11.Intermediate patching using Olly's "pane window"

2012 년 1 월 29 일 일요일
오후 7:47

Hello everybody.
모두들 안녕.
Welcome to this Part 11 in my series about reversing for newbies/beginners.
나의 초보자 reversing series Part 11 에 온 것을 환영해.
This "saga" is intended for complete starters in reversing, also for those without any programming experience at all.
이 "saga"는 완벽히 reversing 초보자를 맞춰서 만들어졌다. 또한 어떠한 programming 경험이 없어도 된다.
Lena151 (2006)

Set your screen resolution to 1152*864 and press F11 to see the movie full screen !!!
Again, I have made this movie interactive.
You screen 해상도를 1152*864 로 설정해 그리고 full screen 으로 movie 를 보기 위해 F11 를 눌러
So, if you are a fast reader and you want to continue to the next screen, just click here on this invisible hotspot. You don't see it, but it IS there on text screens.
그래서, 네가 이것을 빨리 읽고 다음 screen 을 보고 싶다면, 보이는 hotspot 여기를 눌러. 보고 싶지 않을 때는 여기에 두지마.
Then the movie will skip the text and continue with the next screen.
Movie 는 text 와 다음 screen 을 skip 할 수 있다.
If something is not clear or goes too fast, you can always use the control buttons and the slider below on this screen.
무언가 명확하지 않거나 빨리 넘기고자 할 때, 항상 control button 과 이 screen 밑에 있는 slider 바를 사용해.
He, try it out and click on the hotspot to skip this text and to go to the next screen now!!!
도전해봐. 그리고 이 text 와 다음 screen 을 보기 위해 hotspot 을 click 해.
During the whole movie you can click this spot to leave immediately
이 movie 어디에서나 즉시 떠나기 위해 이 spot 을 click 할 수 있다.

1. Abstract

In this part 11, we will reverse a "real" application to learn something about the use of the "Pane window" in Olly.

이번 part 11 에서는, 우리는 Olly 에서 "Pane window"를 어떻게 사용하는지 배우기 위해 "real" application 을 reverse 할 것이다.
Whilst finding the path to the patches, we will be looking into another reversing in intermediate patching level.
경로에서 patch 를 찾는 동안, 우리는 intermediate patching level 에서 다른 reversing 을 볼 것이다.
This is because indeed, the best practice is found in real applications.
real application 에서 최고의 연습 방법을 찾았다.
For better comprehension and if you are a newbie, I advise you to first see the previous parts in this series before seeing this movie.
좀 더 이해력을 높이고 네가 초보자라면, 너는 이 movie 를 보기 전에 이 series 의 이전 part 를 먼저 보라고 권유한다.
The goal of this tutorial is to teach you something about a program's behaviour.
이 tutorial 의 목표는 너에게 program's behaviour 에 대하여 가르치기 위한 것이다.
In my search not to harm authors, I found an old version of Flash Jigsaw producer (version 2.0.0.22) which is no longer available for download.
나의 연구는 제작자에게 해가 되지 않는다. 나는 Flash Jigsaw producer 의 old version 을 찾았다. 그것은 더 이상 download 되지 않는다.
Taking a look in the specialized media, I also found this application to be "cracked" already.
특화된 media 를 봐. 나는 이미 "cracked" 된 application 을 찾았다.
Here, this applications is only chosen because it is ideal for this tutorial in reversing and it is targeted for educational purposes only.
여기, 이 applications 은 오직 선택됐다. 왜냐하면 이것은 reversing tutorial 에 이상적이다. 그리고 이 target 된 것은 오직 교육적인 목적이다.
I hope you will exploit your newly acquired knowledge in a positive way.
네가 얻은 새로운 지식을 긍정적인 방향으로 이용하길 바란다.
In this matter, I also want to refer to Part 1.
문제가 있으면, Part 1 를 참조하기 바란다.

2. Tools and Target

이것도 똑같음
The tools for today are : Ollydebug and... your brain.
오늘의 tools 은 : Ollydebug 와 너의 두뇌다.
The first can be obtained for free at
먼저 무료로 여기에서 얻을 수 있다.
http://www.ollydbg.de

Again, the brain is your responsibility ;)

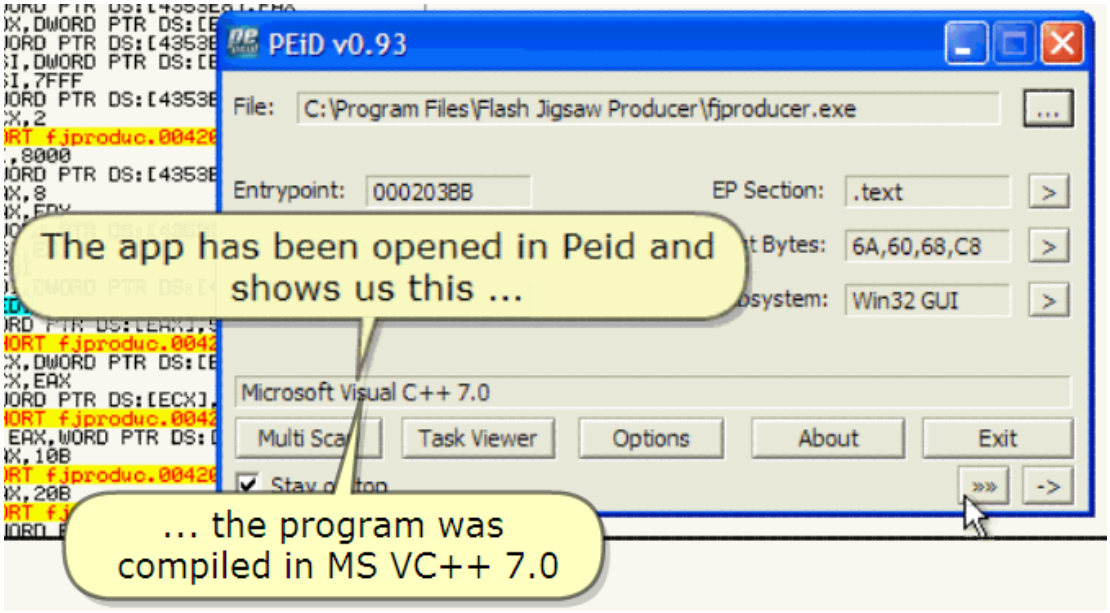
다시, 두뇌는 너의 책임이다 ;)
Todays target is a program called Flash Jigsaw Producer 2.0.0.22
오늘 target program 은 Flash Jigsaw Producer 2.0.0.22 로 불린다.
Because it can no longer be downloaded, I have included it in this package for research.
왜냐하면 이것은 더 이상 download 되지 않는다, 나는 이 package 에 연구용으로 첨부했다.

3. Behaviour of the program

Because you know meanwhile about the importance of decent study of your target and because you've seen how to do it in previous Parts in this series.
그 동안 너의 target 에 대한 적절한 공부는 중요했다. 그리고 너는 이 series 의 이전 parts 에서 어떻게 하는지 봤다.
I will only show you the things we will need to take care of.
오직 너에게 things 를 보여줄 것이다. 우리는 조심할 필요가 있다.
As you can see, I have already opened the target program in Olly and in PEiD
네가 볼 때, 나는 Olly 나 PEiD 에서 이미 target program 을 열었다.

REMARK :
This program uses a trick to detect the debugger when run without protection and will exit.
실행할 때는 보호되지 않고 종료될 수도 있다. 이 program 은 debugger 를 찾는 trick 을 사용한다.
It's not the goal of this Part to explain this feature now.
이제 이 Part 의 목표는 특징을 설명하는 것이 아니다.

Hence, download the HideDebug or similar plugin for now and extract into Olly's plugin directory.
그리하여, 지금 HideDebug 나 비슷한 plugin 을 download 받는다. 그리고 Olly's plugin directory 에 압축을 푼다.
I'll come back to this extensively in later Parts.
나는 광범위하게 다음 part 에서 돌아올 것이다.
http://www.tuts4you.com/download.php?view.57



The app has been opened in PEiD and shows us this ...

... the program was compiled in MS VC++ 7.0

이 app 은 PEiD 에서 열었다. 그리고 우리에게 이것을 보여준다.

Program 은 MS VC++ 7.0 으로 compile 됐다.

...and this

(Though this is not so important for us now. I will also come back to this in later Parts ...)

We can close PEiD now.

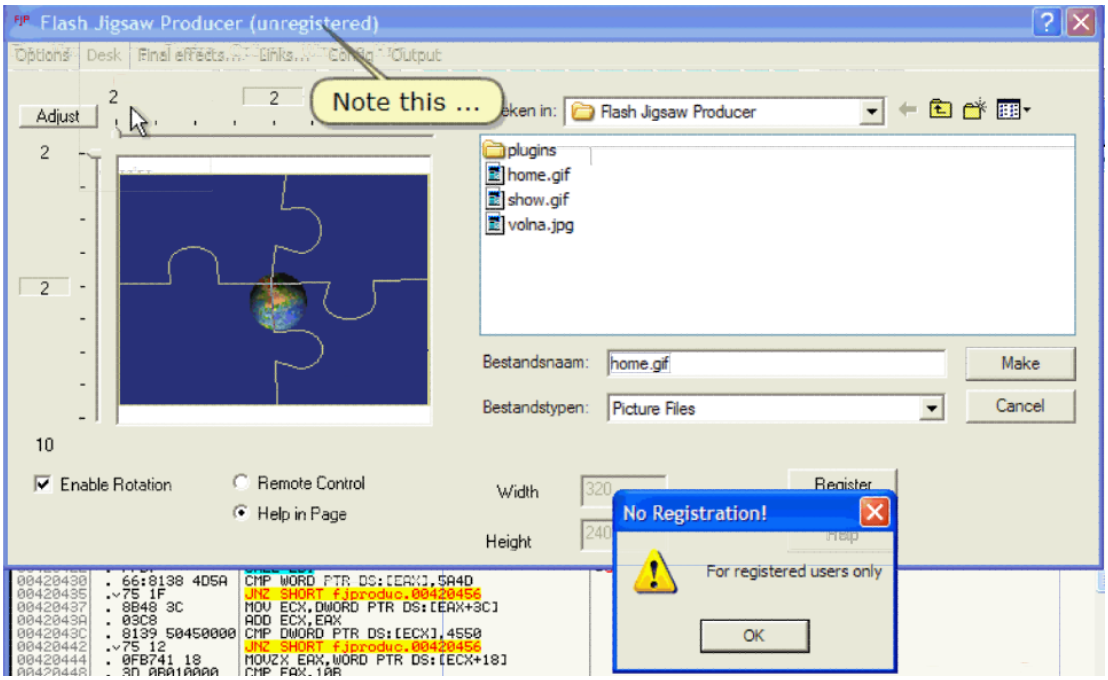
그리고 이거.

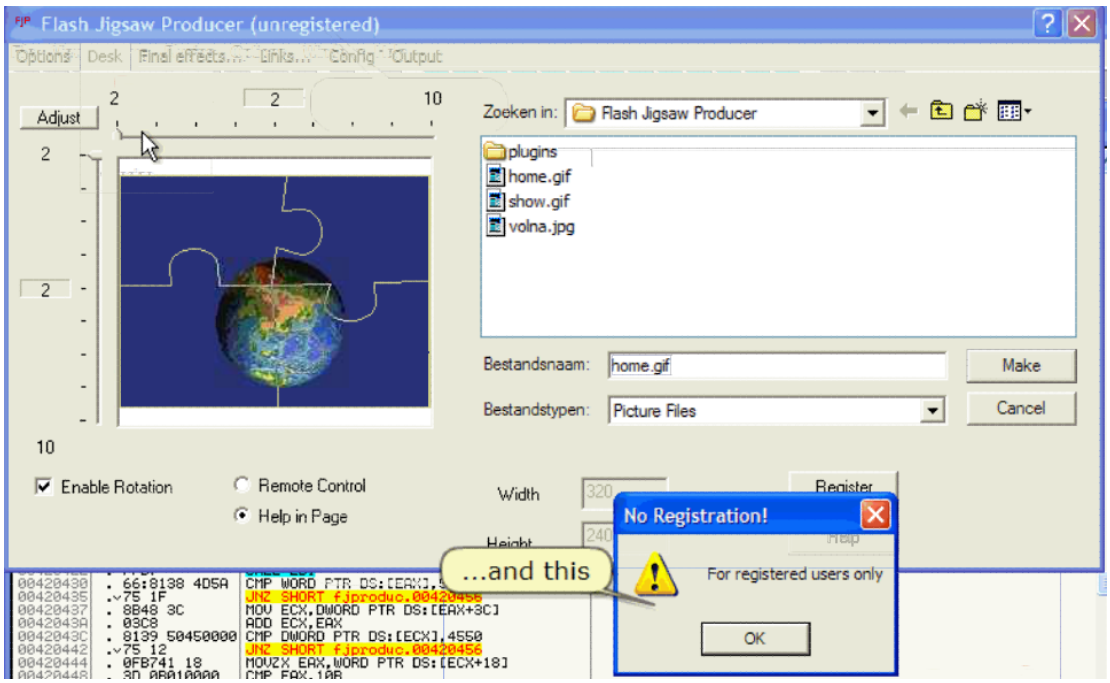
현재 이것은 우리에게 중요하지 않다. 나는 나중 part 에서 이것을 위해 돌아올 것이다.

우리는 이제 PEiD 를 닫을 수 있다.

Ok, always study the soft thoroughly first
I've done that already and will show you the limitation for non-registered
Ok, 항상 철저하고 가볍게 공부해 보자.
나는 이미 해냈다. 너에게 미등록을 위하여 제한을 보여주겠다.

So, run the soft
그래서, 가볍게 실행하자.





Note this ...

...and this
이것과 이것

Which tells us that the program is crippled
Program 은 훼손됐다고 말한다.

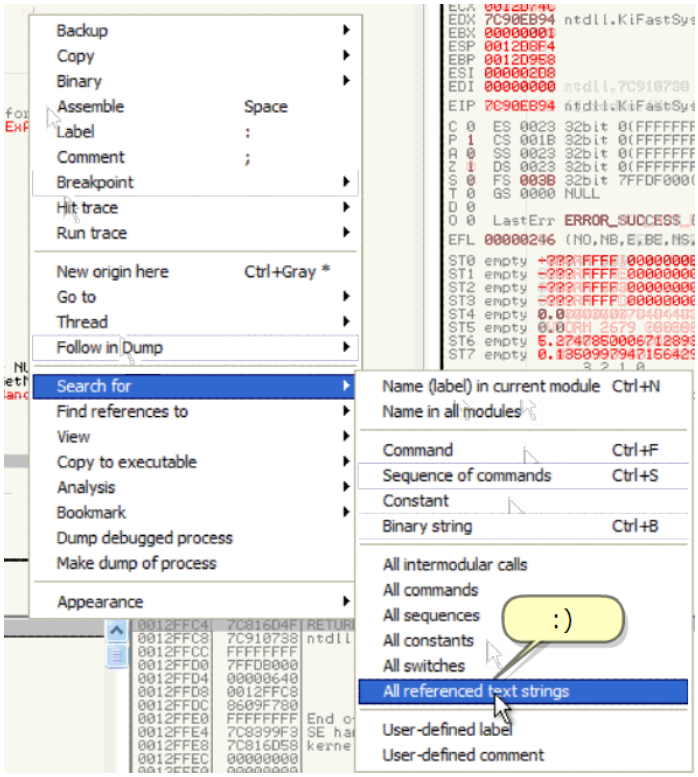
INFO :
I'm using "crippled" in the sense of "not all features available while unregistered".
"미등록 되었을 때 사용할 수 없는 모든 기능"에서 "제한" 되어 있다.
It is clear that there are also crippled programs that are never meant to be fully working.
이것은 그곳은 훼손된 program 이 있다는 것이 명확하다. 결코 충분히 일하는 것을 뜻하지 않는다.
These are also often called "demo" because this soft can only "show" some of the features whilst some of the (other) features are then simply not coded/implemented.
그것들은 자주 "demo"로 불린다. 왜냐하면 다른 특징들이 간단히 암호화/실행 될 때 이 soft 는 오직 약간의 특징들을 보여준다.

Ok. We have all the information we need. So, off to ...
Ok. 그래서 우리가 필요로 하는 정보를 모든 정보를 가지고 있다.

4. Finding the patches

Now let's return to Olly (click in Olly)
이제 Olly 로 돌아가자 (Olly 에서 click 해)

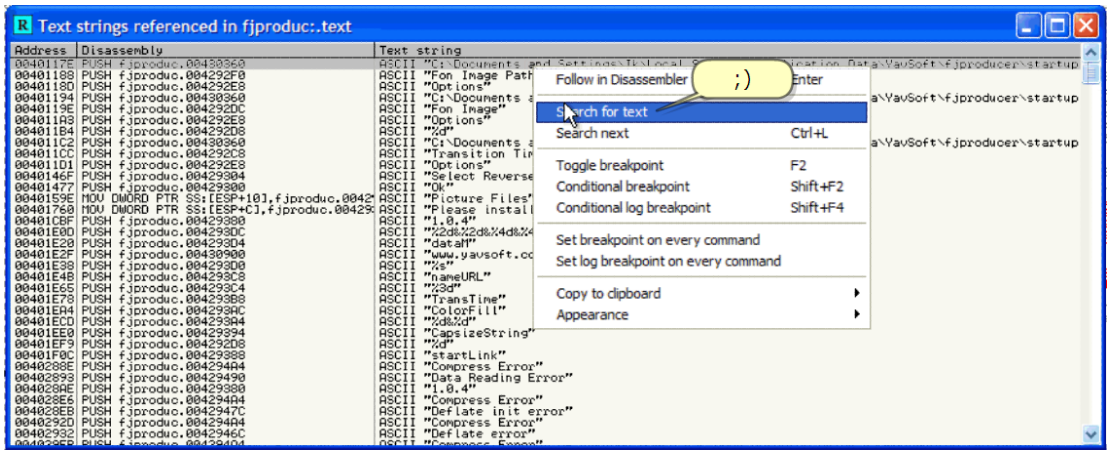
I'm sure you already know what I'll be looking for ...
내가 찾고자 하는 것을 네가 이미 알고 있다.



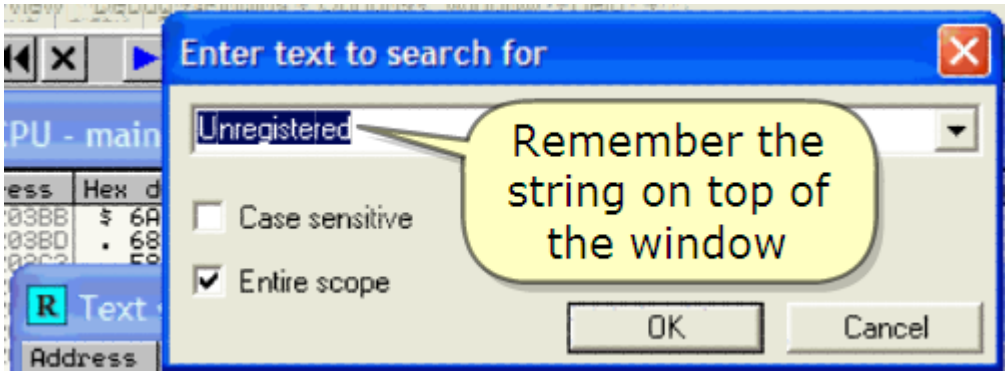
:)

The "newbie way" to find what we want. Better and more advanced ways will be explained later. Stay turned....

"초보자 방법" 우리가 원하는 것을 찾으려고 한다. 좋고 좀 더 진화된 방법은 나중에 설명하겠다. 켜져 있어요....

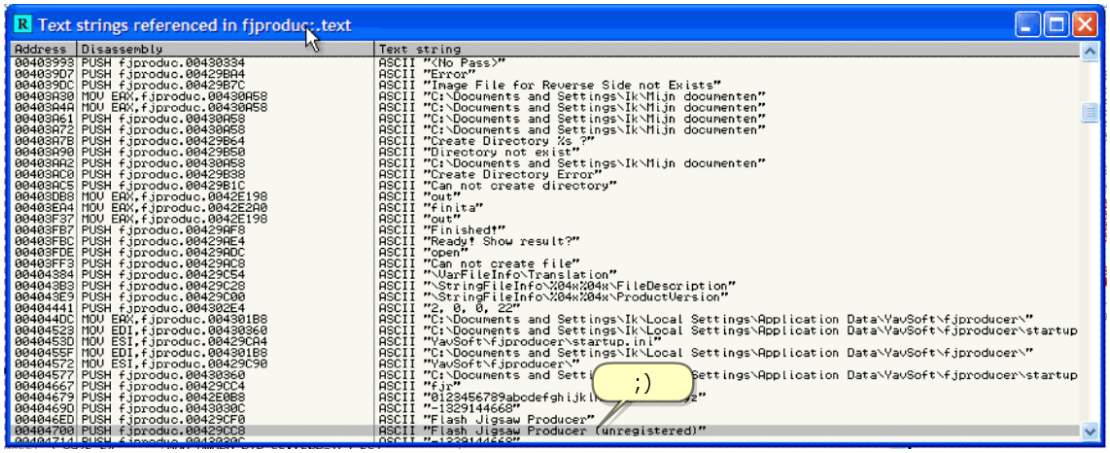


;)



Remember the string on top of the window

Window 의 top 에 있던 string 을 기억해.



;)

Scroll up to see better

좀 더 좋게 보기 위해 scroll 올려.

Yes! This is what's on the window top

예! 이것은 window top 이다.

Doubleclick to go there

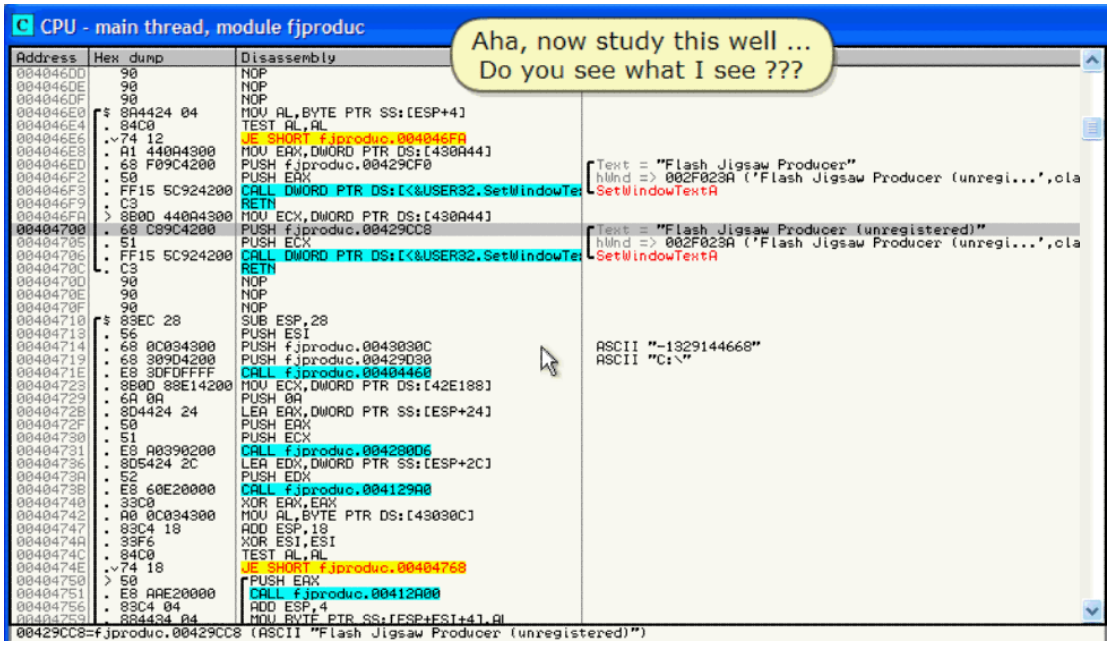
Doubleclick 해서 가자.

Now let's see

이제 살펴보자.

And scroll down

그리고 scroll 내려.



Aha, now study this well ...

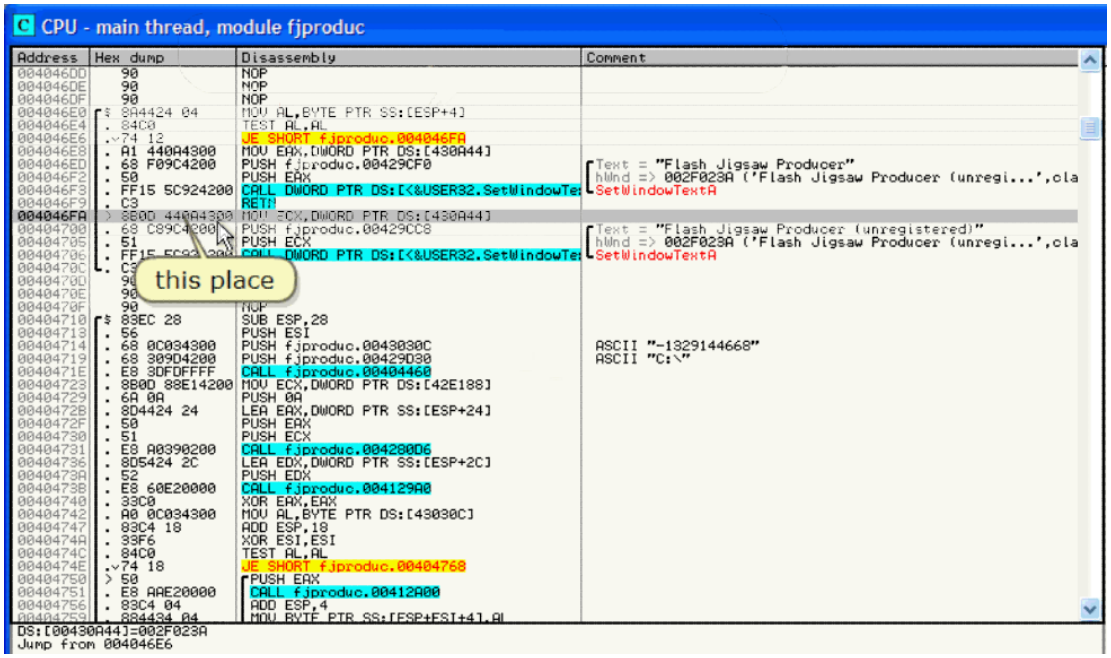
Do you see what I see ???

아하, 이제 이것을 공부하자.

내가 보는 것을 너도 보고 있어 ???

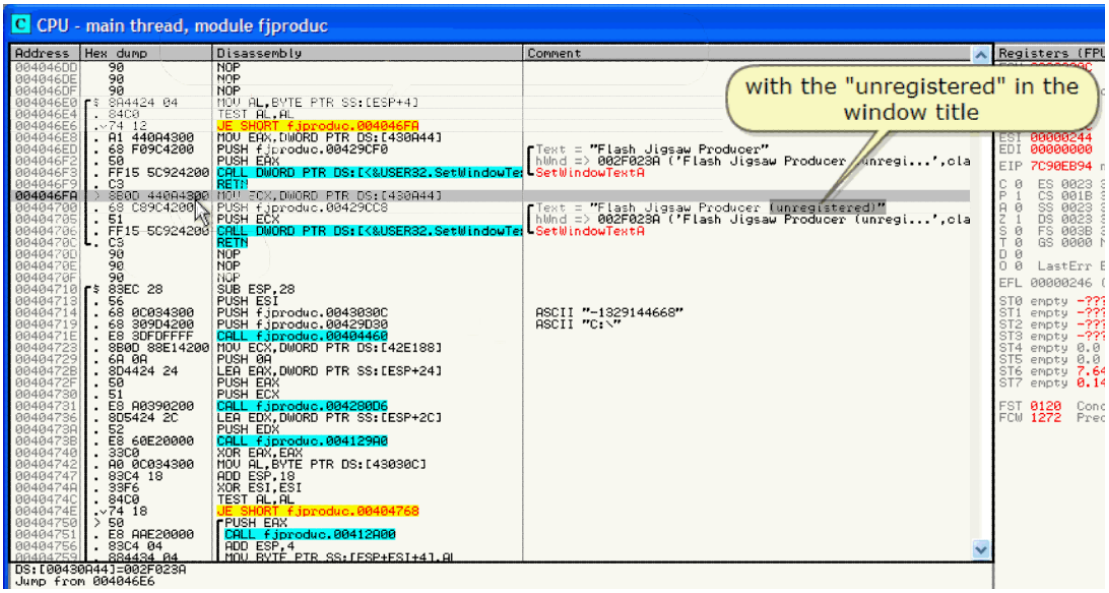
In short : ESP + 4 will decide whether jump is taken to ...

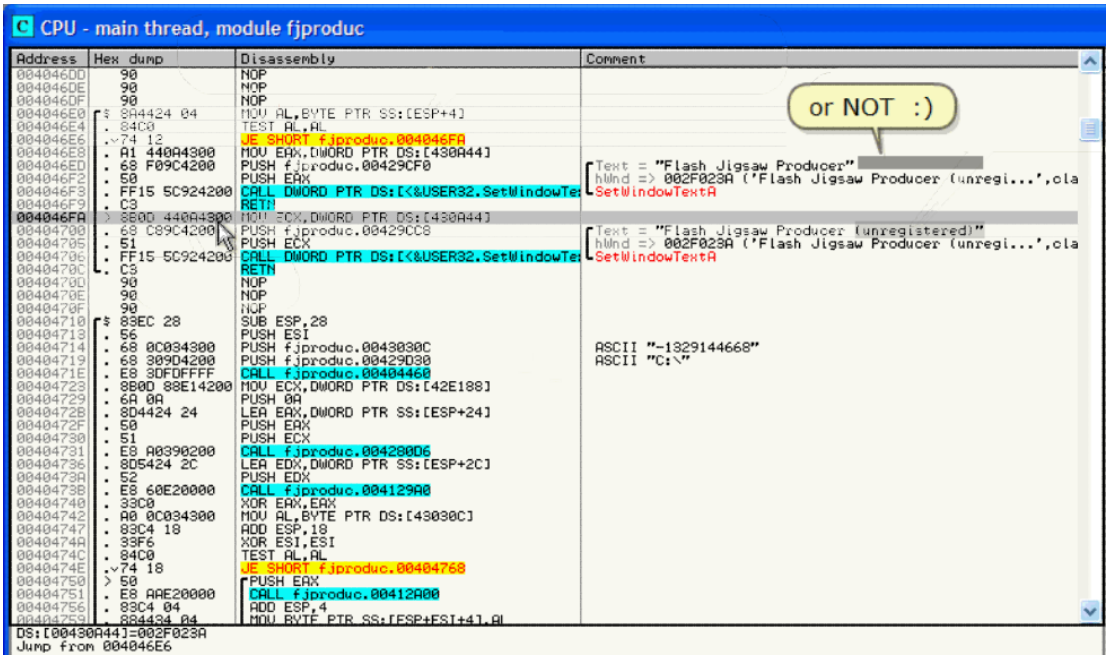
요약 : ESP + 4 는 jump 할 것인지 말 것인지 결정할 수 있다.



This place

이곳에 위치해.





With the "unregistered" in the window title

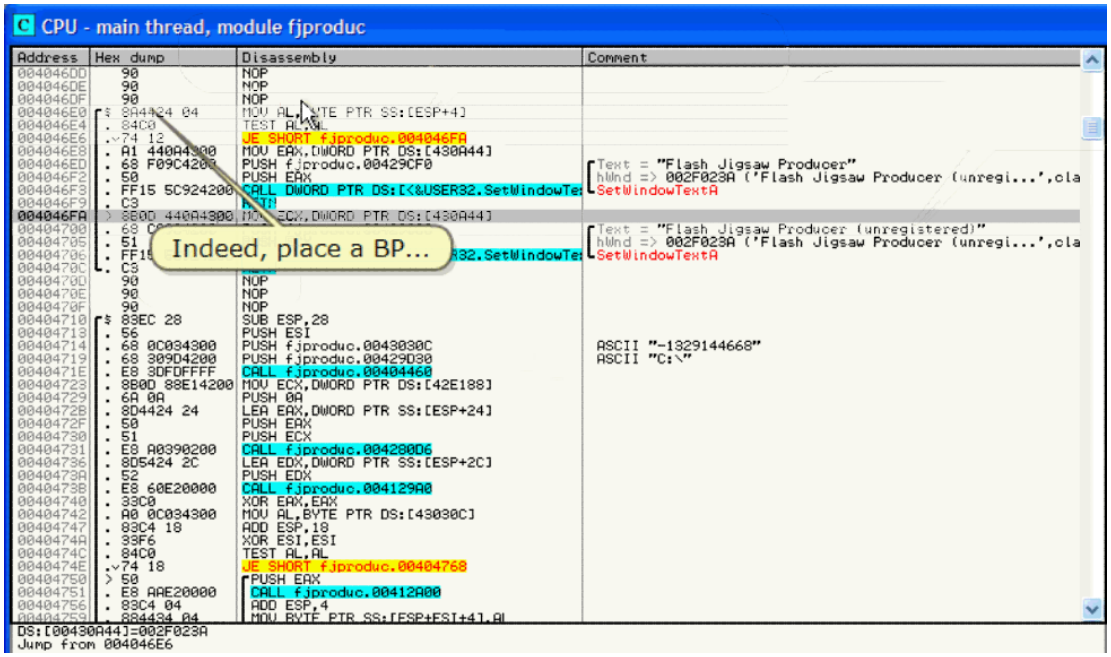
Or NOT :)

Window title 에서 "미등록" 함께 있다.

또는 없다 :)

Now, what does this tell us ??

이제, 무엇을 해야 하는지 알겠지 ??



Indeed, place a BP...

BP 를 설치해...

Here by doubleclicking the opcode

Opcode 를 doubleclick 해.

And re-start the app

그리고 app 을 re-start 해.

And re-run

그리고 재시작 해.

We land at our BP

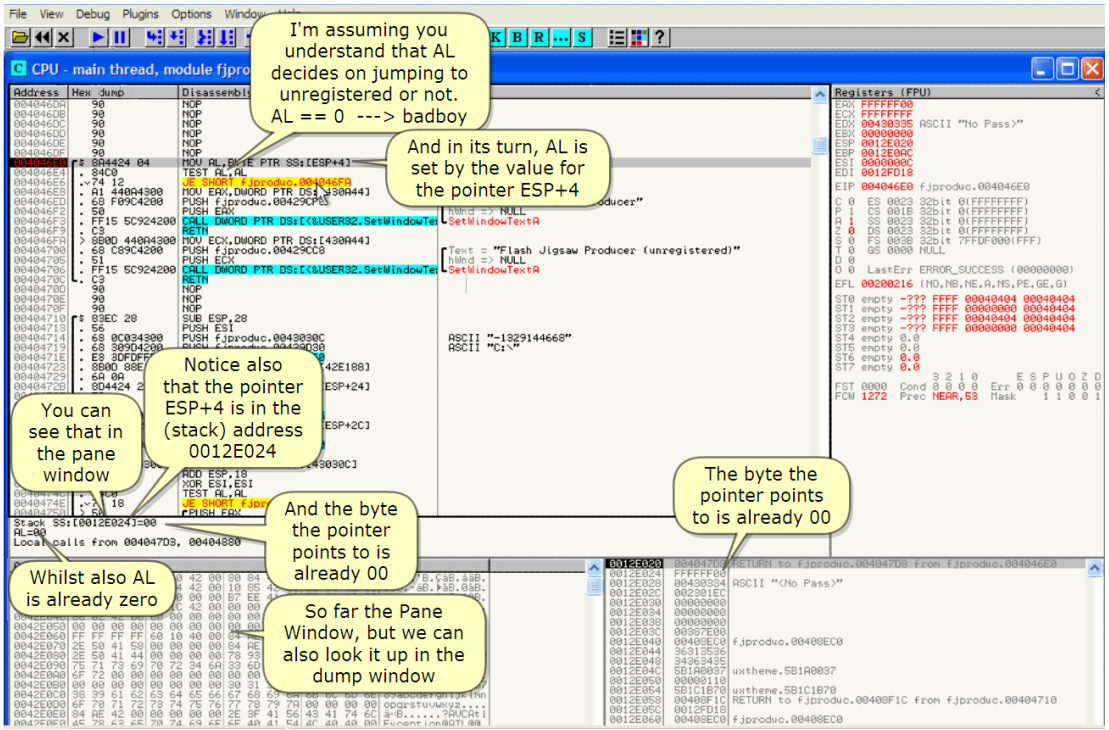
우리는 우리의 BP 에 도착했다.

Scroll up a little to see clear ;)

명확히 보기 위해 scroll 을 약간 올려 ;)

And study the code here.

여기 code 를 공부해.



I'm assuming you understand that AL decides on jumping to unregistered or not.

AL == 0 ---> badboy

AL 이 jump 에서 미등록 됐는지 등록 됐는지 결정한다는 것을 이해할 거라고 생각한다.

AL == 0 ---> badboy

And in its turn AL is set by the value for the pointer ESP+4

AL 이 바뀌는 것은 pointer ESP+4 의 값에 의해 set 된다.

You can see that in the pane window

너는 pane window 를 볼 수 있다.

Notice also that the pointer ESP+4 in the (stack) address 0012E024

Stack address 0012E024 에 ESP+4 pointer 가 있다.

And the byte the pointer points to is already 00

Byte pointer 는 이미 00 을 가리킨다.

The byte the pointer points to is already 00

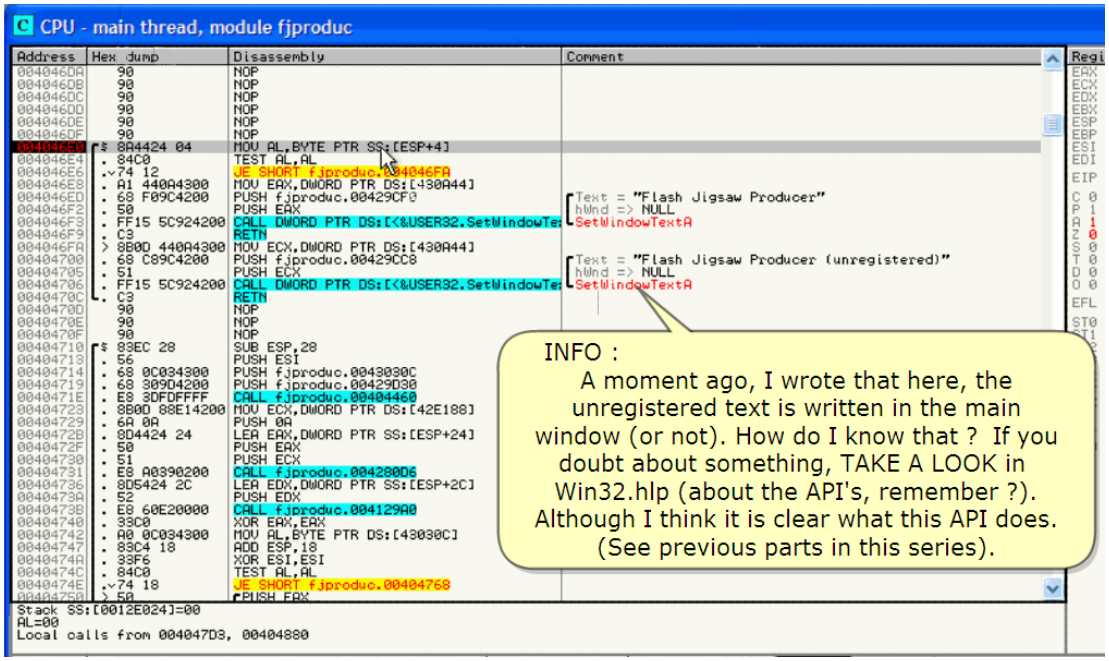
Byte pointer 는 이미 00 을 가리킨다.

Whilst also AL is already zero

또한 al 은 이미 zero 다.

So far the pane Window, but we can also look it up in the dump window

지금까지 pane Window 를 살펴봤다. 그러나 우리는 dump window 에서 볼 수 있다.



INFO :

A moment ago, I wrote that here, the unregistered text is written in the main window (or not).

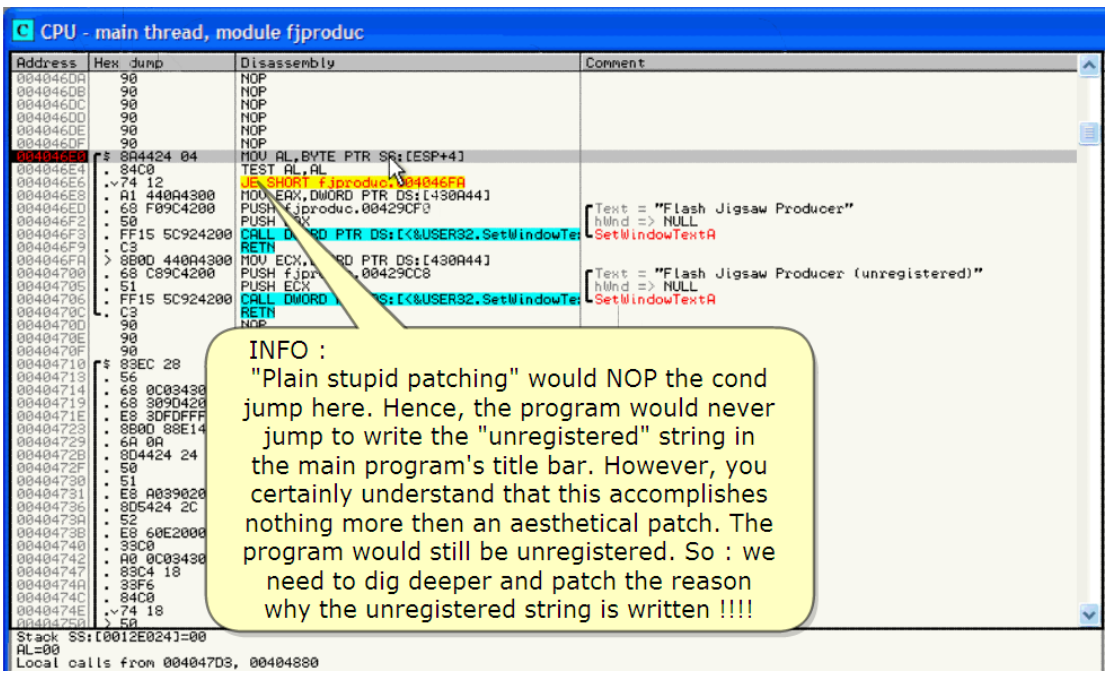
약간 전에, 나는 이것을 썼다. 미등록 test 는 main window 에 저장되어 있다.(아니거나)

How do I know that? If you doubt about something, TAKE A LOOK in Win32.hlp (about the API's remember ?).

내가 어떻게 아는가? 무언가를 의심한다면, Win32.hlp 를 살펴 봐.(API's 에 대하여 기억해?)

Although I think it is clear what this API does. (See previous parts in this series).

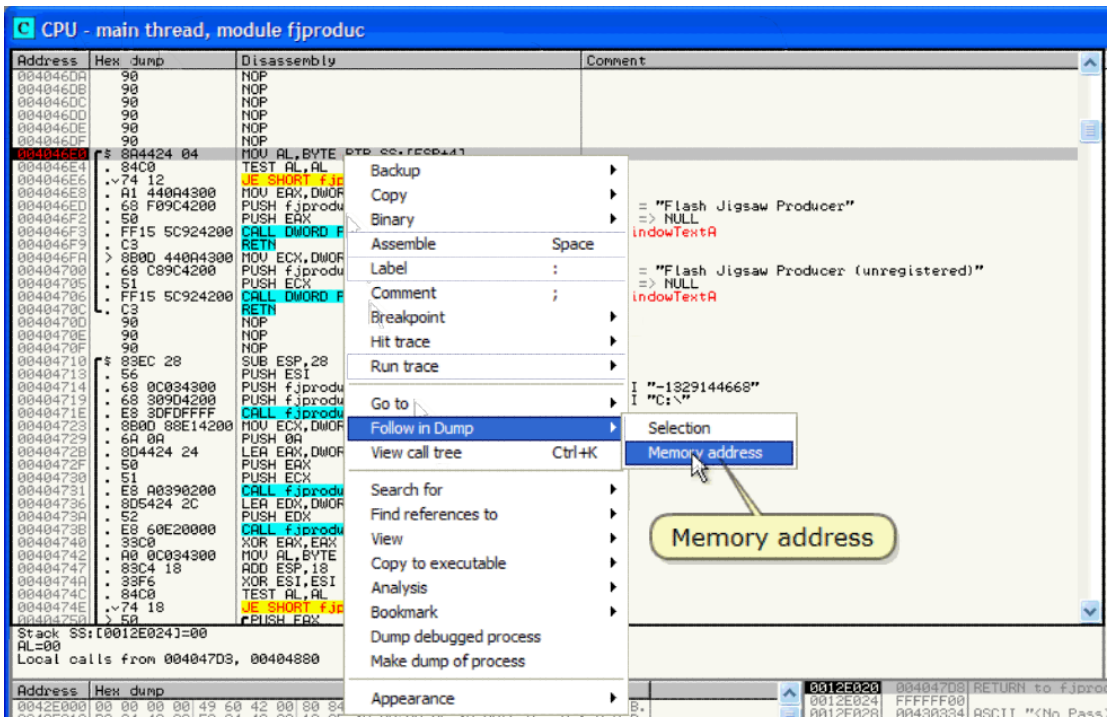
이것은 내 생각이지만 이 API 가 하는 일이 명확하다.(이 series 의 이전 parts 를 봐라)



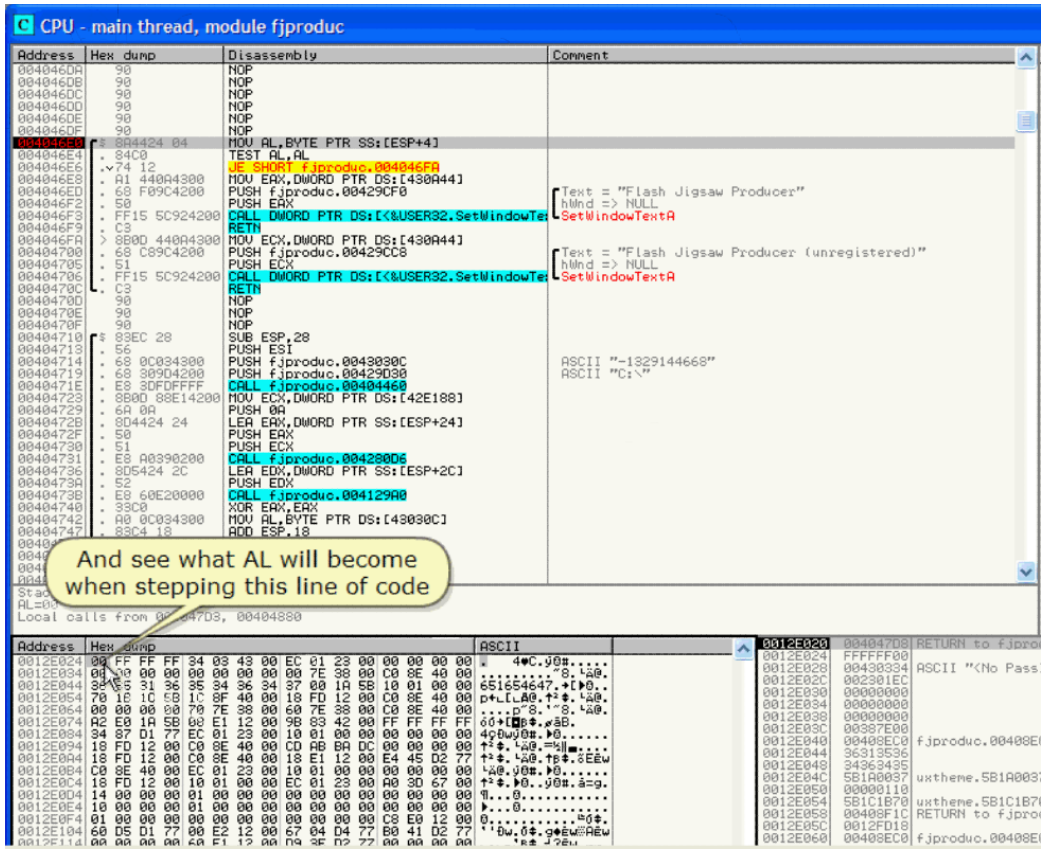
INFO :

"Plain stupid patching" would NOP the cond jump here.

"명백히 멍청한 patching" cond jump 를 NOP 하는 것이다.
Hence, the program would never jump to write the "unregistered" string in the main program's title bar.
그리하여, program 은 main program's title bar 의 "unregistered" string 을 쓰기 위하여 절대 jump 하지 않는다.
However, you certainly understand that this accomplishes nothing more then an aesthetical patch.
그러나 너는 명확히 심미적인 patch 보다 더 이상 성취된 것이 없다는 것을 이해했다.
The program would still be unregistered. So : we need to dig deeper and patch the reason why the unregistered string is written !!!!
Program 은 여전히 미등록 상태다. 그래서 : 우리는 깊게 들어가는 것과 미등록 된 string 을 작성된 이유를 patch 해야 합니다.

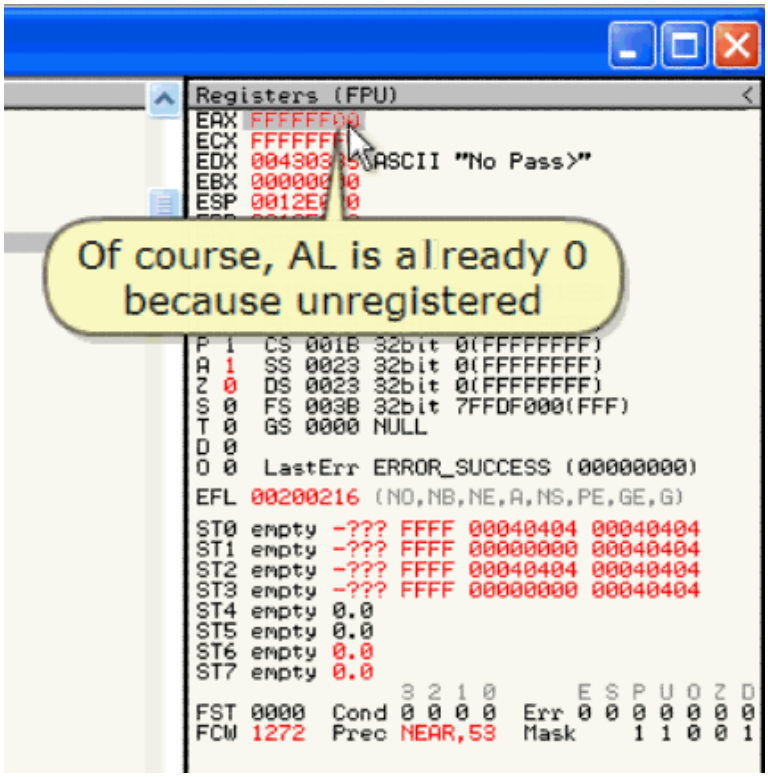


Follow in dump
Dump 를 따라가자.
Memory address



And see what AL will become when stepping this line of code

이 code line 을 넘길 때 AL 이 어떻게 되는지 보자.

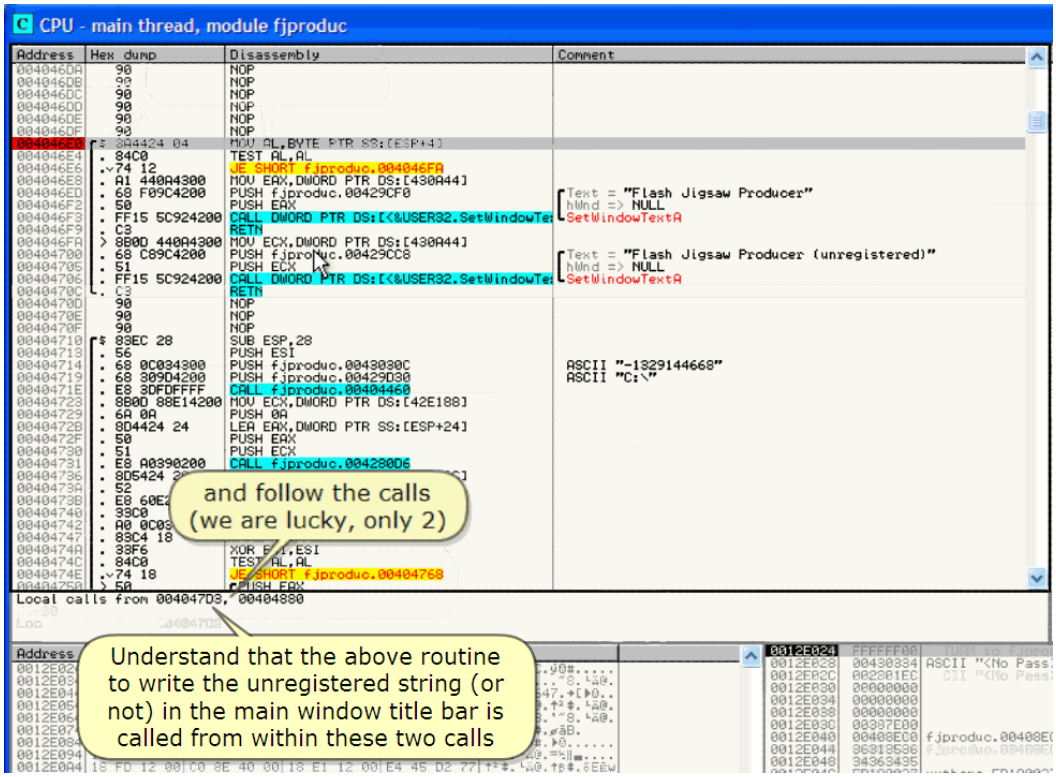


Of course, AL is already 0 because unregistered

물론, AL 은 이미 0 이다. 왜냐하면 미등록이기 때문이다.
So, let's see where ESP+4 is set to 0
그래서, ESP+4 가 0 으로 set 되는 것을 보자.
Now, let's take a look just before this code
이제, 이 code 후를 보자.
And find where ESP+4 is set to 0
ESP+4 가 0 으로 set 되는 곳을 찾자.

INFO :
I will show you here the basics of finding your way in the code.
code 에서 기본적인 너만의 찾는 방법을 보여줄 것이다.
There are a lot more advanced ways to do this.
이 일을 하기 위해 꽤 많은 진화된 방법이 있다.
We will discuss these in later parts in this series.
우리는 이 series 의 나중 part 에서 이것을 의논할 수 있다.
However, like told many times already : I think it is good to know the basics too ...
그러나, 이미 많은 시간 동안 말했다 : basic 을 좋다는 것을 알고 있다.
For example : we can get where we want to arrive by using the Call Stack or by using the Stack or by placing a memory BP on write on the byte at 12E024 etc etc.
예를 들어 : 사용하고 있는 Call Stack 이나 사용하고 있는 Stack 이나 memory 에 위치한 BP 에 의하여 12E024, etc, etc 에 도착할 수 있다.
Be patient, all in due time ;)
다 때가 되면 돼. 조금만 참아.

By the way,
This place of code is in calls
Calls 에서 code 조각이다.
So point this line
이 line 은 가리킨다.



And follow the calls

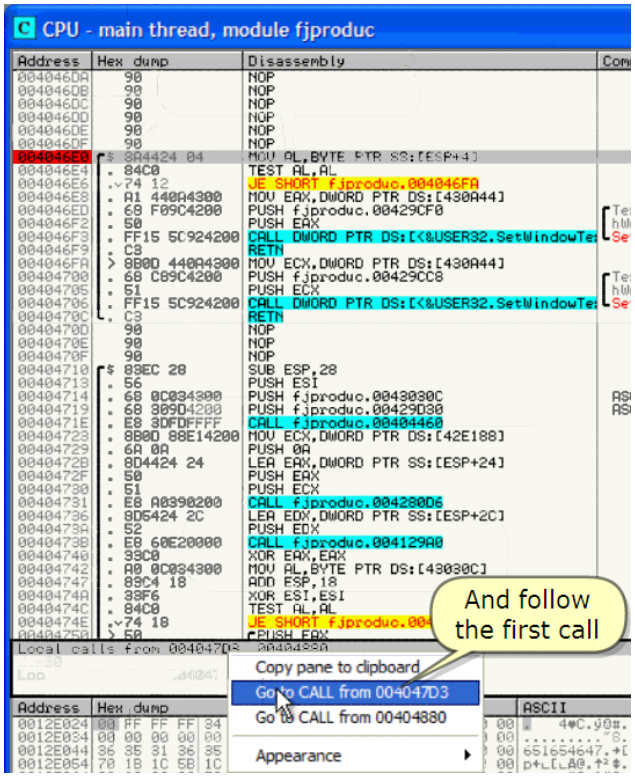
(we are lucky, only 2)

이 calls 을 따라간다.

(우리는 운이 좋다, 오직 2 가지 밖에 없다.)

Understand that the above routine to write the unregistered string (or not) in the main window title bar is called from within these two calls

Main title bar 에 등록되지 않은 문자열(또는 아니거나)을 작성하려면 위의 routine 이 2 호출에서 호출되는 것으로 알고 있습니다.



And follow the first call

Place BP

And push <enter> to follow the call back

첫번째 call 을 따라가자.

BP 를 설치해.

<enter>를 눌러 Call back 을 따라가자.

번역 주)번역을 하면서 알게 된 점. Enter 는 바로 call 문으로 들어가는 것이고 F7 은 그 line 에서 call 문이 있을 때 따라 들어가는 것이다. 즉 enter 는 다음 EIP 가 가리키고 있는 주소가 아니더라도 바로 call 문을 따라 들어갈 수 있다.

And follow the second call too

And put BP

And restart app ….

2 번째 call 을 따라가자.

BP 를 설치하자.

App 을 재시작 해.

And re-run the app to see ….

보기 위해 app 을 재시작 하자.

….(scroll up)….

Scroll 올려

….where ESP+4 is set

ESP+4 가 set 되는 곳이다.

So, this is the call that is interesting for us

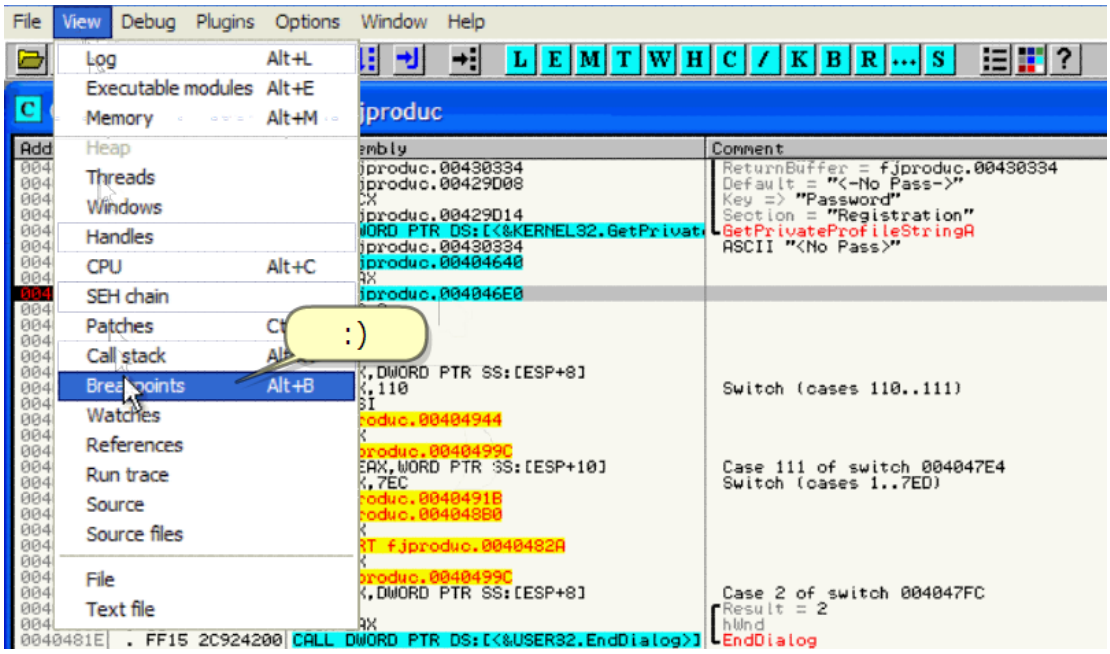
그래서, 이 call 은 우리에게 흥미롭다.

Meaning that this is the call that calls the routine at startup that sets the unregistered text(or not) on top of the main window

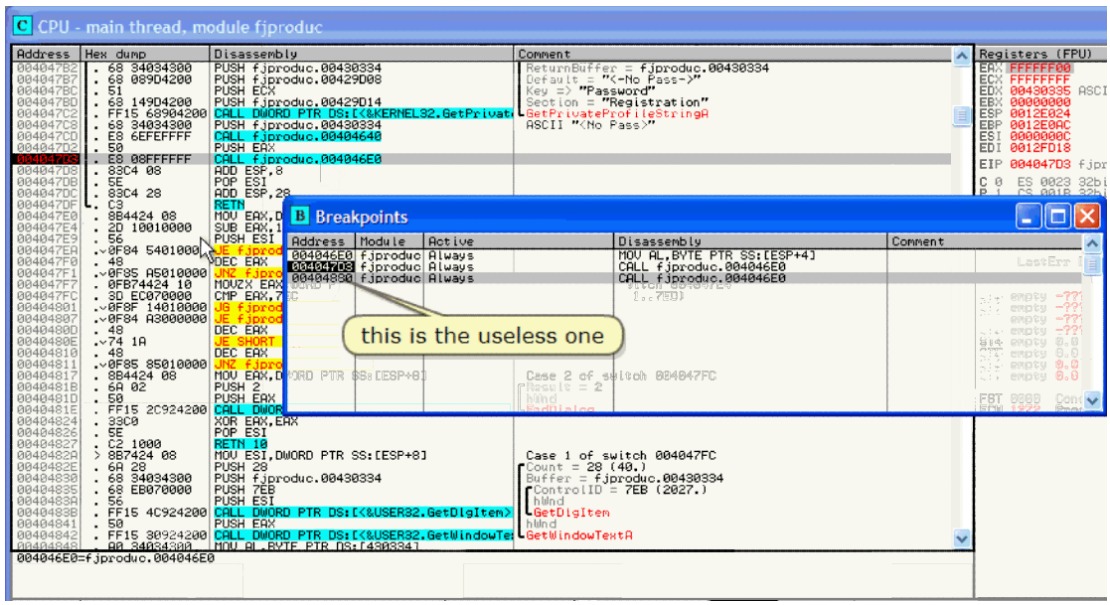
이것은 Startup routine 을 부르는 call 이다. 이 call 은 main window top 에서 미등록 text 로 set 한다.

Let's remove the other BP

다른 BP 는 삭제해.

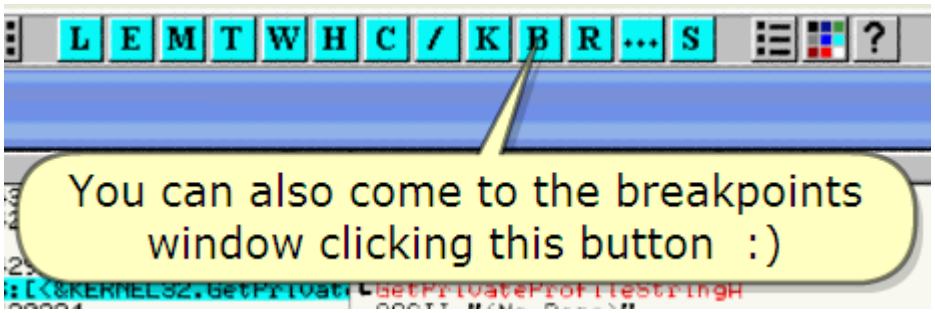


:)



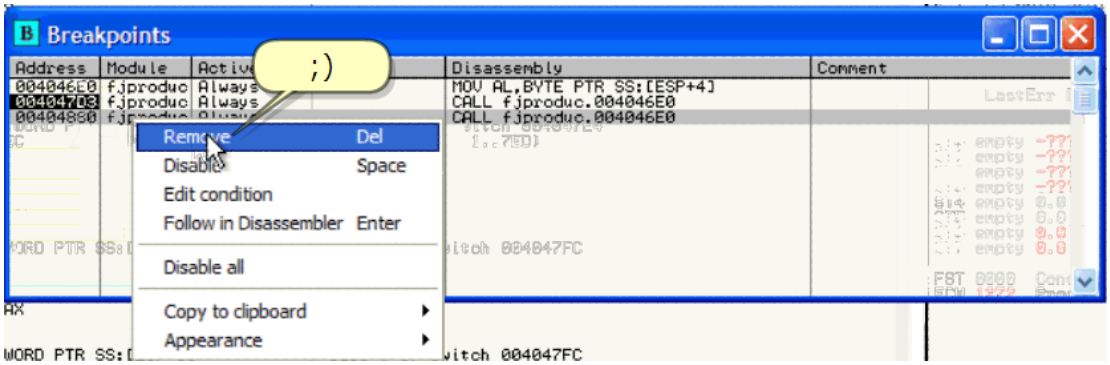
This is the useless one

이것은 필요없다.



You can also come to the breakpoints window clicking this button :)

너는 또한 이 button 을 click 해 breakpoints window 로 갈 수 있다.



;)

Now think with me ...

나와 같이 생각하자...

... it must all happen in this call

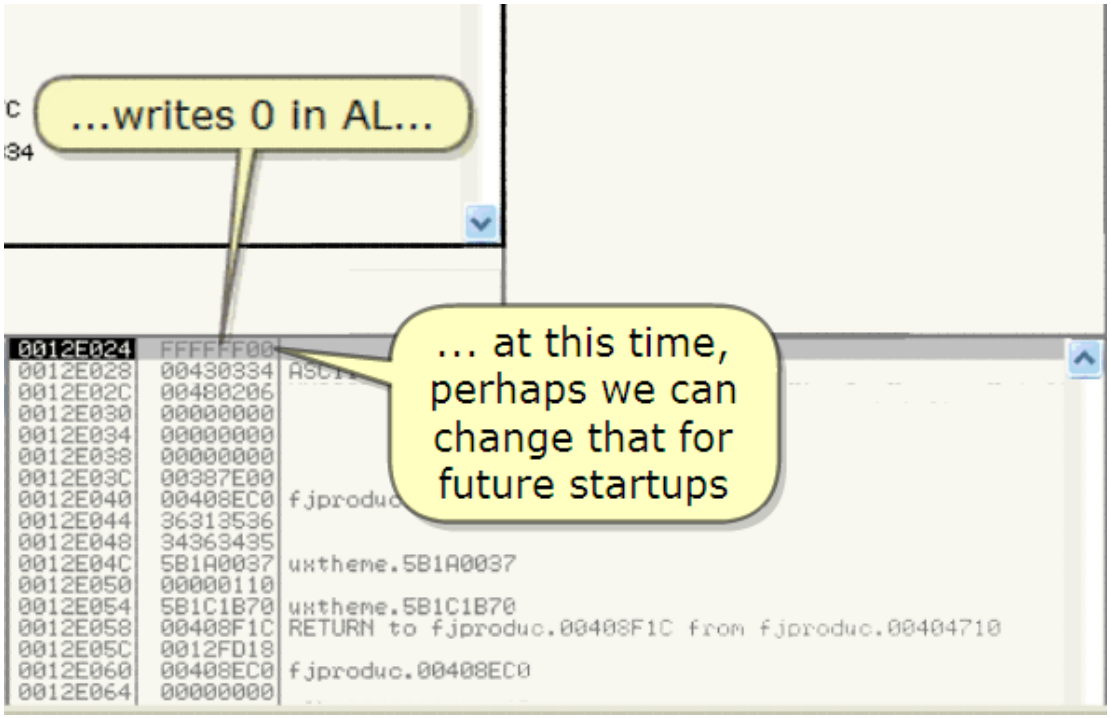
이 call 은 무슨 일이 발생한다.

Because in this call, the program needs to know already if it is registered or not(to put the "unregistered" or not)

왜냐하면 이 call 은, program 은 이것이 등록하는 것인지 아닌지 알기 위해 필요하다.("unregistered"를 넣는지 아닌지)

And we note that push EAX....

EAX 에 넣어라.



...writes 0 in AL...

AL 에 0 을 써.

... at this time, perhaps we can change that for future startups

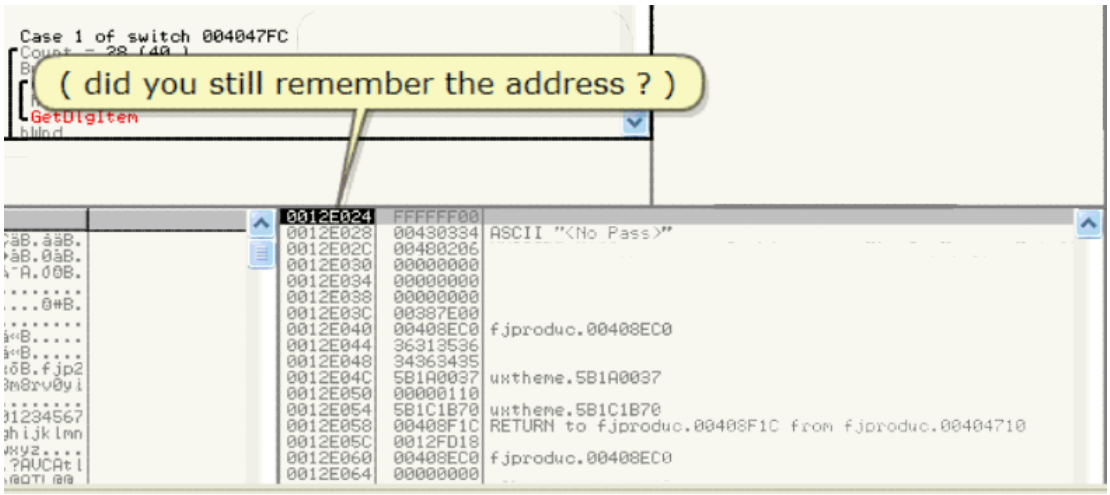
... 이 시간에, 아마 우리는 미래의 startup 을 바꿀 수 있다.

... in ESP+4 ...

ESP+4 에서

So, AL writes zero ...

AL 은 zero 를 썼다.

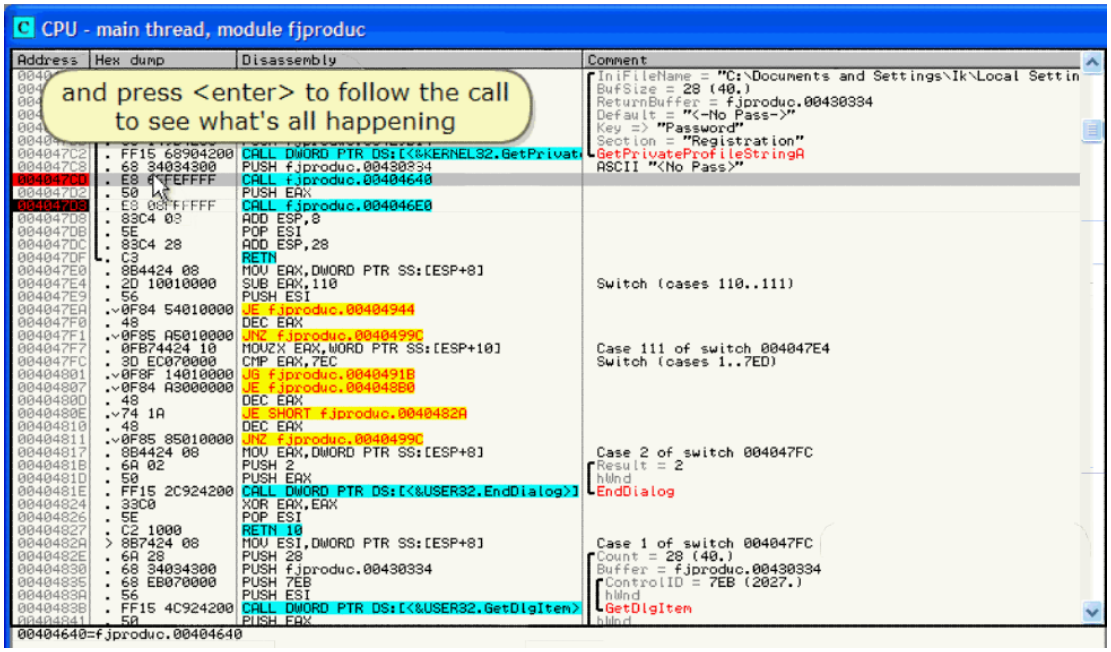


(did you still remember the address ?)

(지금도 이 address 를 기억해?)

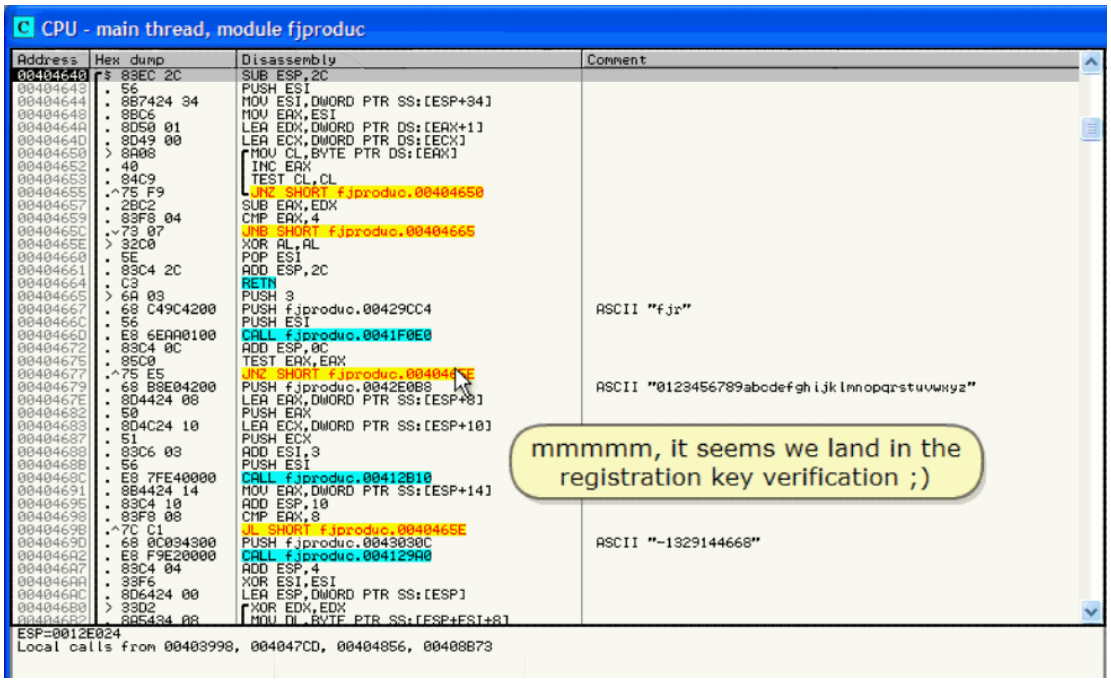
So, place BP here

BP 를 여기에 설치해.



And press <enter> to follow the call to see what's all happening

무슨 일이 일어나는지 보기 위해 이 call 을 따라가자. Enter 를 눌러.



Mmmmm, it seems we land in the registration key verification ;)

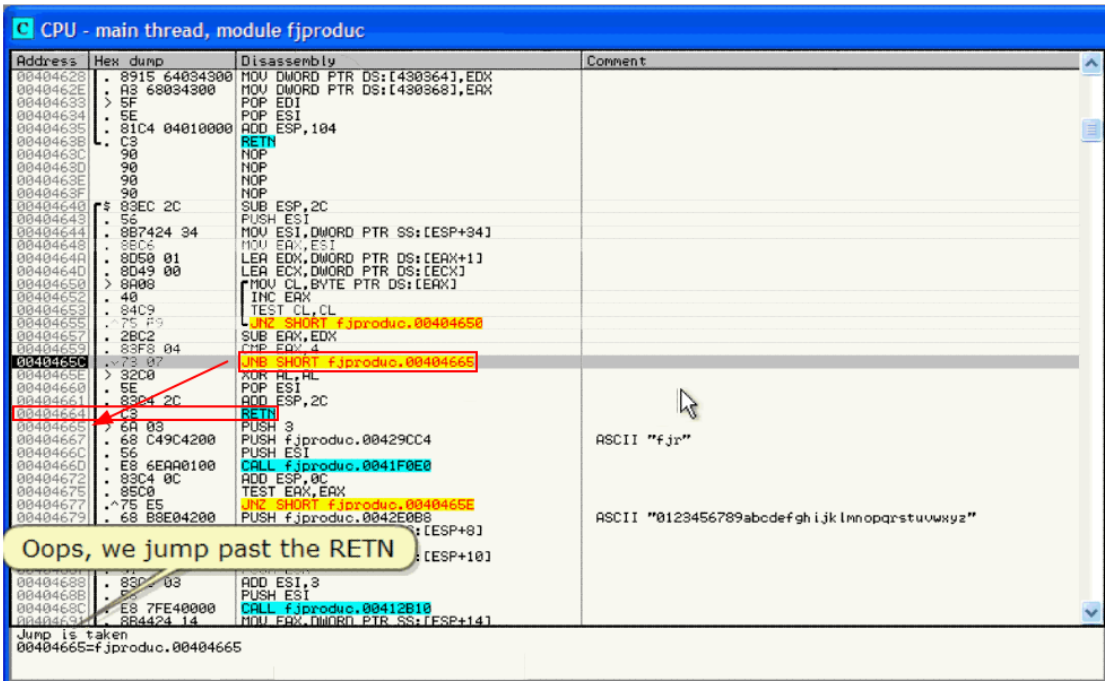
음, 우리는 registration key 검증하는 곳에 도착했다는 것을 보여준다.

Scroll down to see better

좀 더 보기 좋게 scroll 내려.

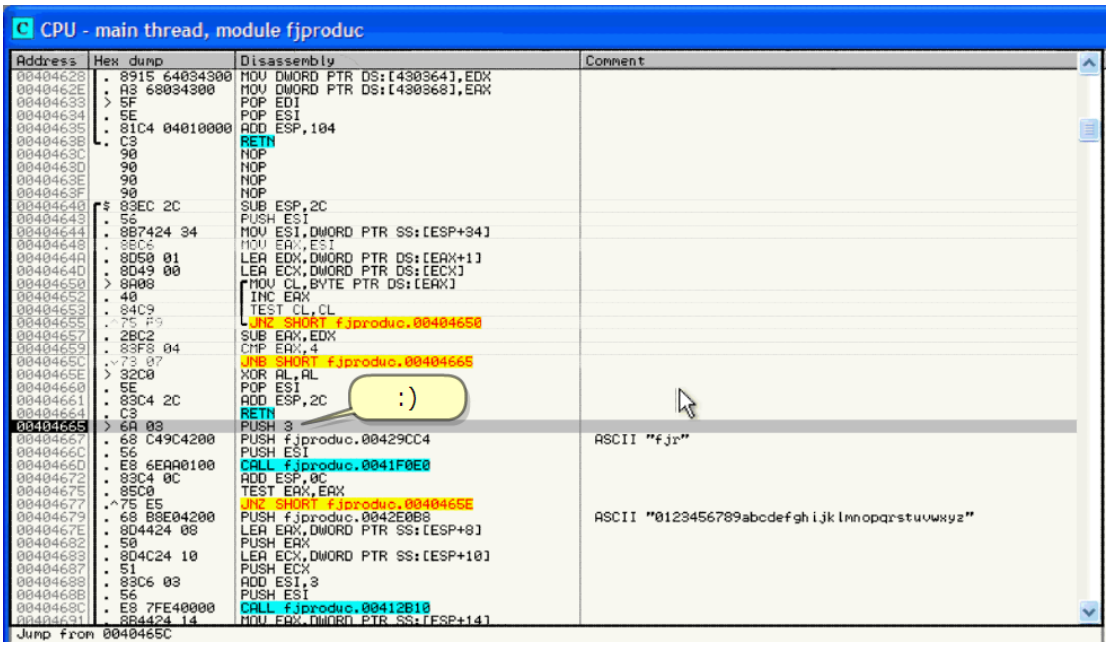
Step over the code to see what happens(F8)

무슨 일이 일어나는지 보기 위해 code 를 실행 해(F8)

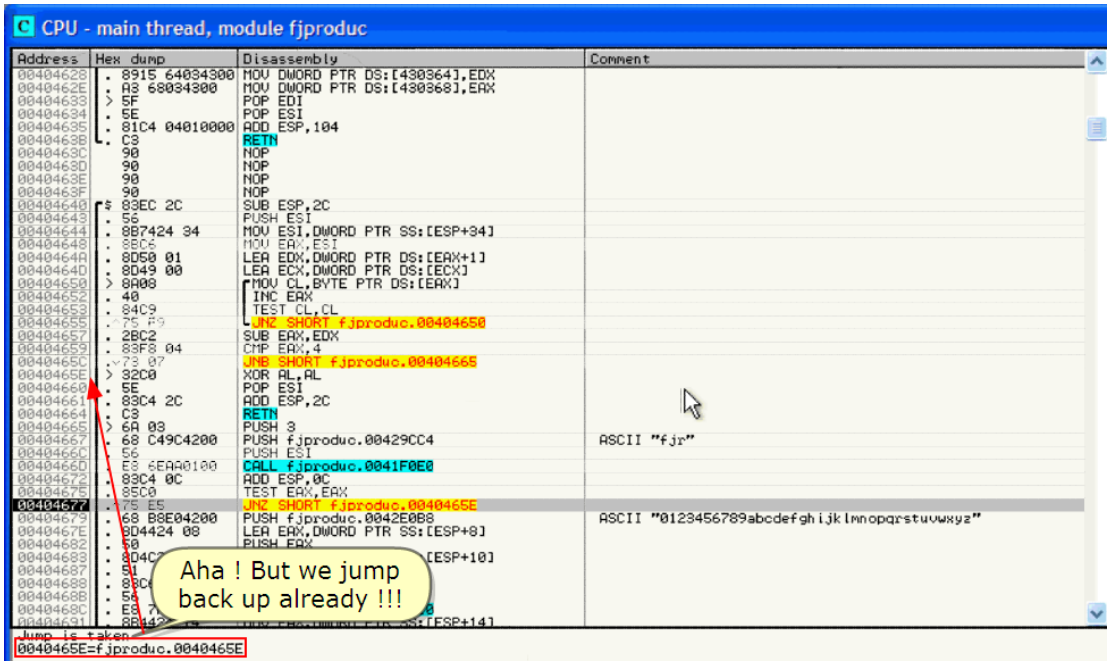


Oops, we jump past the retn

웁스, 우리는 retn 을 지나서 jump 한다.



:)



Aha! But we jump back up already !!!

아하! 우리는 jump 하기 전으로 jump 한다.

We jumped to this XOR, AL, AL

우리는 XOR AL, AL 을 jump 했다.

So, from here on we can't miss the RETN anymore

그래서, 여기에서 더 이상 retn 을 놓치지 않을 수 없다.

And it is clear that AL is set to zero here

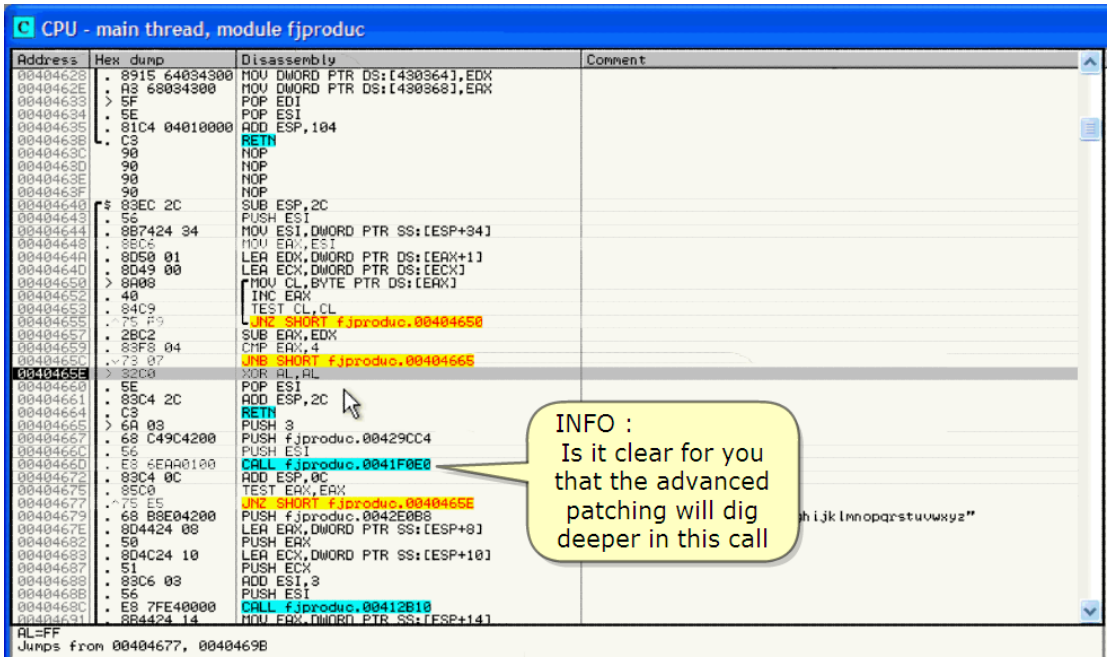
이것은 명확하다. AL 은 여기에서 zero 로 set 된다.

This is the place to patch

Patch 하기 위해 여기에 놓자.

(INTERMEDIATE patching level)

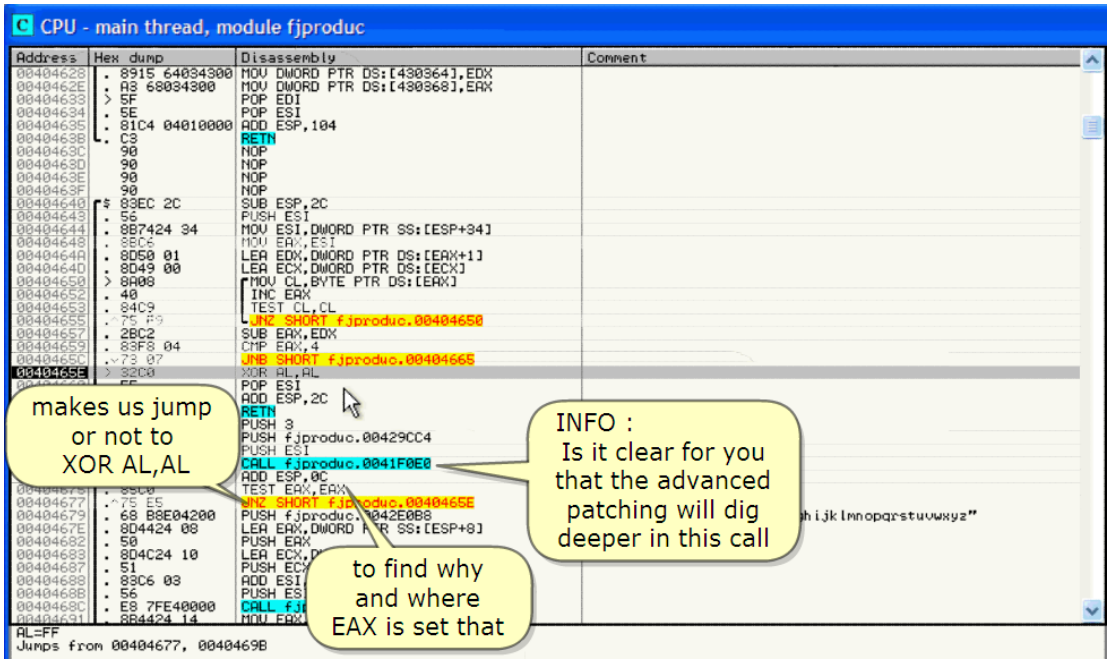
(중간 patching level)



INFO :

Is it clear for you that the advanced patching will dig deeper in this call

진화적인 patch 는 이 call 에서 더 깊게 파고 들어야 한다.



To find why and where EAX is set that

왜 그리고 어디에서 EAX 가 set 되는지 찾자.

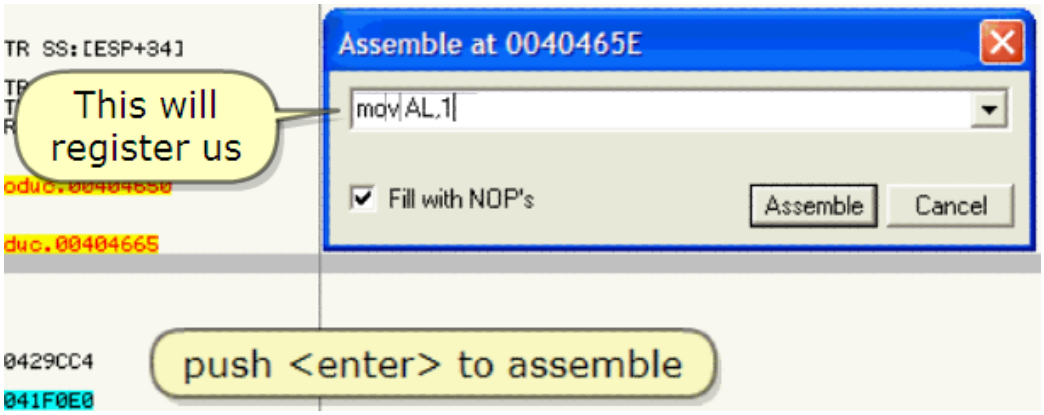
Makes us jump or not to XOR AL, AL

XOR AL, AL 에서 Jump 하거나 하지 않을 수 있다.

Scroll down

Scroll 내려.

Let's try intermediate patching and see if it works.
NOTE : advanced patching is often required !!! (See later parts in this series)
중간 patch 를 해라. 그리고 어떤 일을 하는지 봐.
기록 : 진화된 patching 은 자주 요구된다 !!!(이 series 의 나중 part 를 봐)



This will register us
Push <enter> to assemble
등록할 수 있다.
Assemble 하기 위해 <enter>를 눌러.

And save the changes
바뀐 것을 저장해.

;)
;)
Under another name
다름 이름으로
And press <enter> to save
저장하기 위해 <enter>를 눌러.
Load the patched program and verify
Patch 된 program 을 load 해 그리고 검증 해
;)

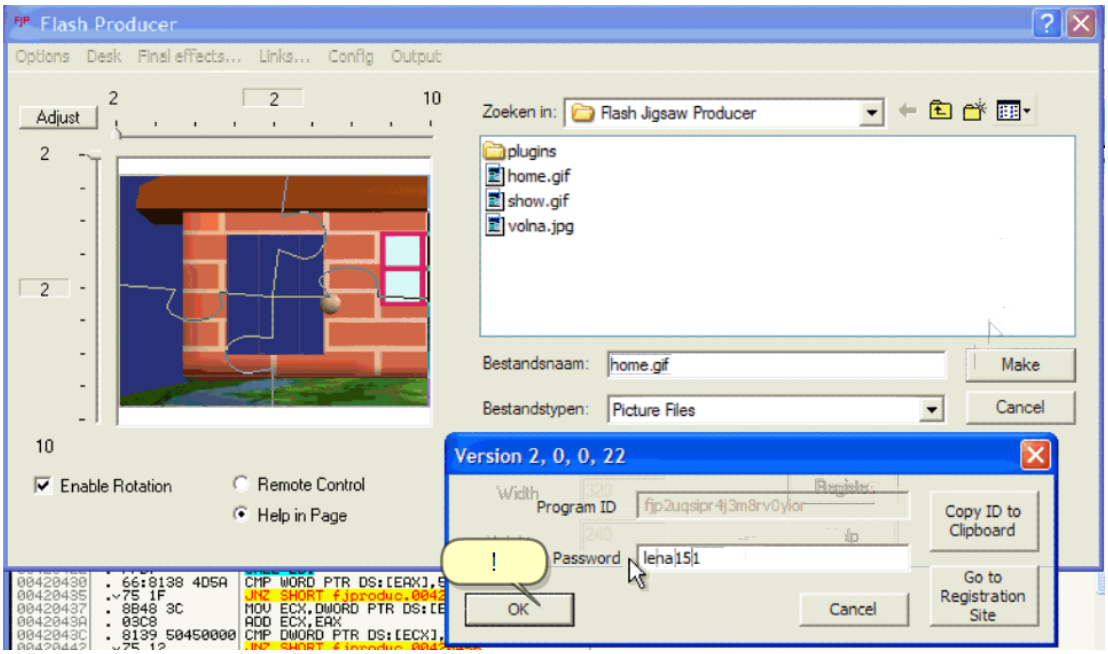
5. Testing the patched app

And run
실행 해.

Mmmm, see this here ...

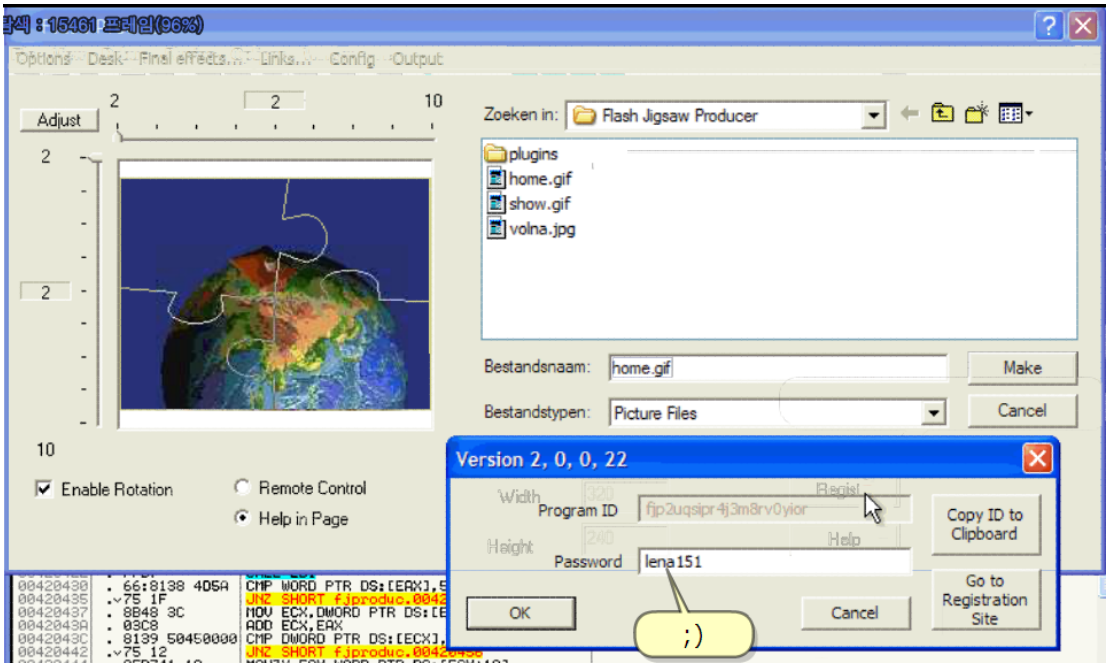
... no unregistered
음, 여기를 봐.
Unregistered 가 없다.

Aha, and this works too
And what if we ....
아하, 우리는 할 수 있다.
만약 우리가...



!
Let's close and restart huh
닫고 재시작 하자.

What do you think ???
어떻게 됐을 거라 생각해?



;)

6. Conclusion

In this part 11, the primary goal was to learn to think like a reverser by simply following the indications from Olly's Pane window while patching an application at intermediate level.
이번 part11 에서, application 을 중간 level 로 patching 할 때 중요한 목표는 Olly's Pane window 에서 간단히 지시자를 따라가며 배우고 생각하기 위해 reverser 같이 해야한다.

I hope you understood everything fine and I also hope someone somewhere learned something from this. See ma back in part 12 ;)
네가 모든 것을 좋게 이해했기를 희망한다. 또한 누구든지 어디서든지 이것에서 무언가를 배웠으면 좋겠다. Part 12 에서 보자.

The other parts are available at
다른 parts 는 사용 가능하다.
http://tinyurl.com/27dzdn (tuts4you)
http://tinyurl.com/r89zq (SnD Filez)
http://tinyurl.com/l6srv (fixdown)

Regards to all and especially to you for taking the time to look at this tutorial.
Lena151 (2006, updated 2007)
모두에게 안부를 전하고 특별히 이 tutorial 에 시간을 투자해준 너에게 감사한다.