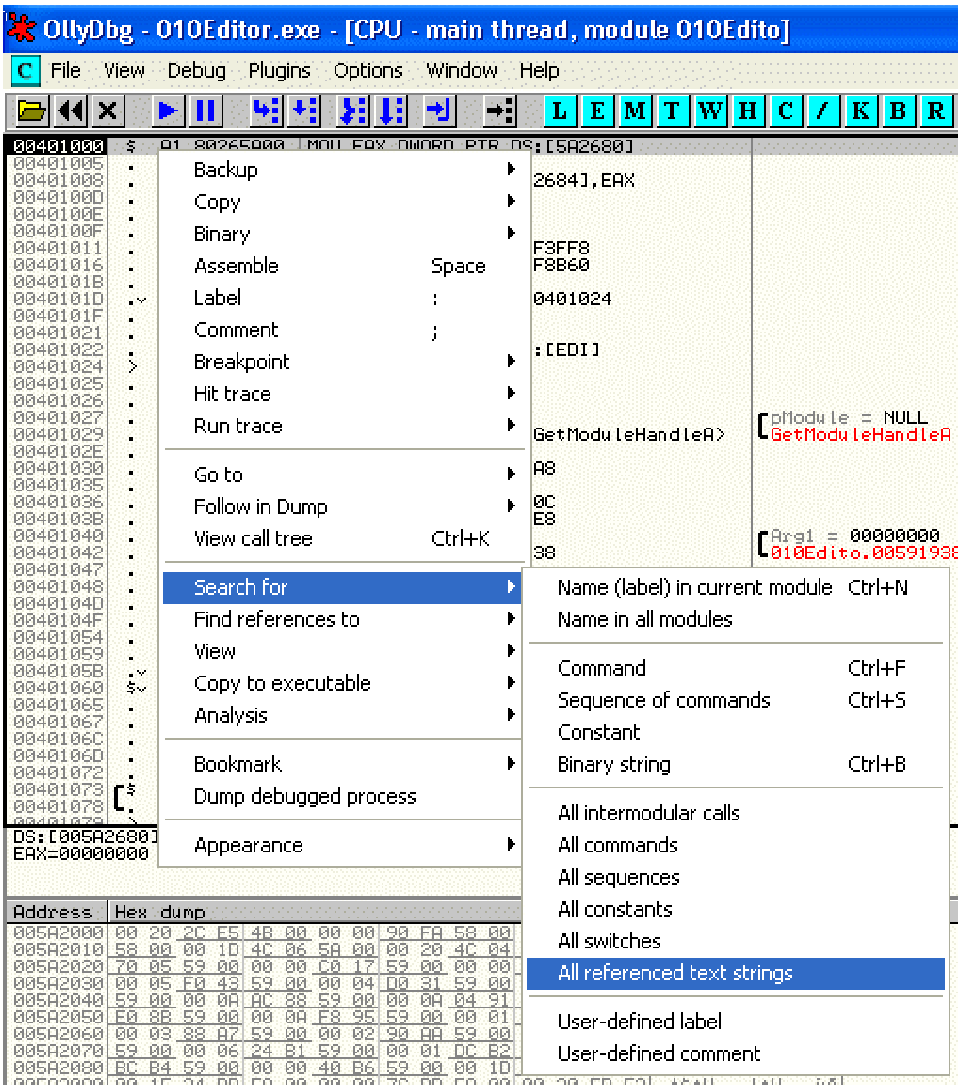

Fig. 3 String search.

Now scroll to the top and then right click to perform a text search, we have to know if there are some string referred to registration, typical string to search can be Thank, you, serial, registration or similar stuff, well start with Thank string.

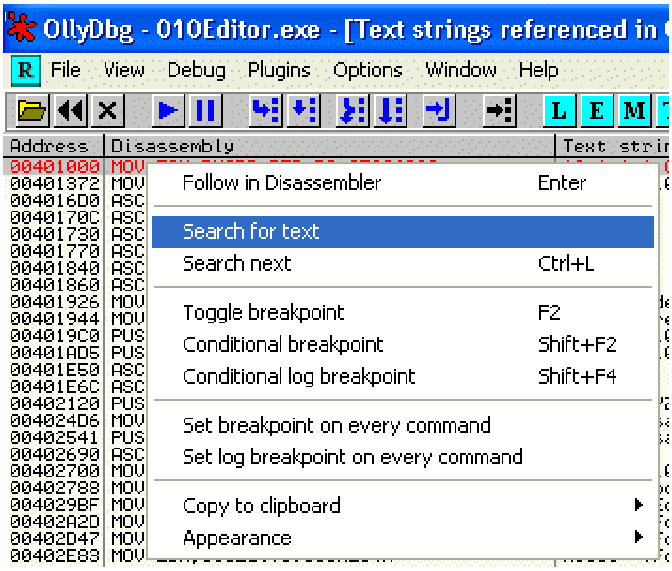

Fig. 4 Perform the text string.

Fig. 5 Write the text related to the string to search.

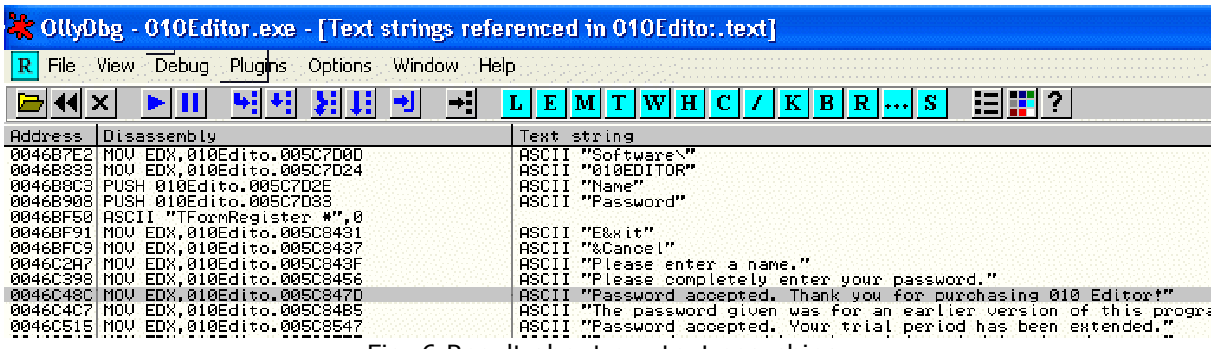

Fig. 6 Result about our text searching.

Double click on the highlighted string in order to show the related code,

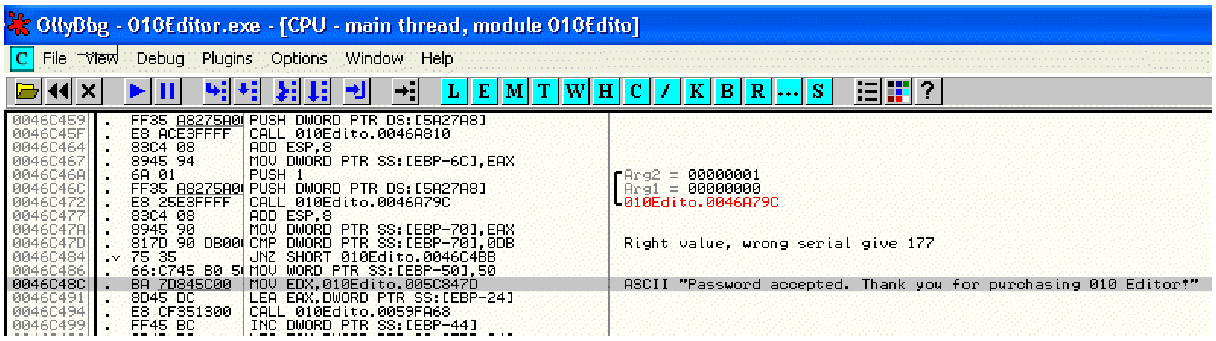

Fig. 7 Registration message.

As you can see, there is a conditional jump at address 0046C484, just above you find a CMP instruction, this jump is keep when EAX is different from 0DB (a wrong serial keep EAX equal to 0x177, to check it just make the registration stage with a fake serial). Our goal now is find where this value is computed (and then find the serial checking algorithm) to do this you've some option:

- trace into the code (using F7 and F8 and look for code)
- find a constant value searching for 0x177 (wrong serial) or search for 0xDB (right serial)

first option can be quite tedious but give better understand of the code (especially if you want to write a keygenerator or when serial fishing isn't possible), but for now we keep the last option, more faster and useful in some cases.

- ***Searching interesting point by constant value search***

  About this target you have to perform a constant value searching through the code, OllyDbg is able to do this search for you, just press right-click and choose Search For -> All constants:

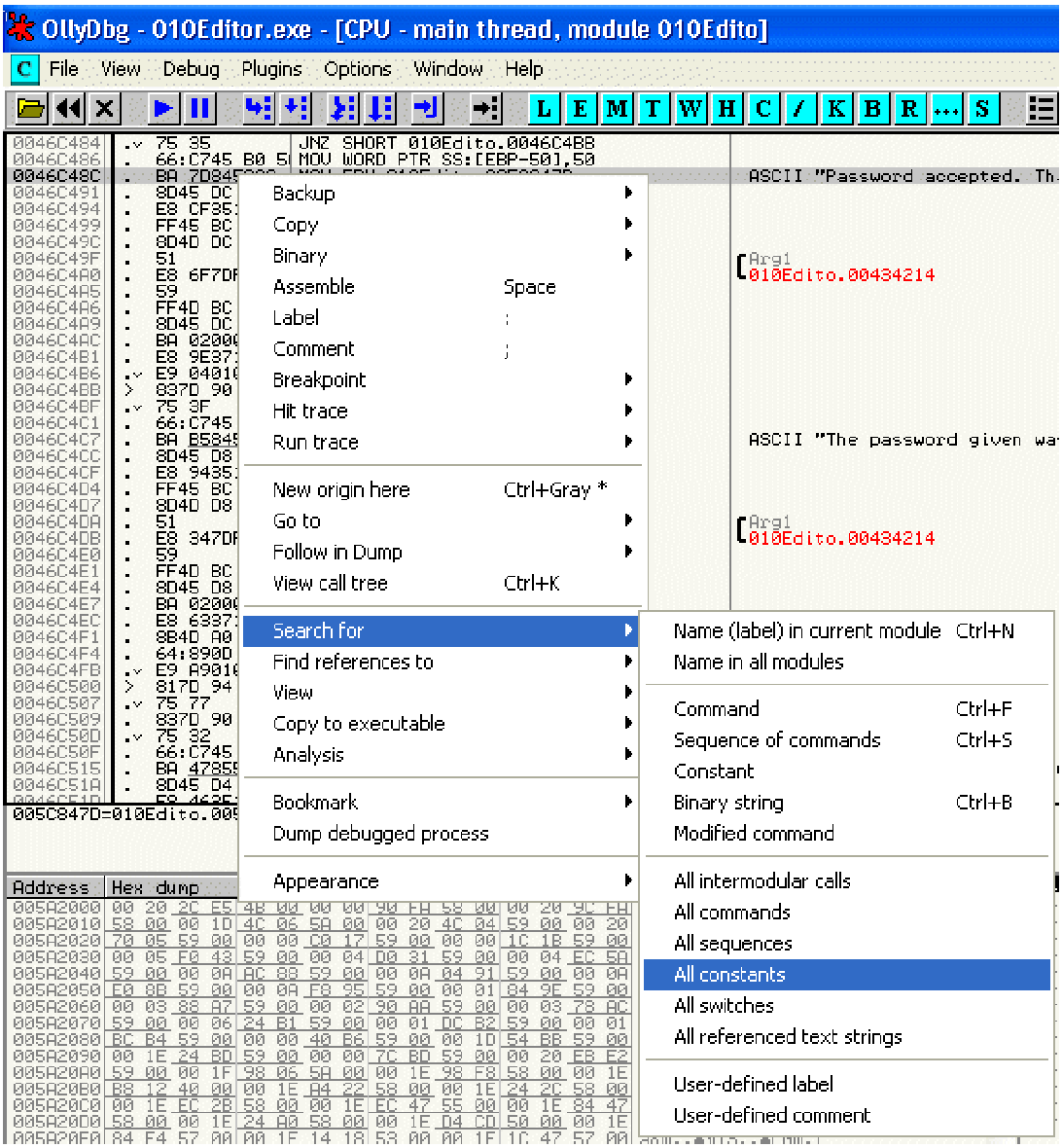

Fig. 8 Search for all constants.

Now just write your constant to search, 0x177 (bad registration value):

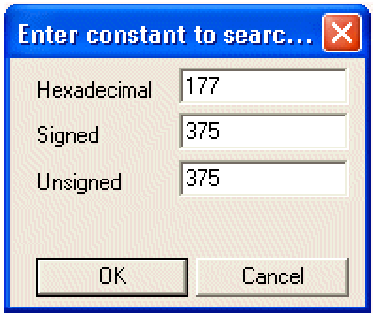

Fig. 9 Perform a search for 0x177 constant value.

Then press OK, OllyDbg perform this search and show all the entry founded:

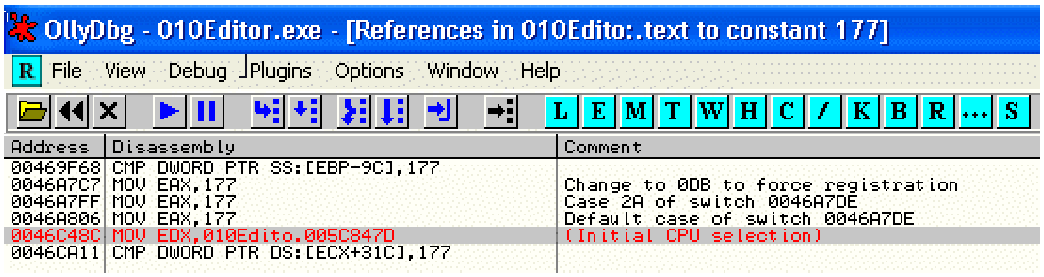

Fig. 10 Constant search.

Now you've to put a breakpoint in every entry in order to find the code executed when the serial checking will be made, to do this right-click and choose Set breakpoint on every command option:

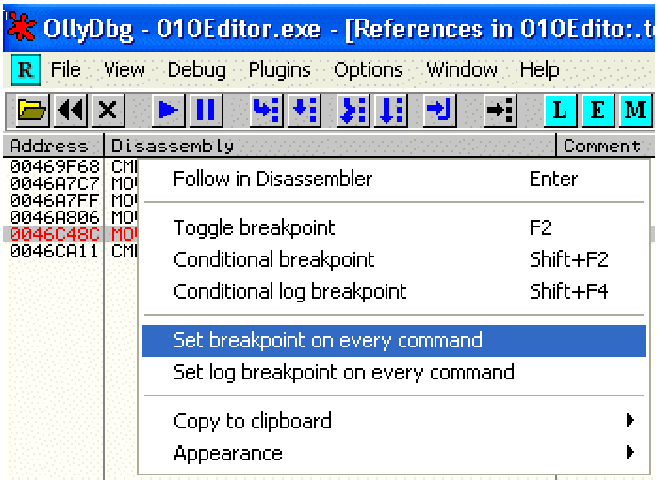

Fig. 11 Place a breakpoint in every entry.

Run the target (press F9) and enter a fake registration like this one:

Fig. 12 Enter your name and a fake serial.

Press OK, now OllyDbg break on 0046A7C7, the wrong registration value 0x177 is moved into the EAX register:

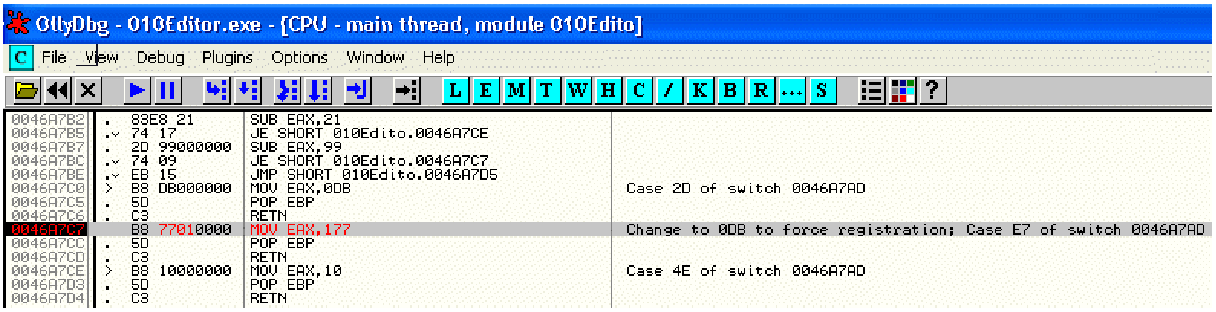

Fig. 13 Break

But this piece of code is also very interesting, look just some line above and you're able to see the 0xDB value related to the right registration, then our conclusion is simple, to force all serial entered to be valid just change 0x117 with 0xDB:

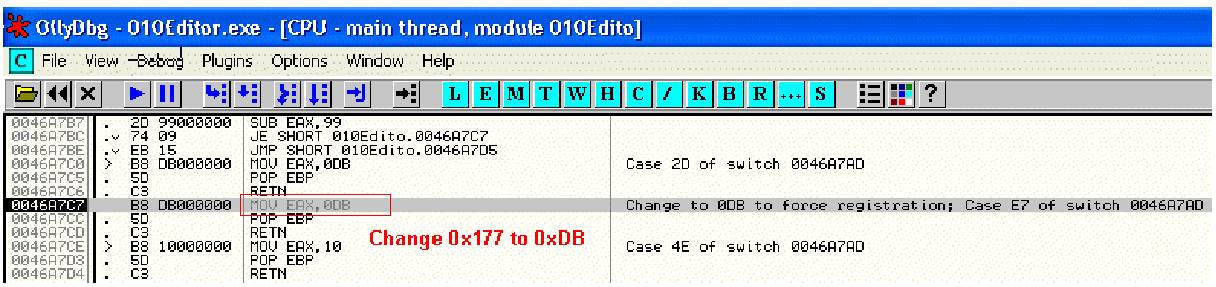

Fig. 13 Registration check patching

Now press F9 to run the target and see what happen:

Fig. 14 Succesfully registered!

Well we have keep the right registration message, now you've to check if target is really registered. This test is important in order to sure about full registration and you've, usually, to check again:

- the registration dialog box / the About dialog box / the main window caption
- restart the program and set your PC clock into the expiration period and look again for all the above mentioned window and caption in order to check the real full registration behaviour.

then first look at the register info dialog box:

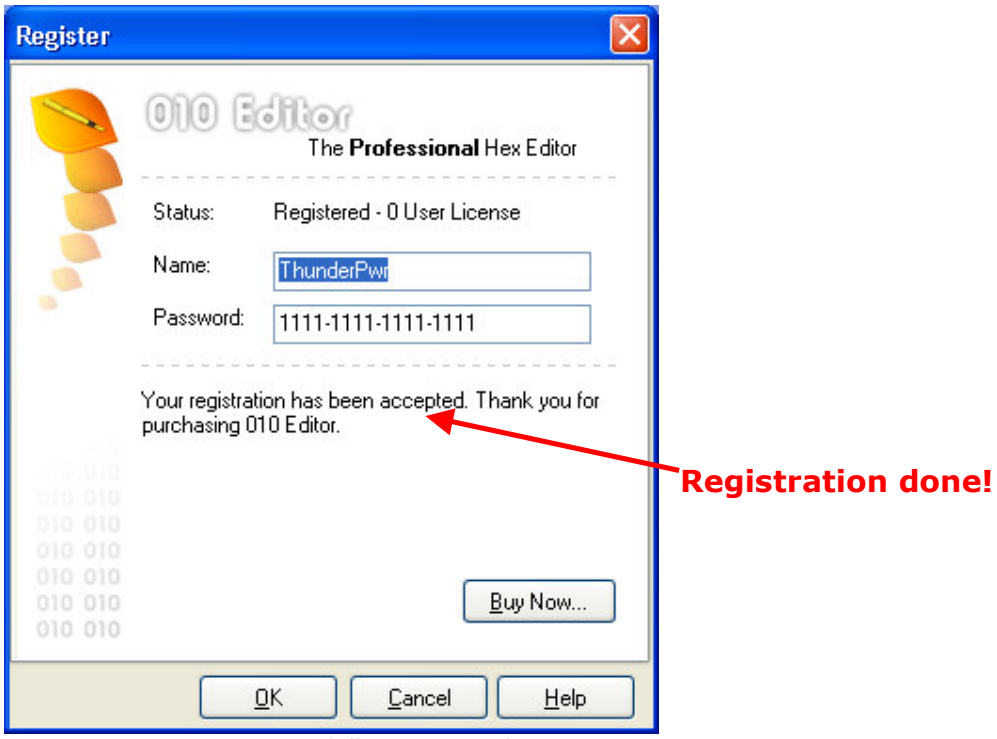

Fig. 15 Succesfully registered!

Also restart target after put the PC clock in order to keep the expiration time and see, our patch is working and program is really registered!

Fig. 16 Starting nag after registration.

Now we have to adjust the 0 user License to a Single User License message, this is no needed to keep full functionality in this target, but from more rigorous point of view this isn't correct, at least we should have 1 User License or some positive not zero number.

To keep the right message you've just to perform a search for text string "User License", after it you have to go to the 00469C7D address and some lines above change the DEC EAX instruction on 00469C74 with XOR EAX,EAX in order to force the Single User License message.

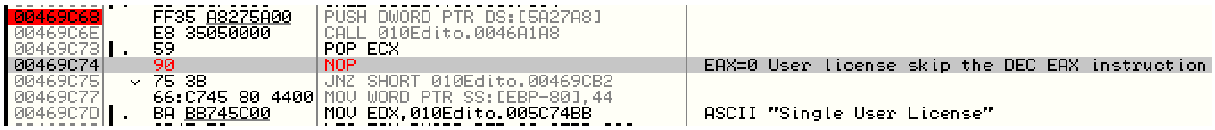

Fig. 17 Forcing the Single User License message on the start nag-screen.

Similar is about the registration dialog box:

Fig. 18 Forcing the Single User License message on the start nag-screen.

NOTE

In some case this type of patching can be detected by the target and some protection action can be taken, in that case simply check about your result and use a INC EAX or DEC EAX or some combination of this instruction in suitable way. We have to keep EAX=0, execution give just EAX=0 then you've to skip the DEC EAX instruction. This can be made in several way without using NOP instruction, one is change the JNZ to JZ or simply change the DEC EAX with DEC EDX, this is of the same size and don't affect the program flow because EDX is not involved in this piece of code.

Now the starting nag screen look as:

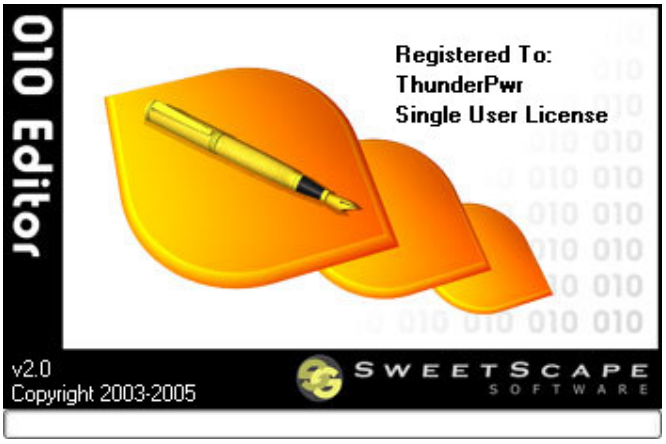

Fig. 19 Starting nag after patching

Well we have changed the text caption, but this is just a joke because really the total user license was keep to 0 into the program, this is really a program bug because other than the serial checking should be also check the total license number when this option is available (multiple license). Then our purpose now is discover where the total number of license is checked and then force it to the Single User License.

A first step can be made by looking the line above the Single User License message (refer to the Fig. 17 and 18), there is a CALL 0046A1A8 and also a PUSH DWORD PTR DS:[5A27A8] instruction, value inside this address is related to the total license number (you can proof it by checking the CALL 0046A1A8 code, there is just some instruction). To trace into the code restart the program (CTRL+F2) and place a breakpoint memory on write on 005A27A8.

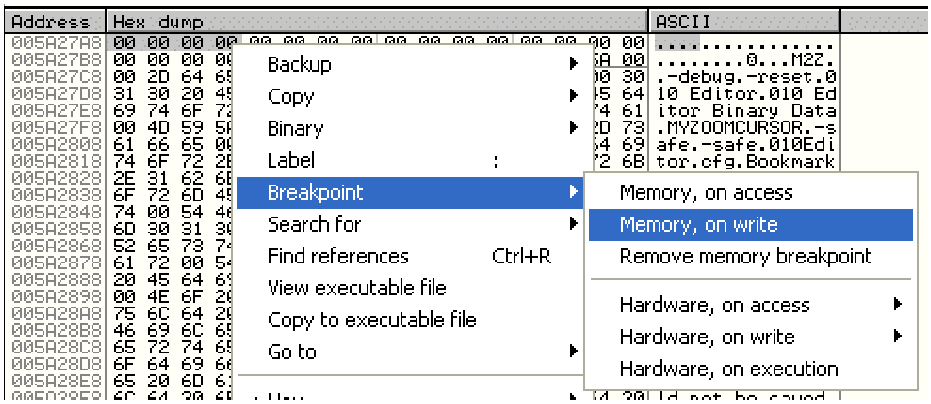

Fig. 20

Press F9 and OllyDbg break:

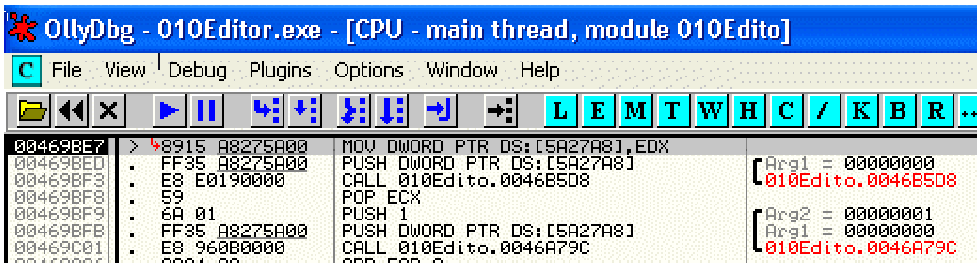

Fig. 21 Breakpoint on memory writing on 005A27A8.

NOTE
Another option can be perform a search for all constant 005A27A8 and place a breakpoint on each instruction, then run the target and wait for the first break.

Well, value which is written in our address is in my case (in your machine can be different value, but this is not important, just write down this value for future use) 00C769F8:

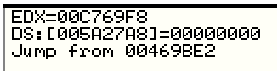

Fig. 22 Value written in 005A27A8.

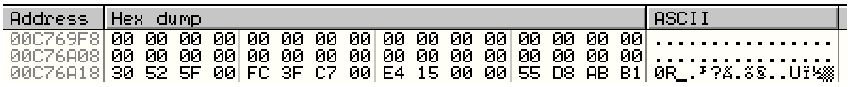

Fig. 23 Some address around 00C769F8.

Now we have to find when and where this address will be written by the total licensed user number available, again we have to use memory breakpoint, but now you have to place a memory breakpoint when writing to the **00C769F8+1C=00C76A14** address, because this address is related to the CALL 0046A1A8 code reported into the figure 24 and the returned value will be read from [EAX+1C], where EAX keep the address pushed into the stack just before this call (see also fig 17).

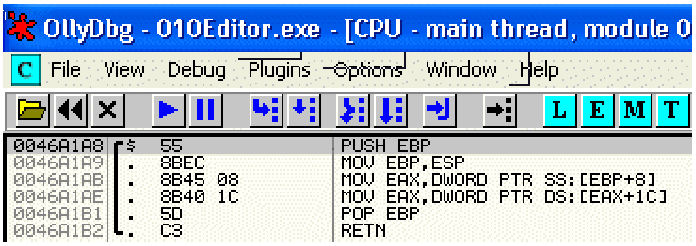

Fig. 24 Code at 0046A1A8.

Now you've some option, this code is called also from other part then is no good idea alter it to force in writing the 00C76A14 address then you've to apply the change before this calling.

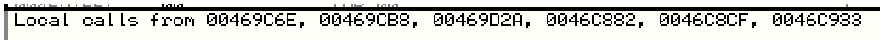

Fig. 25 Other place where this function is call.

There is only some entry, check about it give for some of this the well know structure as you can see in fig. 17 and 18, other place is referred to other option which is not involved when registration is taken from Single User License (in 00469CAD there is a jump which skip all the other call and again in 0046C8C4 there is another jump related to the registration dialog box which skip all the remaining call in 0046C8CF and 0046C933).

NOTE
For this target really is not necessary patch the 00C76A14 address, but is enough the simple patching shown into the figure 17 and figure 18, but to sure about your future crack for other target you've to keep in mind also this consideration, in some cases is enough patching the call result, but in other you've to find the source address and patch the value inside it because this address can be used in other place to check registration or other feature.

3. Conclusion

Simple search can be useful to perform more complex task like registration code searching and then know how the program make the registration step, in this target a more simple registration checking was made but concept can be generally extended to perform other searching e.g. expiration time searching (for 30 days trial expiration you can search for constant value equal to 0x1E and so on). Also we have talking about some commonly program bugs about registration checking.

Remember also the main things:

> If you like this software you have to BUY IT or simply uninstall it after expiration.
> This tutorial are used for educational purposes only.

4. Greetings

Thanks to the whole ARTeam:

[Nilrem] [JDog45] [Shub - Nigurrath] [MaDMAn_H3rCuL3s] [Ferrari] [Kruger] [Teerayoot] [R@dier] [ThunderPwr] [Eggi] [EJ12N] [Gabri3l][Stickman 373] [Bone Enterprise]

Thanks to all the people who take time to write tutorials.
Thanks to all the people who continue to develop better tools.
Thanks to Exetools, Woodmann, SND, TSRH, MP2K and all the others for being a great place of learning.
Thanks also to The Codebreakers Journal, and the Anticrack forum.

If you have any suggestions, comments or corrections contact me in usual places.

Byez
ThunderPwr