07.Intermediate level patching, Kanal in PEiD

2012 년 1 월 28 일 토요일
오후 8:27

Hello everybody.
모두들 안녕.
Welcome to this Part 7 in my series about reversing for newbies/beginners.
나의 초보자 reversing series Part 7 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.

1. Abstract

This Part 7 is about reversing a "real" application to learn something about patching a registration scheme in the intermediate level patching method.
이번 Part 7 에서는 "real" application 을 중간 level patching 방법 중의 registration scheme patching 하는 법을 배우겠습니다.
Remember that Part #6 dealt with patching in the plain stupid way.
기억해? Part 6 에서 명백히 멍청한 방법으로 patch 했다.
A real application 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.
네가 초보자라면 좀 더 좋은 이해력을 얻으려면, 이 movie 를 보기 전에 이 series 의 이전 part 를 먼저 봐라.
The goal of this tutorial is to teach you something about a program's behaviour.
이 tutorial 의 목표는 program's behaviour 를 너에게 가르치는 것이다.
In my search not to harm authors, I came across MrBills.
나의 MrBills 연구는 제작자에게 피해를 주지 않는다.
This Program does not exist any longer : the company was sold and the program discontinued.
이 program 은 더 이상 존재하지 않는다 : 회사는 program 을 판매를 중단했다.
However, because it could be misused, I only included the main executable (not the install exe !) for your research in the shown techniques.
잘못된 사용을 하게 두지 않는다. 나는 보여지는 기술에서 너의 연구를 위해 오직 main executable 을 포함했다.(install exe 가 아니다)
This makes the program useless for other purposes then studying material.

Program 은 다른 목적으로 공부하는 것이 필요없다.
Taking a look in the specialized media, I also found this application to be "cracked" already.
특화된 media 를 봐라. 나는 또한 이 application 에서 이미 "cracked" 된 버전을 찾았다.
Here, this software is only chosen because it is ideal for this tutorial in reversing and it is targeted for educational purposes only.
여기, 이 software 는 오직 선택됐다. 왜냐하면 그것은 reversing 에서 이 tutorial 은 이상적이다.
그리고 그것은 오직 공부 목적의 target 이 됐다.
I hope you will exploit your newly acquired knowledge in a positive way.
In this matter, I also want to refer to Part 1.
네가 새롭게 얻은 지식을 긍정적인 방향으로 이용해.

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.

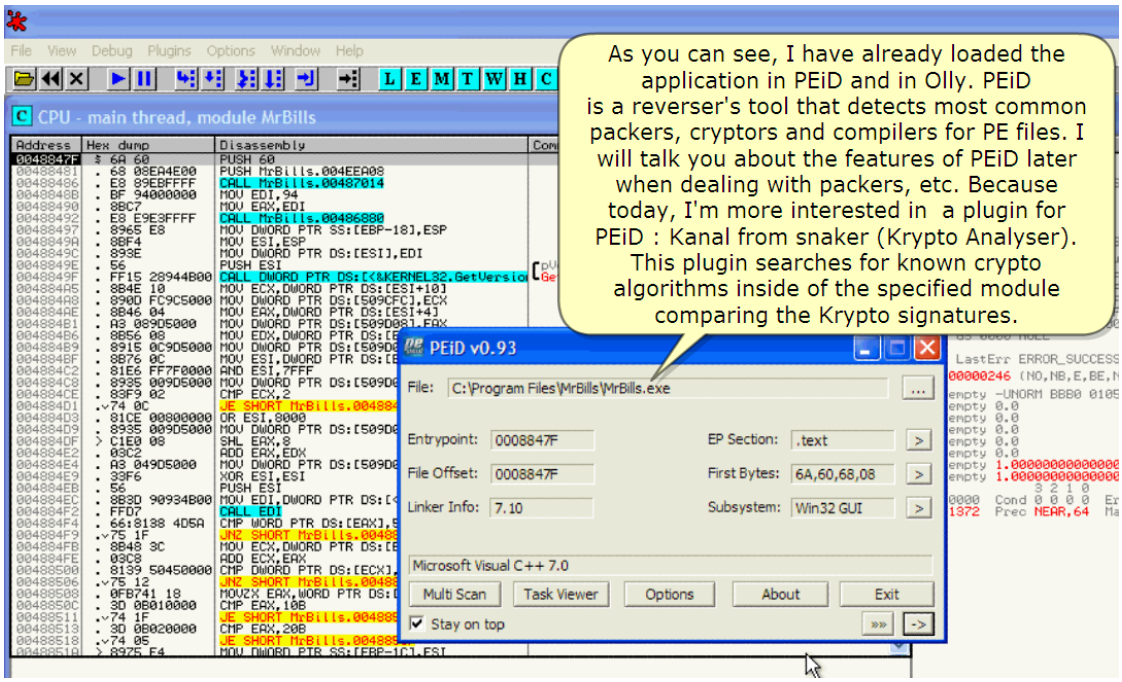
2. Tools and Target

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

PEID is a free tool and can be downloaded :
PEiD 는 공짜 tool 이고 download 할 수 있다.
http://peid.has.it

Again, the brain is your responsibility ;)
두뇌는 너의 책임감이다.
Todays target is a program called MrBills.
오늘 target program 은 MrBills 로 불린다.
Because it can no longer be downloaded, I have included the main executable-only in this package for your research.
왜냐하면 그것은 더 이상 download 되지 않는다. 나는 이 package 에서 너의 연구를 위해 오직 main executable 파일만 포함했다.

3. Behaviour of the program



As you can see, I have already loaded the application in PEiD and in Olly.
네가 봤을 때, 나는 이미 PEiD 와 Olly 에서 application 을 load 했다.
PEiD is a reverser's tool that detects most common packers, cryptors and compilers for PE files.

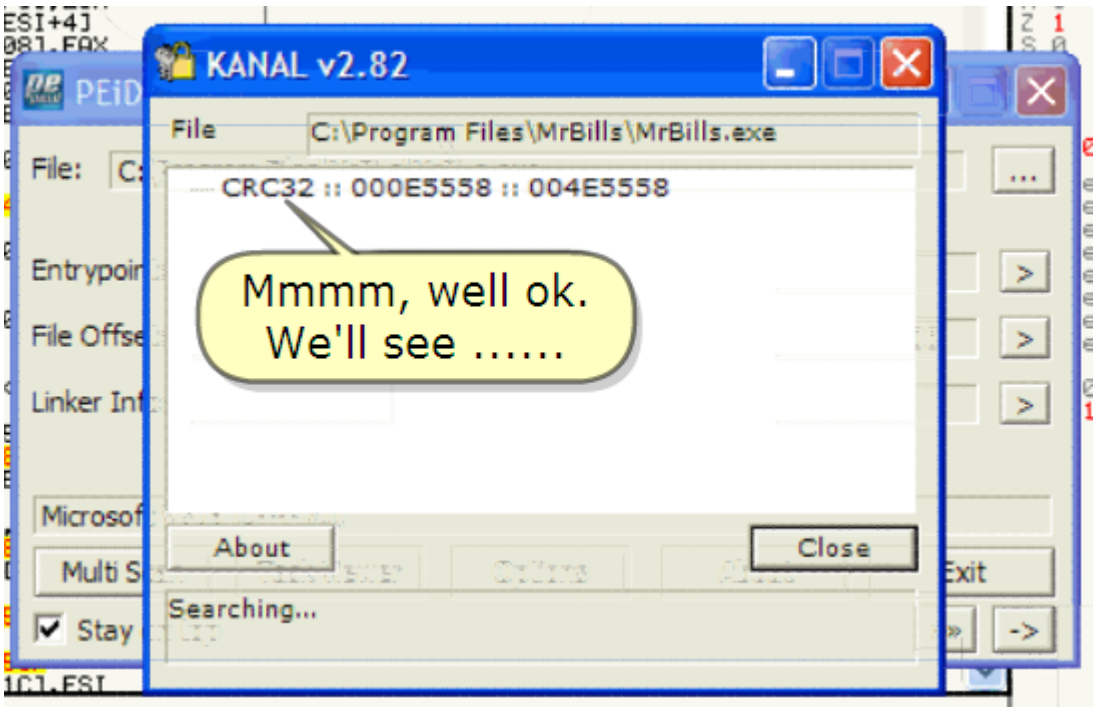
번역 주)Compiler: 특정 프로그래밍 언어로 쓰여 있는 문서를 다른 프로그래밍 언어로 옮기는 프로그램을 말한다. 원래의 문서를 소스 코드 혹은 원시 코드라고 부르고, 출력된 문서를 목적 코드라고 부른다. 목적 코드는 주로 다른 프로그램이나 하드웨어가 처리하기에 용이한 형태로 출력되지만 사람이 읽을 수 있는 문서 파일이나 그림 파일 등으로 옮기는 경우도 있다. 원시

코드에서 목적 코드로 옮기는 과정을 Compile 이라고 한다. Compiler 는 소스 프로그램을 읽어서 즉시 결과를 출력하는 인터프리터와는 구분된다. 소스 코드를 Compile 하는 이유는 대부분 사람에게 이해하기 쉬운 형태의 고수준 언어로부터 실행 가능한 기계어 프로그램을 만들기 위해서이다. 좁은 의미의 Compiler 는 주로 고수준 언어로 쓰인 소스 코드를 저수준 언어(assemble)로 번역하는 프로그램을 가리킨다.

PEiD 는 reverser's tool 이다. 그것은 공통 packer, cryptor, compiler 를 찾을 수 있다.
I will talk you about the features of PEiD later when dealing with packers, etc.
나중에 우리가 packer 와 deal 할 경우 PEiD 의 특징에 대해서 이야기 할 것이다.
Because today, I'm more interested in a plugin for PEiD : kanal from snaker (Krypto Analyzer).
오늘은, PEiD 를 위한 plugin 에서 좀 더 흥미롭다. : kanal from snaker(Krypto 해석기)
This plugin searches for known crypto algorithms inside of the specified module comparing the Krypto signatures.
이 plugin 은 알고 있는 crypto algorithm 의 명시된 module 에서 Krypto signature 를 비교하여 찾는다.

But, we can also already see that MrBills is not packed and that it has been compiled in MS VC++7.0
그러나, 우리는 이미 볼 수 있다. MrBills 는 pack 되어있지 않고 MS VC++7.0 으로 compile 됐다.

Today's software is MrBills v.2.1.0.1
오늘 software 는 MrBills v.2.1.0.1 이다.



Mmmm, well ok.

We'll see ……

우리는 볼 수 있다.

BTW, we will look into CRC checks in later parts in this series. Just stay tuned :)

우리는 이 series 의 나중에 나오는 part 에서 CRC check 를 볼 수 있다. 돌릴 수 있다. :)

So, here the target is opened in Olly…

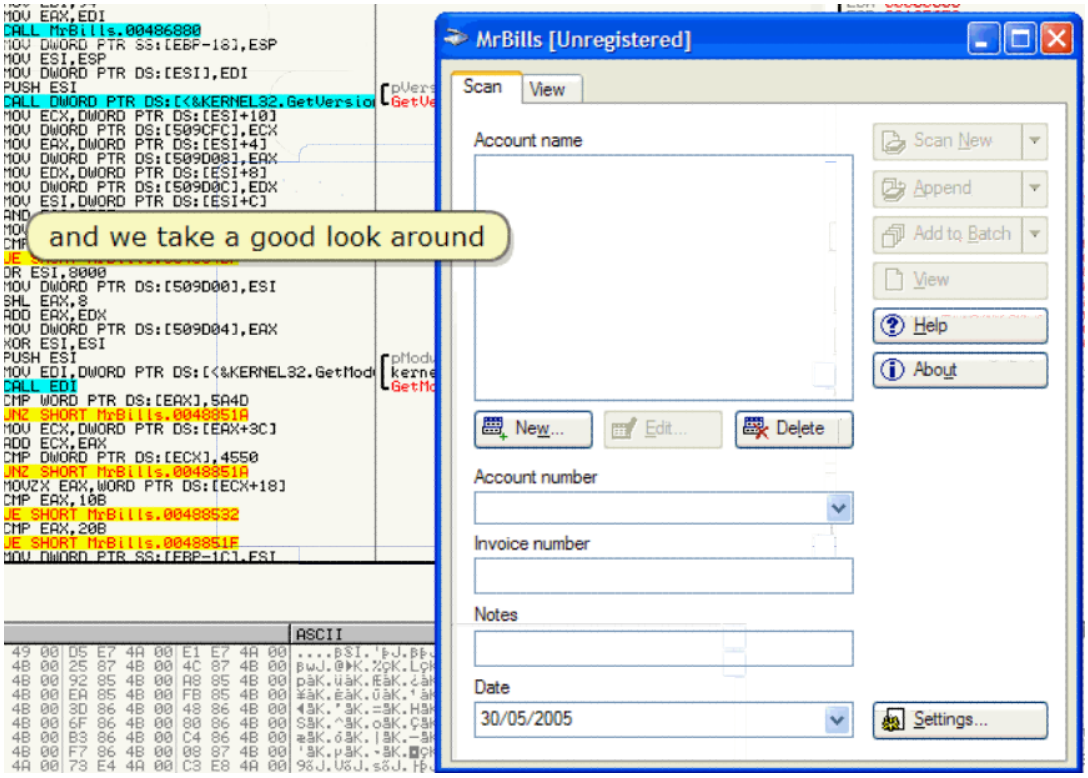
…and let's first explore

So, we run the software…

그래서, 여기 target 을 Olly 에서 열어봤다.

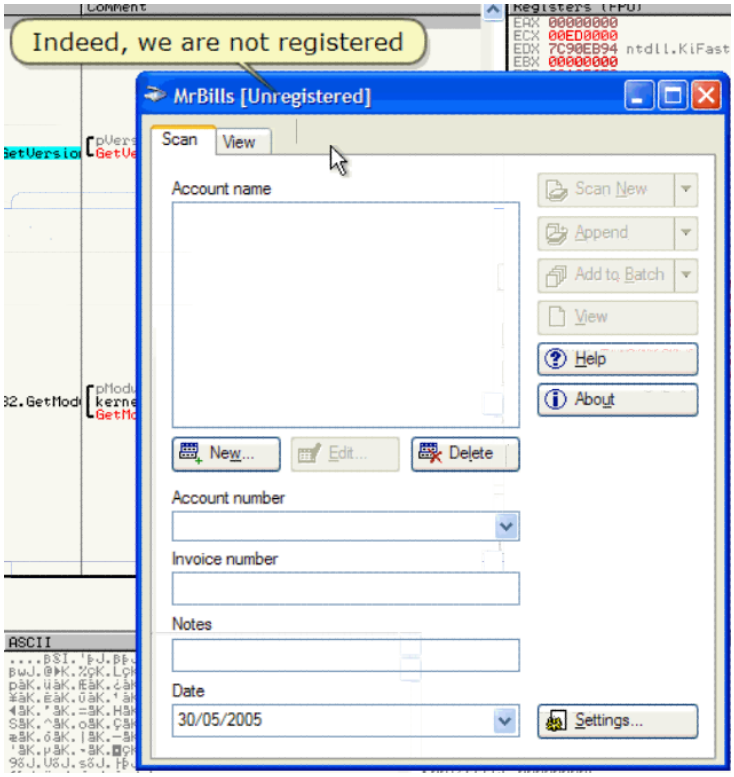
그리고 먼저 탐험하자.

그래서, 우리는 software 를 실행했다.



And we take a good look around

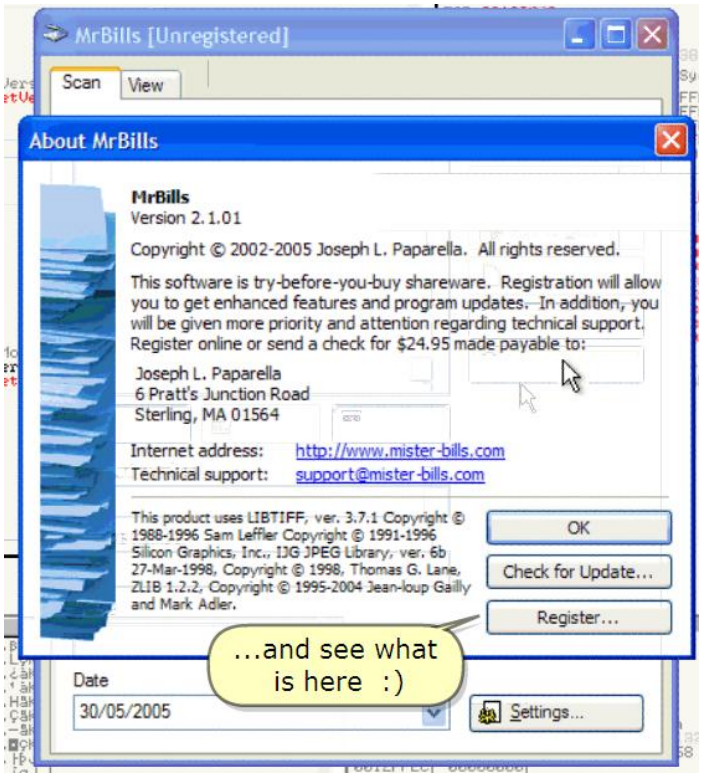
그리고 주변을 살펴봐.



Indeed, we are not registered

정말, 우리는 등록하지 않았다.

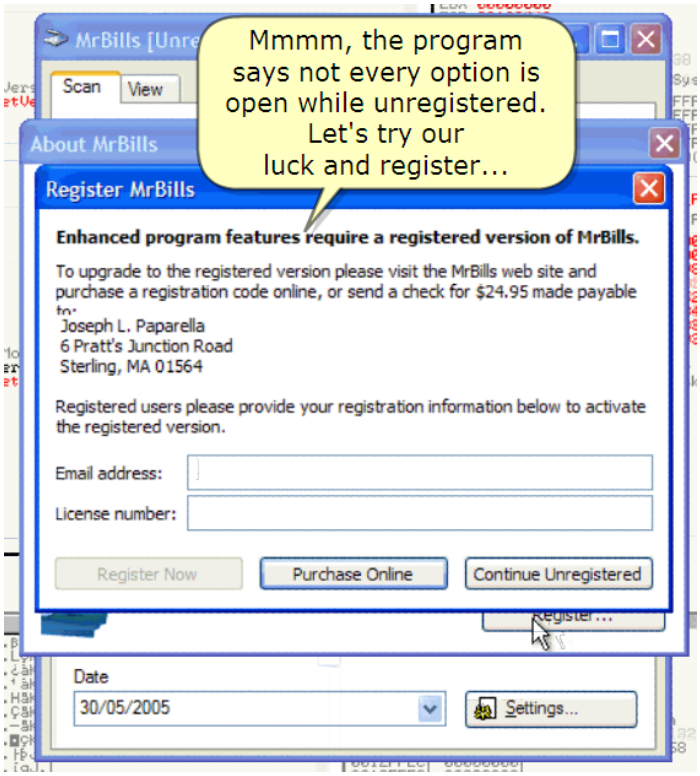
Mmmm, ok, let's go for it



...and see what is here :)

음, 이것을 위해 계속 가자.

그리고 이것이 무엇인지 봐.



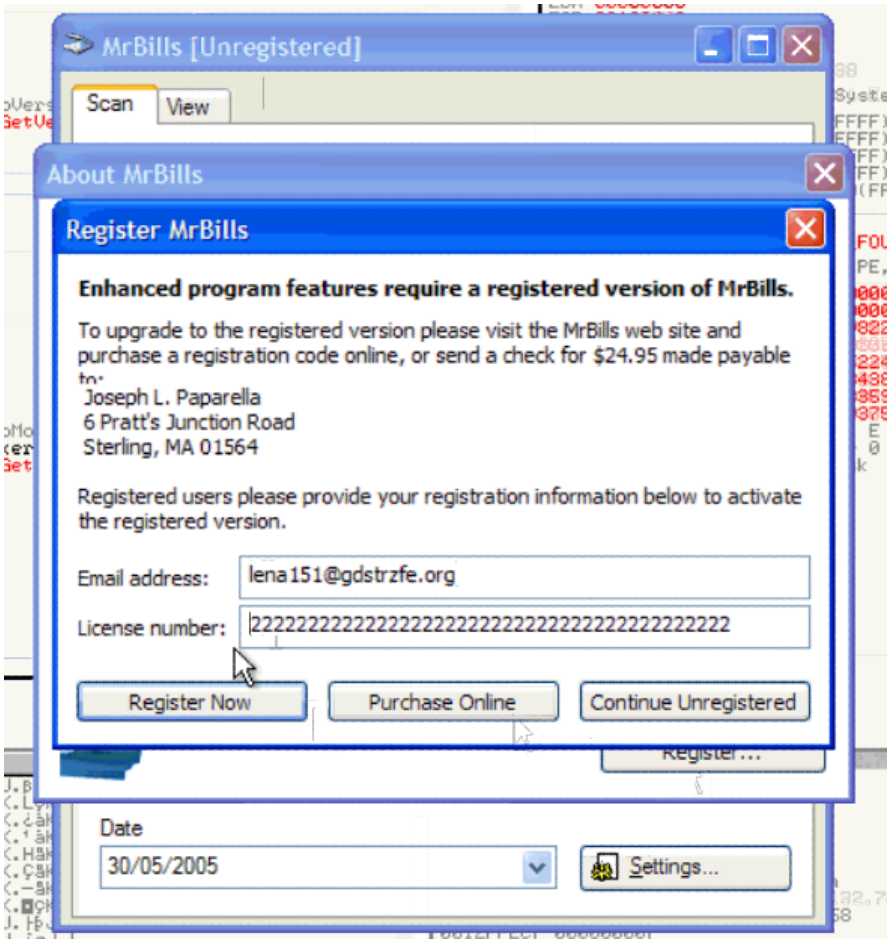
Mmmm, the program says not every option is open while unregistered.

Let's try our luck and register...

음, program 은 등록되지 않았을 때, 모든 option 을 열 수 없다.

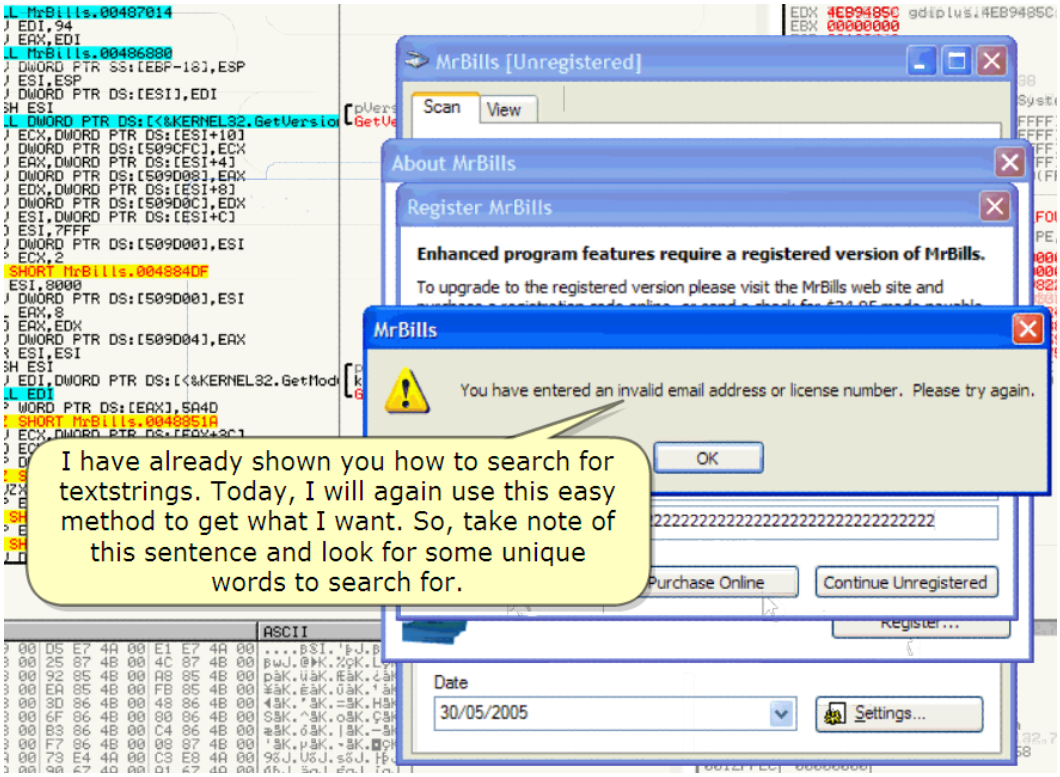
우리의 운을 믿고 등록해보자.

;)



All right. This should give us enough information. Let's go !
좋아. 너에게 충분한 정보를 주었다. 시작하자!

4. Finding the patches



I have already shown you how to search for textstrings.

나는 이미 너에게 어떻게 textstring 을 찾는지 보여주었다.

Today, I will again use this easy method to get what I want.

오늘, 나는 다시 내가 원하는 쉬운 방법을 사용하겠다.

So, take note of this sentence and look for some unique words to search for.

이 문장을 적어 그리고 약간의 유일한 단어들을 찾아라.

Fine, let's study all this better in the code.

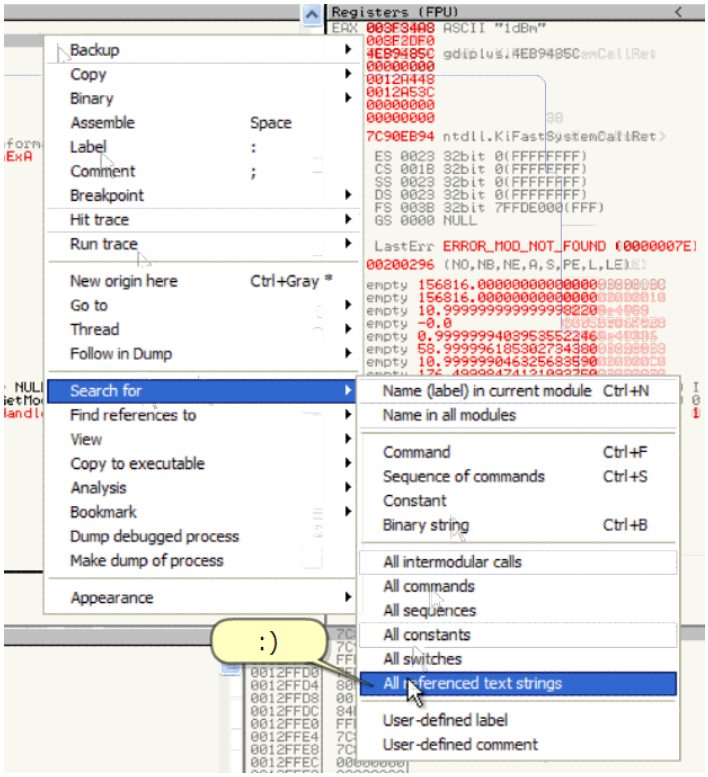
좋아, code 를 공부하자.

Return to Olly

And rightclick

Olly 로 돌아가.

그리고 rightclick



;)

When searching for textstrings, you need to remember to scroll up first.

Textstring 을 찾을 때, 먼저 scroll 을 올린 후에 찾는 기억이 필요하다.

Olly searches top to bottom and we don't want to miss anything. Now rightclick ...

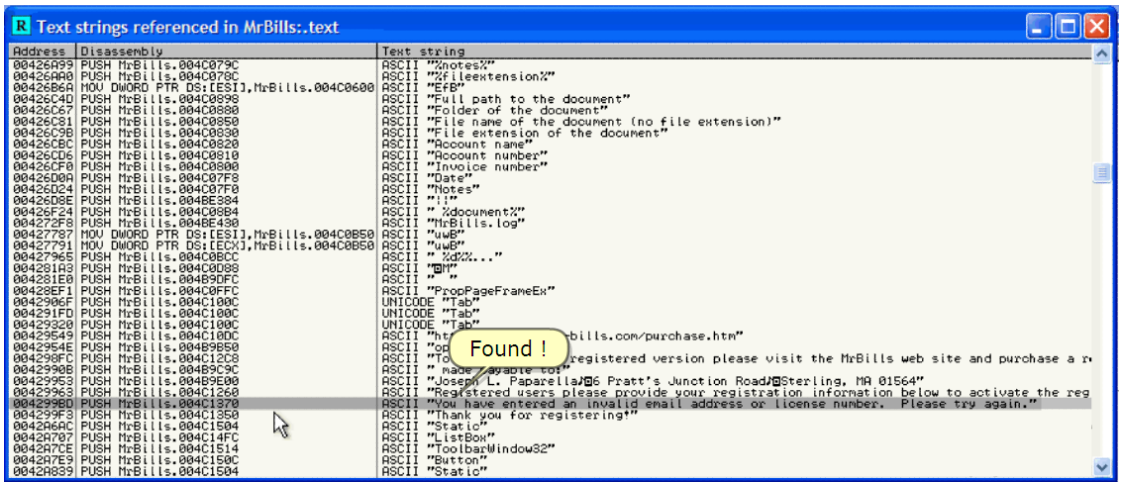
Olly 는 top 에서 bottom 까지 찾는다. 그리고 우리는 어떤 실수도 원하는지 않는다. 이제 rightclick...

And search for this textstring. Remember the beginning of the sentence?

그리고 textstring 을 찾아. 문장의 첫번째를 기억하니?

Best is to search case UNsensitive in the entire scope so we don't miss anything again

좋은 것은 모든 범위에서 case 구별없이 검색. 그래서 우리는 어떤 것이라도 빠뜨리지 않는다.



Found!

Doubleclick to go there

;)

Nice, we have found where the textstring is pushed on the stack. (== where the string is prepared to write it in the messagebox)

좋아, 우리는 textstring 이 stack 에 넣어지는 곳을 찾았다. (== string 은 messagebox 에 쓰기 위해 준비됐다.)

Scroll down to get a better look at it

But see already what nice stuff is here

And now have a good look around

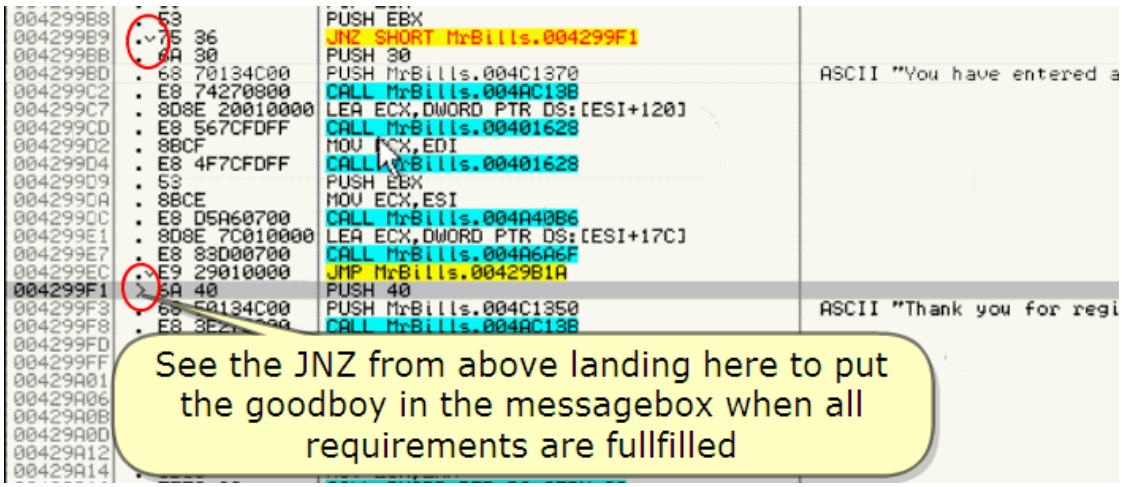
좀 더 좋게 보려면 Scroll 내려.

이미 좋은 재료가 여기 있는 것을 봤다.

그리고 이제 주변을 살펴봐.

GoodBoy

BadBoy

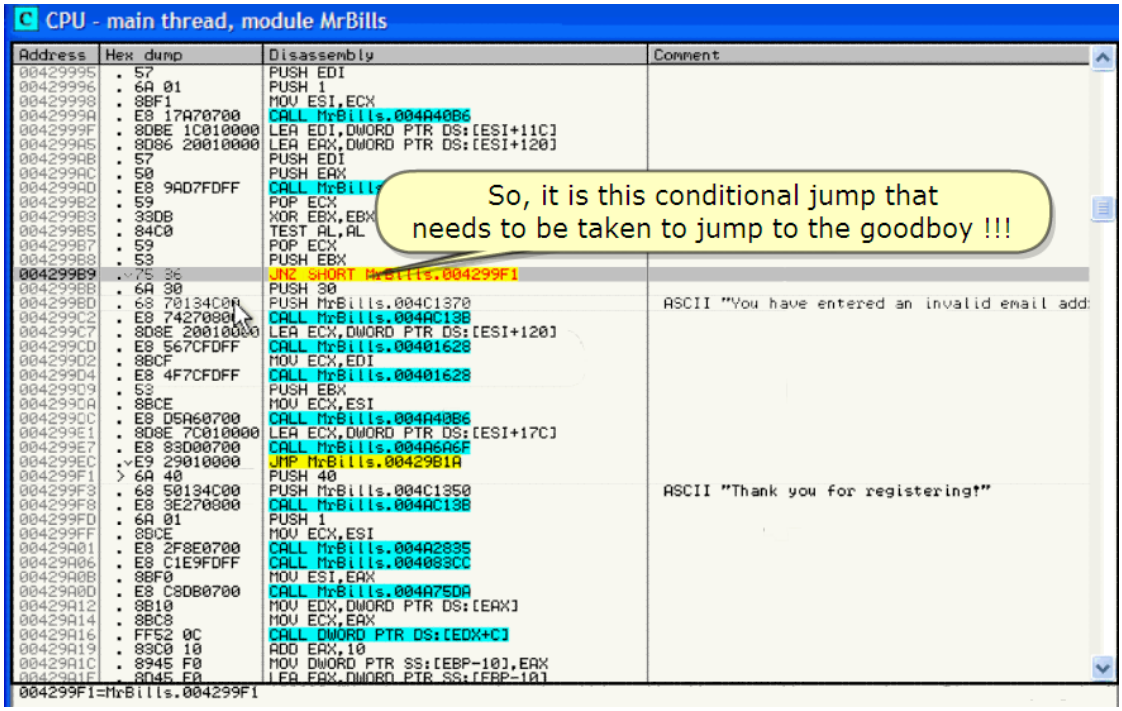


˅ 표시는 jump 가 시작하는 곳

>표시는 jump 가 도착하는 곳

See the JNZ from above landing here to put the goodboy in the messagebox when all requirements are fulfilled

JNZ 를 봐. 위에서 이곳에 도착한다. 모든 요구가 충족됐을 때 Messagebox 에 goodboy 를 넣는다.



So, it is this conditional jump that needs to be taken to jump to the goodboy !!!

그래서, 이것은 조건 jump 다. Goodboy 로 jump 하는 것이 필요하다.

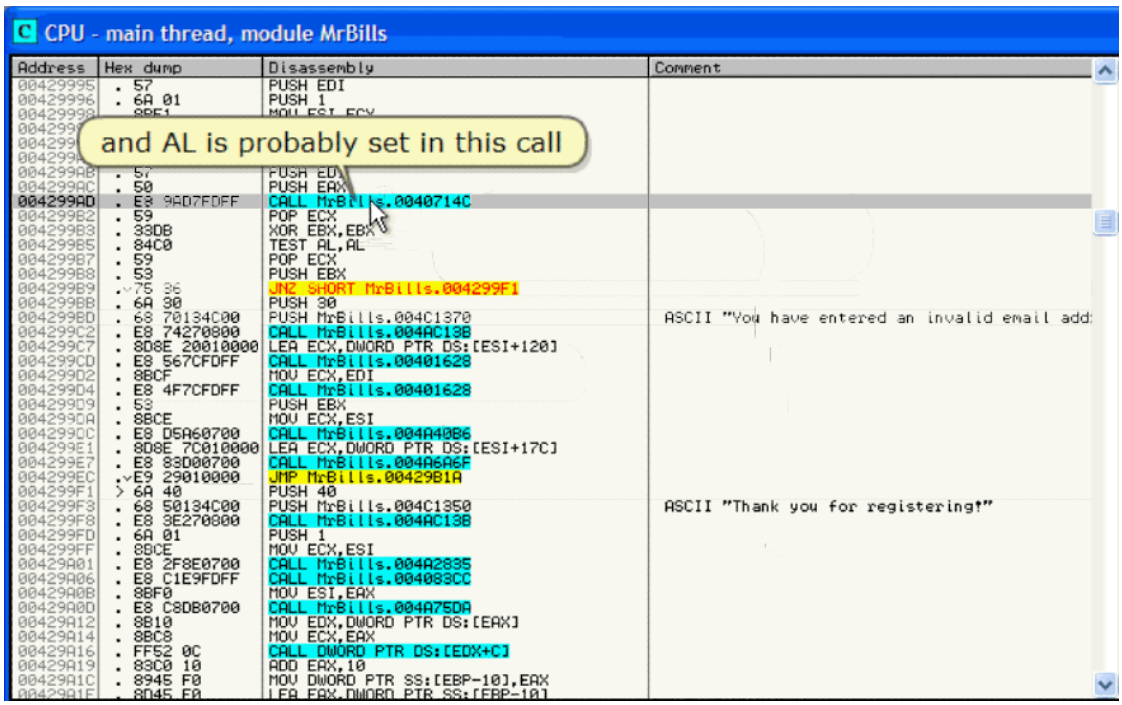
This TEST AL, AL is classic

TEST AL, AL 은 classic 하다.

;)))

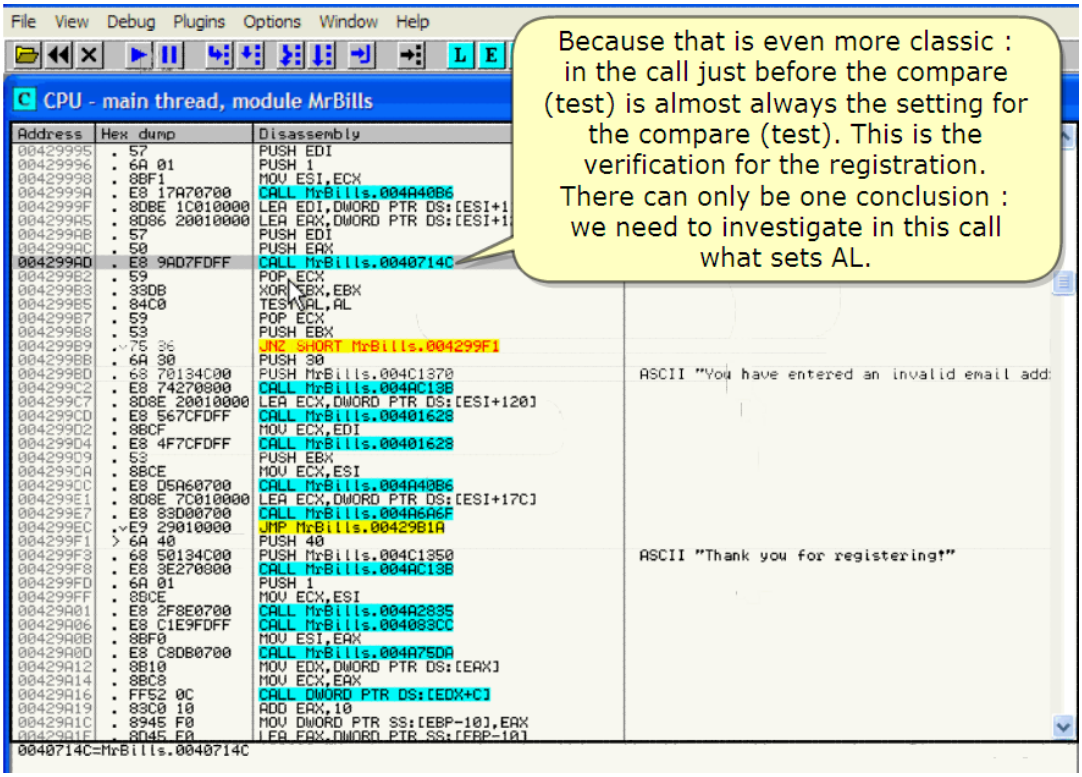
AL will decide about GoodBoy or Badboy

AL 이 Goodboy 인지 Badboy 인지에 대하여 결정할 수 있다.



And AL is probably set in this call

AL 은 아마 이 call 에서 set 된다.



Because that is even more classic : in the call just before the compare(test) is almost always the setting for the compare (test).

그것은 훨씬 더 classic 하다. : 이 call 에서 AL 이 set 되고 compare(test) 된다.

This is the verification for the registration.

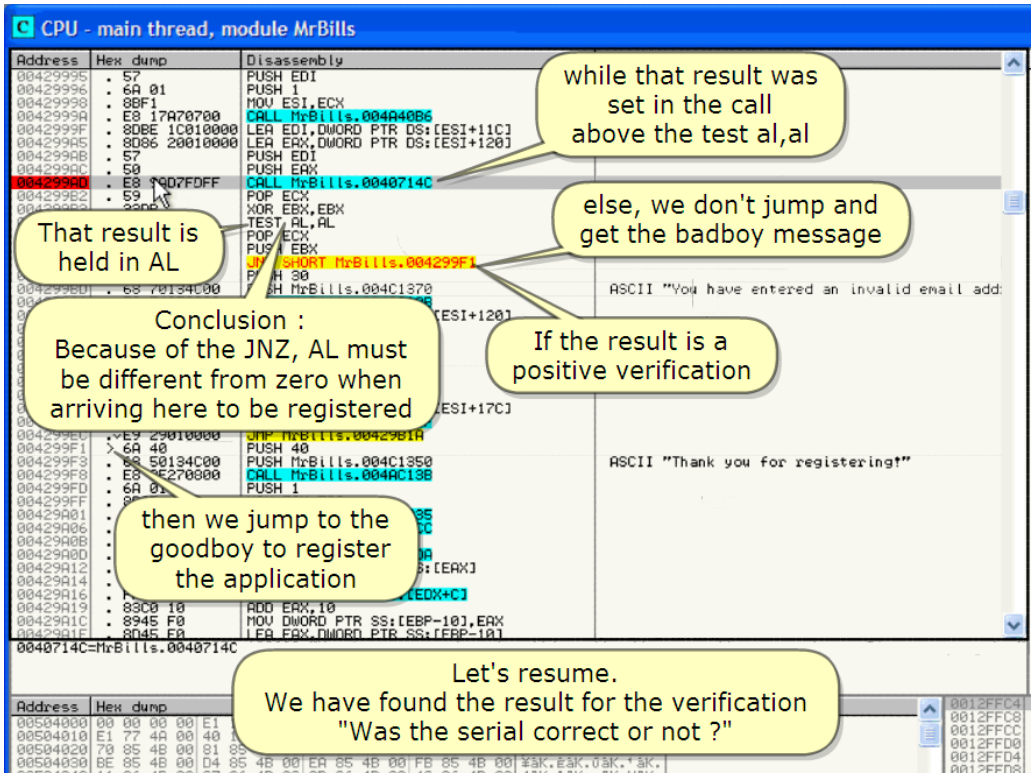
등록하기 위해 검증한다.

There can only be one conclusion : we need to investigate in this call what sets AL.

오직 한 가지 결론 : 우리는 이 call 의 어느 곳에서 AL 이 set 되는지 조사할 필요가 있다.

So, put a BP here(doubleclick opcode)

그래서, BP 를 여기에 설정 해(doubleclick opcode)



Let's resume.

We have found the result for the verification "was the serial correct or not ?"

That result is held in AL

계속하자.

우리는 검증하기 위한 결과를 찾았다. "serial 이 정확하냐 아니냐?"

result 는 AL 에서 결정됩니다.

While that result was set in the call above the test al, al

그 result 는 이 call 에서 test al, al 의 위에서 set 됐다.

If the result is a positive verification

만약에 result 가 긍정적인 검증을 하게 되면

Then we jump to the goodboy to register the application

우리는 application 에 등록하기 위해 goodboy 로 jump 한다.

Else, we don't jump and get the badboy message

아니라면, 우리는 jump 하지 않는다. 그리고 badboy message 를 얻는다.

Conclusion:

Because of the JNZ, AL must be different from zero when arriving here to be registered

결론 :

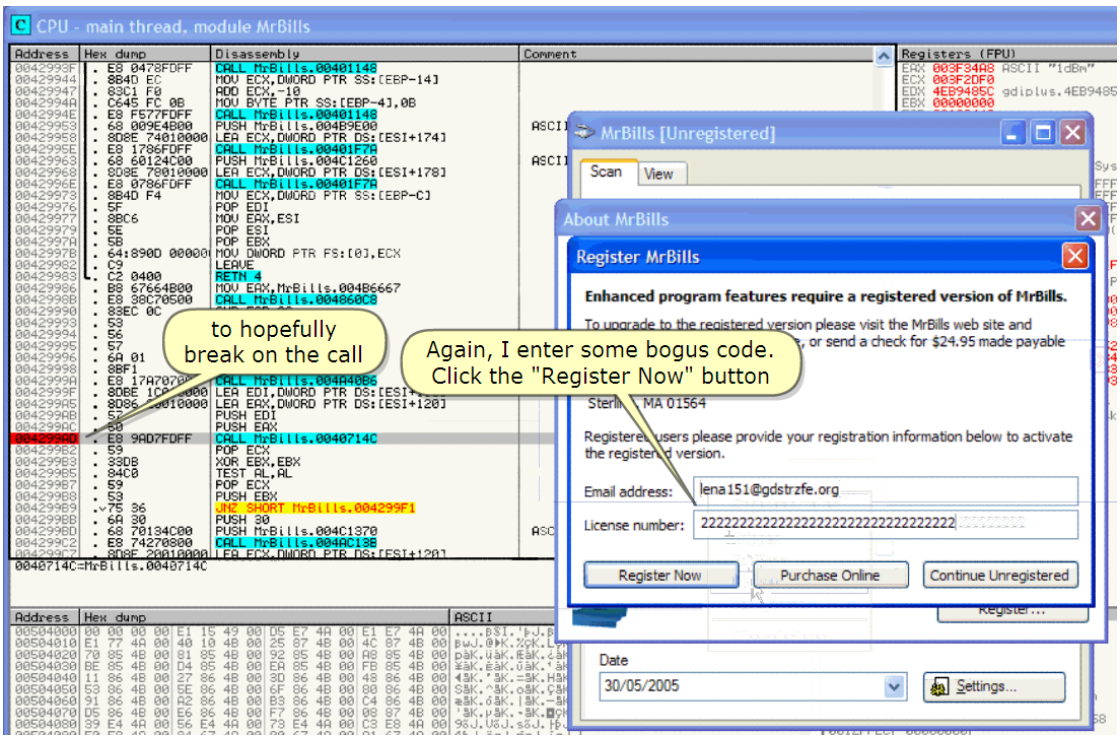
JNZ, 등록된 곳에 도착하기 위해 AL 은 zero 와 달라야 한다.

번역 주) test al, al 에 도착했을 때 al 이 0 이 아니어야 JNZ 에서 goodboy 로 jump 한다. Test 연산은 and 연산인데 결과값을 저장하지 않는 assembly 다.

Now, what is before all this?
이제, 이것 다음에 무엇이냐?

Okay, only the strings for the About box
They are not important for us here.
Okay, 오직 About box 를 위한 strings 이다.
그들은 현재 우리에게 중요하지 않다.

Let's re-run registration
등록 하기 위해 재시작 하자.
To break in the breakpoint and see what is in the call
Breakpoint 에서 멈추자. 그리고 call 안에 무엇이 있는지 보자.
Resume : the plain stupid patching method would only patch conditional jumps to jump the badboy(s). Mostly, this doesn't suffice to also register an application.
재개 : 분명히 멍청한 patching 방법은 오직 badboy 로 조건 jump 를 한다. 대부분, application 에 등록하기 위해 충분하지 않다.
So, today, we will go dig deeper in the call that sets the deciding register (here AL) to patch the setting itself into "registered".
그래서, 오늘, 우리는 call 안으로 깊게 파 보겠다. 결정하는 Register 를 patch 해서 등록하기 위해 Set 한다.



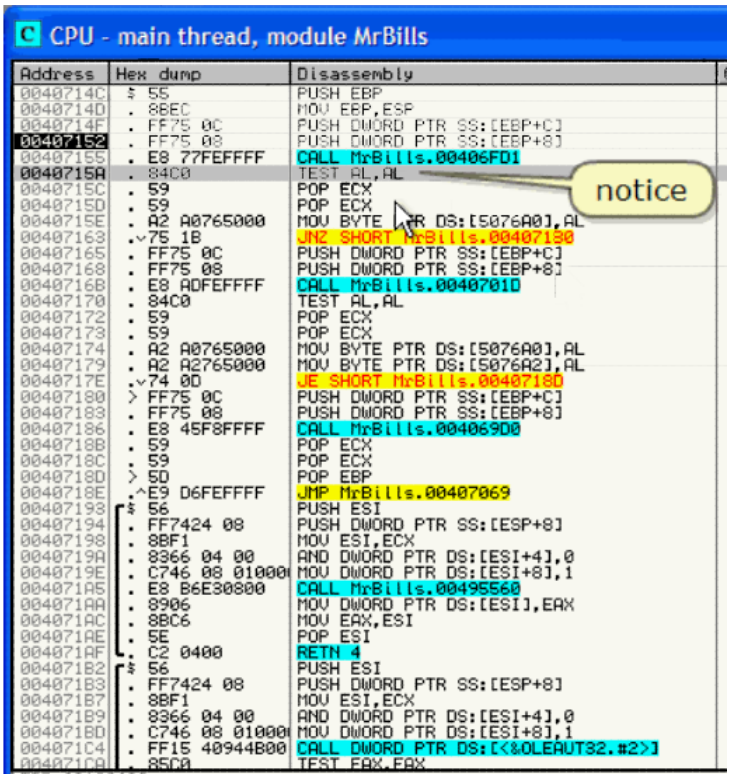
Again, I enter some bogus code.
Click the "Register Now" button

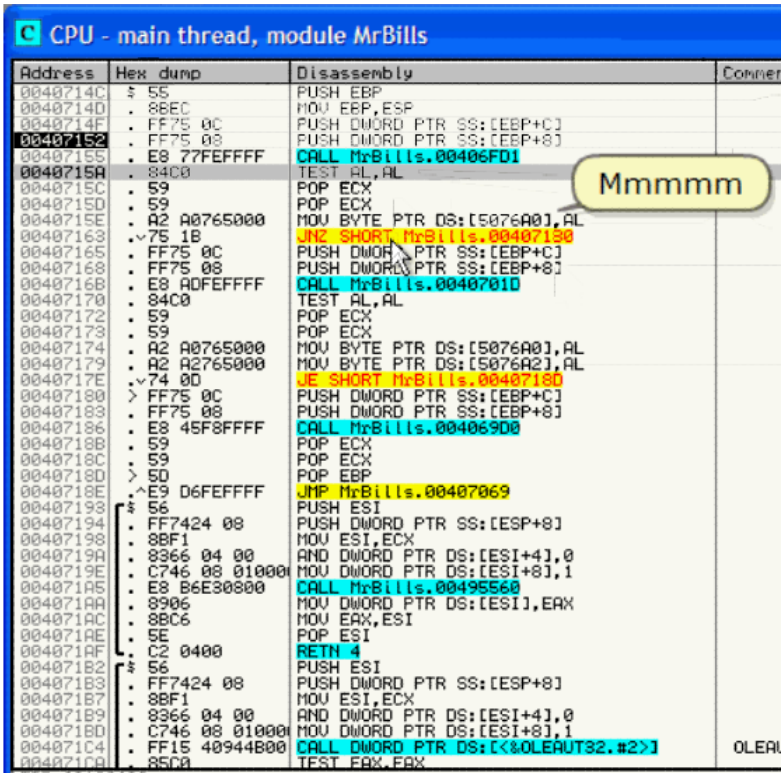
To hopefully break on the call
다시, 가짜의 code 에 들어간다.
"Register Now" button 을 click 해.
여기에서 멈추기를 바란다.

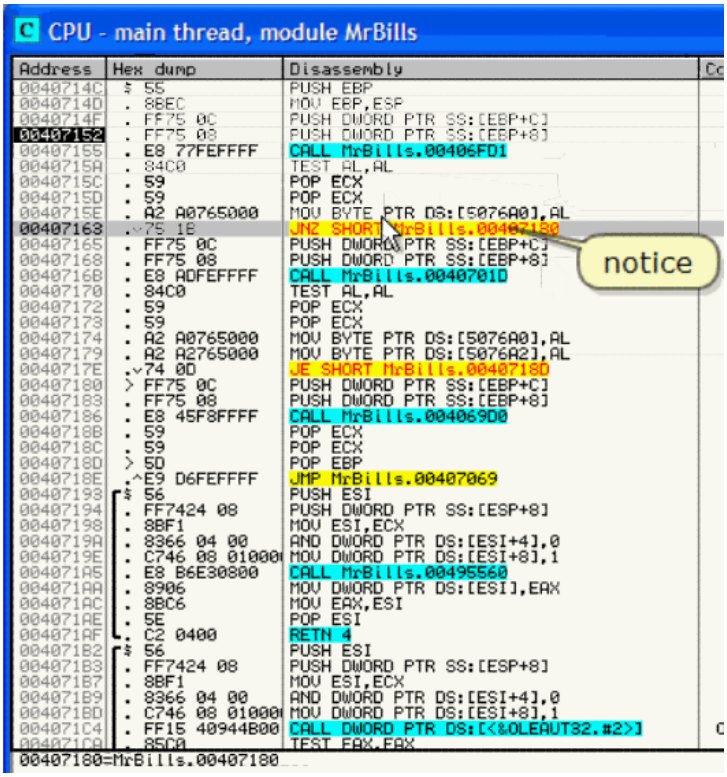
Great!
We land on the call.
Press F7 to step into
좋아!
우리는 call 에 도착했다.
F7 을 눌러 step 안으로 들어가자.

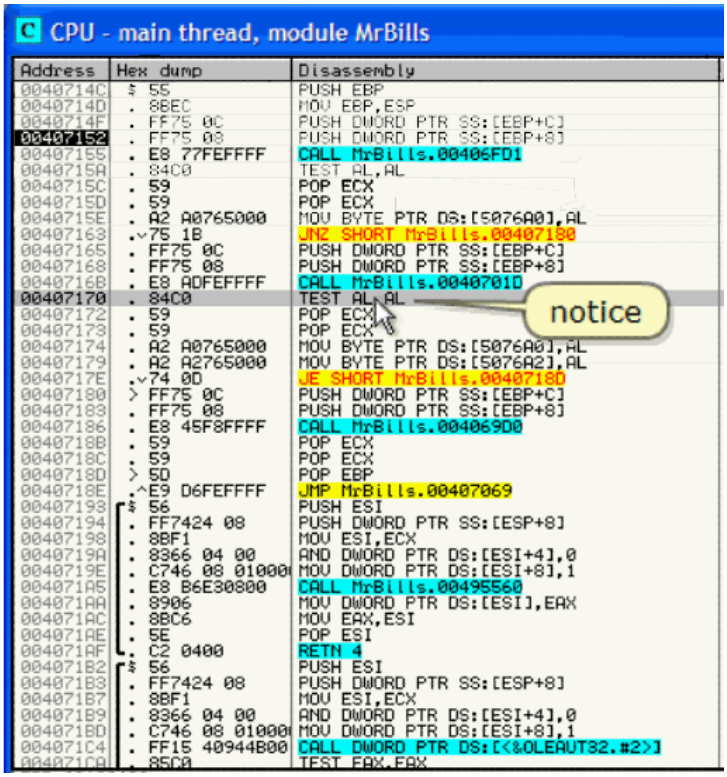
We land here in the call
Step with F8 now to take an overview of what's all happening
우리는 call 안에 도착했다.
F8 을 눌러. 어떤 일이 일어나는지 관점을 가져.

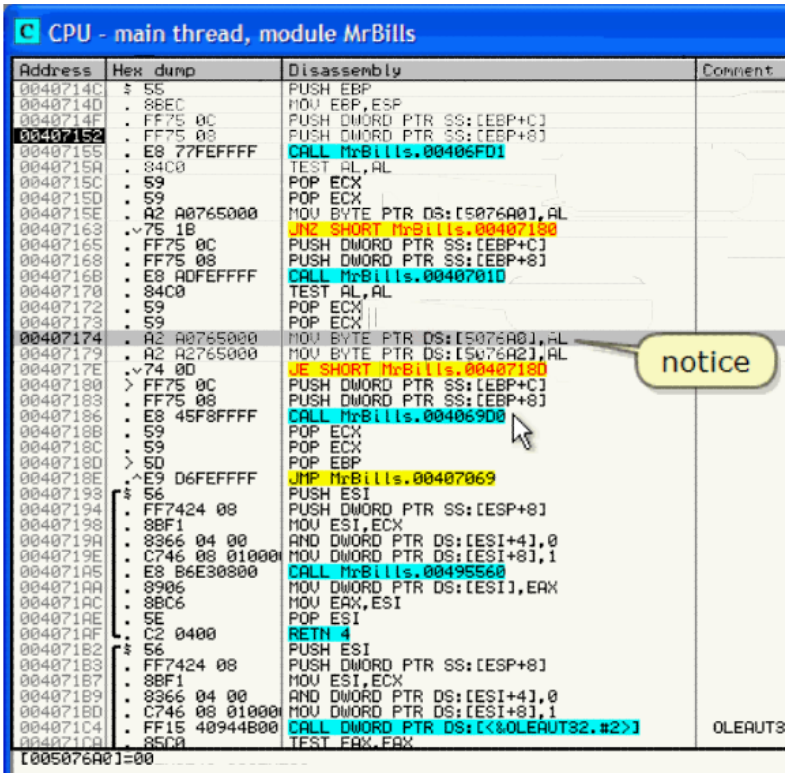
And I think it's clear we need to keep an eye on the value for AL
그리고 이것이 명확하다. 우리는 AL value 을 유지하기 위해 필요하다.

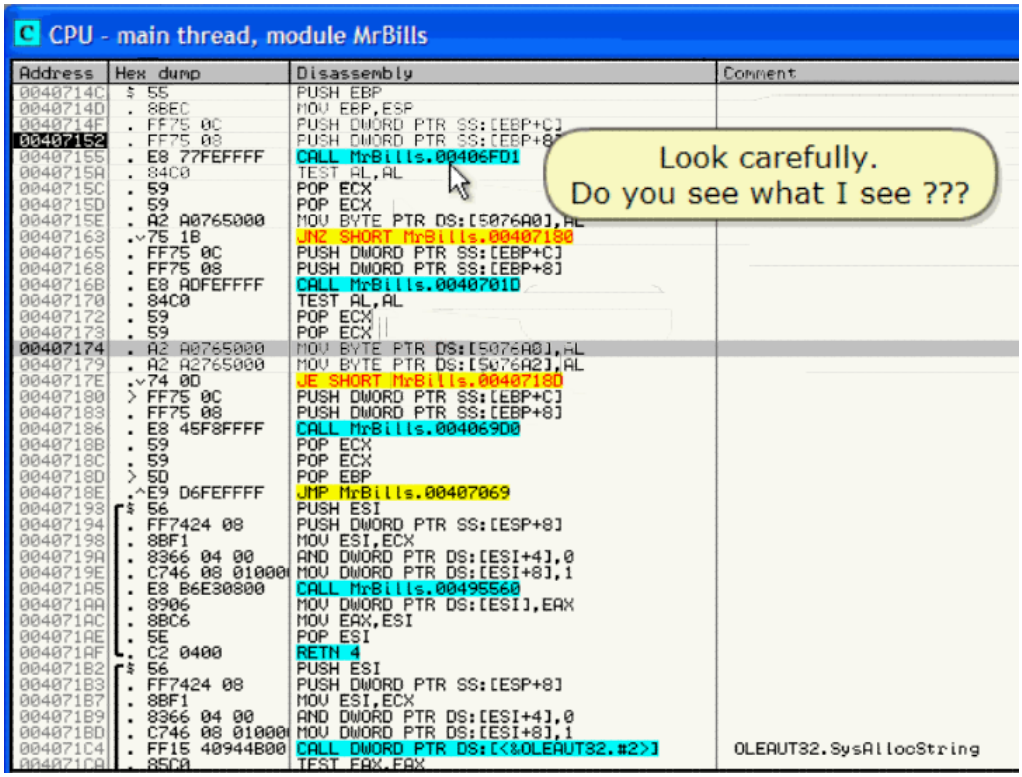



CPU - main thread, module MrBills
Mmmmm


CPU - main thread, module MrBills
notice
00407180=MrBills.00407180


CPU - main thread, module MrBills
Address
Hex dump
Disassembly
00407152
00407170
notice


CPU - main thread, module MrBills
Address
Hex dump
Disassembly
Comment
00407152
00407174
notice
OLEAUT3
[005076A0]=00



But soon we see important stuff

Notice

Mmmmm

Notice

Notice

Notice

Look carefully.

Do you see what I see ???

그러나 우리는 곧 중요한 재료를 본다.

내가 무엇을 봤는지 봤어 ???

It seems ...

AL is already set in this call

AL 은 이미 이 call 에서 set 됐다.

So, let's enter the call to see what it has to offer (push <enter>)

Code 안으로 (PUSH <enter>)를 누가 제공하는지 보기 위해 들어가자.

BTW, when pressing <enter> you can follow the code without effectively executing the code.

<enter>를 누를 때, 효과적인 실행 code 가 없이 code 를 따라갈 수 있다.

So, we go now see the code in the call without executing it

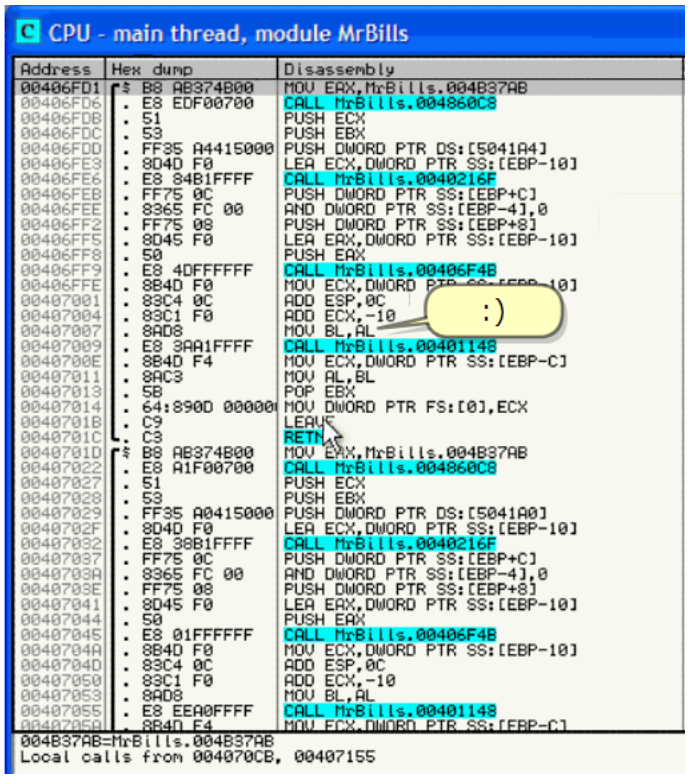
그래서, 우리는 그것을 실행하지 않고 code 를 보기 위해 call 안으로 들어간다.

Mmmm, let's see ....

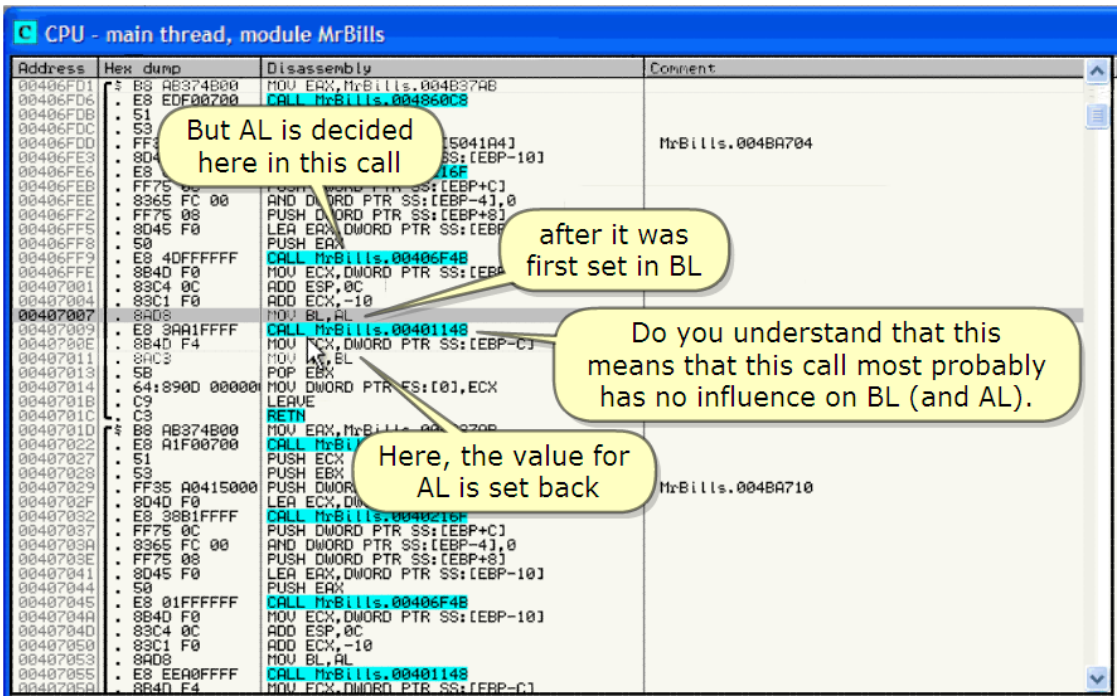
Study the code here

음, 보자....

Code 를 공부하자.



:)

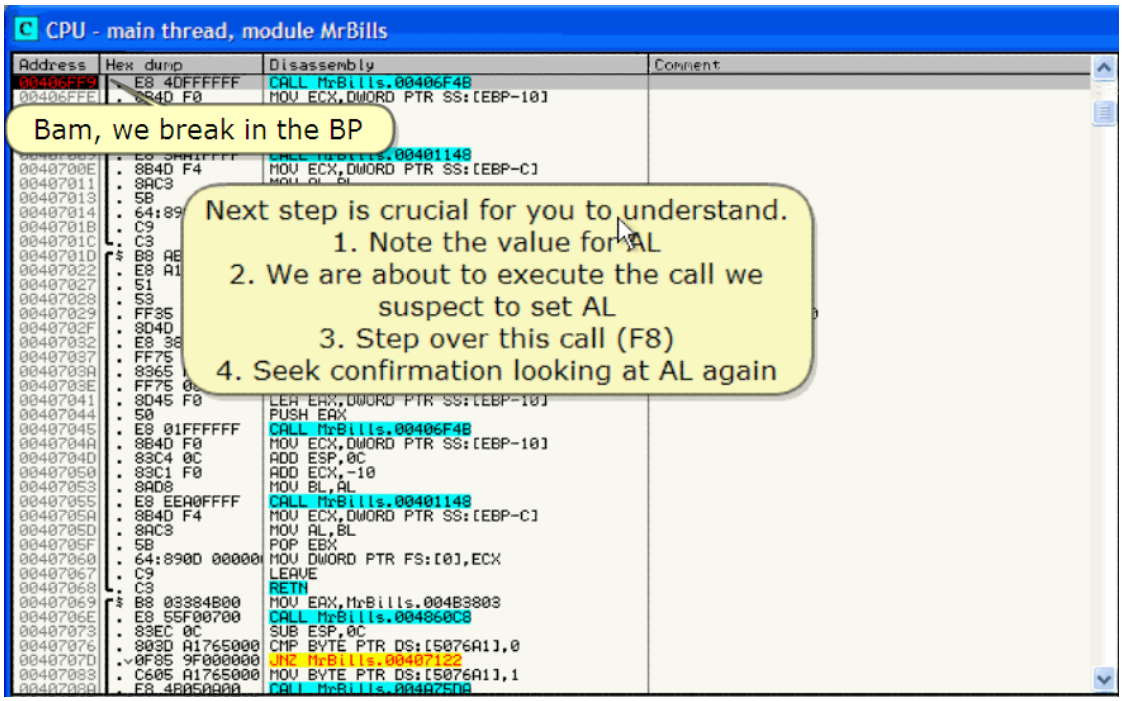


Here, the value for AL is set back

이곳, AL value 가 set 되고 돌아온다.
After it was first set in BL
먼저 BL 이 set 된 후에
Do you understand that this means that this call most probably has no influence on BL (and AL).
이 뜻은 이 call 은 아마 BL 에 영향이 없다. 이해했어?(and AL)
But AL is decided here in this call
그러나 AL 은 이 call 에서 결정된다.

Right, AL is set in the call just above, so put a BP
맞아, AL 은 이 call 의 위에서 set 된다. 그래서 BP 를 넣어라.

Remember that we are a couple of calls deep in the code yet.
기억해. 우리는 이 code 에서 call 의 couple 이 깊어진다. 아직
Also remember that we pressed <enter> to come see the code in this call.
또한 기억해. 우리가 이 call 에서 code 를 보기 위해 enter 를 넣었다.
To verify if AL is really set here, we will run till breaking in this BP. Press F9
검증하자. 만약에 AL 이 정말로 set 됐다면, 우리는 BP 가 걸린 곳까지 run 할 수 있다. F9 눌러.



Bam, we break in the BP
우리는 BP 에서 멈출 수 있다.
Next step is crucial for you to understand.
다음 step 은 너에게 이해하는데 있어 중요하다.
1.Note the value for AL
1.AL 을 적어라.

2.We are about to execute the call we suspect to set AL

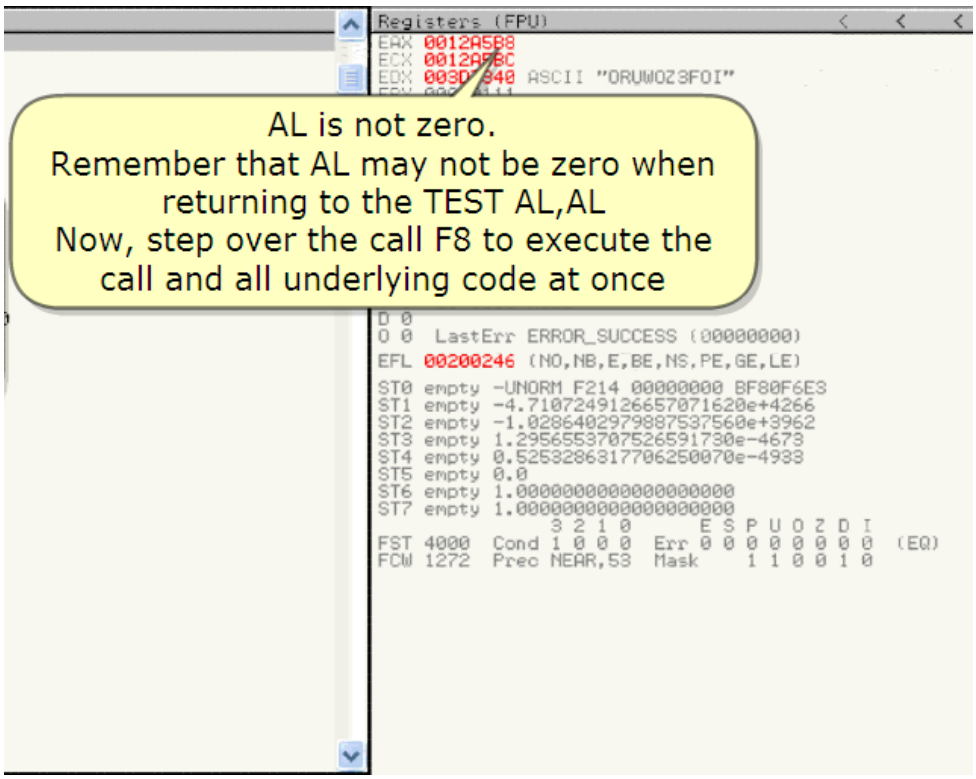
2.우리는 call 을 실행하기 위한 것이 있다. AL 을 set 하기 위해 의심한다.

3.Step over this call(F8)

3.Step 을 넘기기 위해 F8 을 눌러.

4.Seek confirmation looking at AL again

4.AL 을 보고 다시 확인하고자 합니다.



Al is not zero.

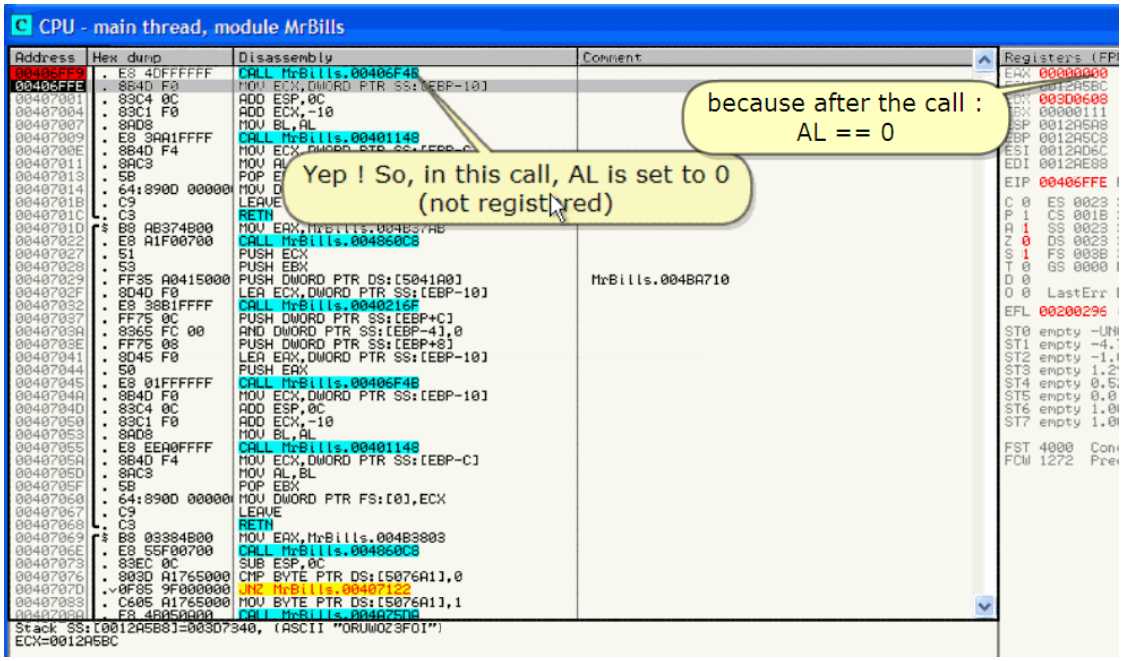
AL 은 zero 가 아니다.

Remember that AL may not be zero when returning to the TEST AL, AL

기억해. TEST AL, AL 을 하고 난 후에 AL 은 zero 가 되지 않을 것이다.

Now, step over the call F8 to execute the call underlying code at once

이제, F8 을 눌러 한 번에 근본적인 call 을 실행 시켜보자.



Yep! So, in this call, AI is set to 0 (not registered)

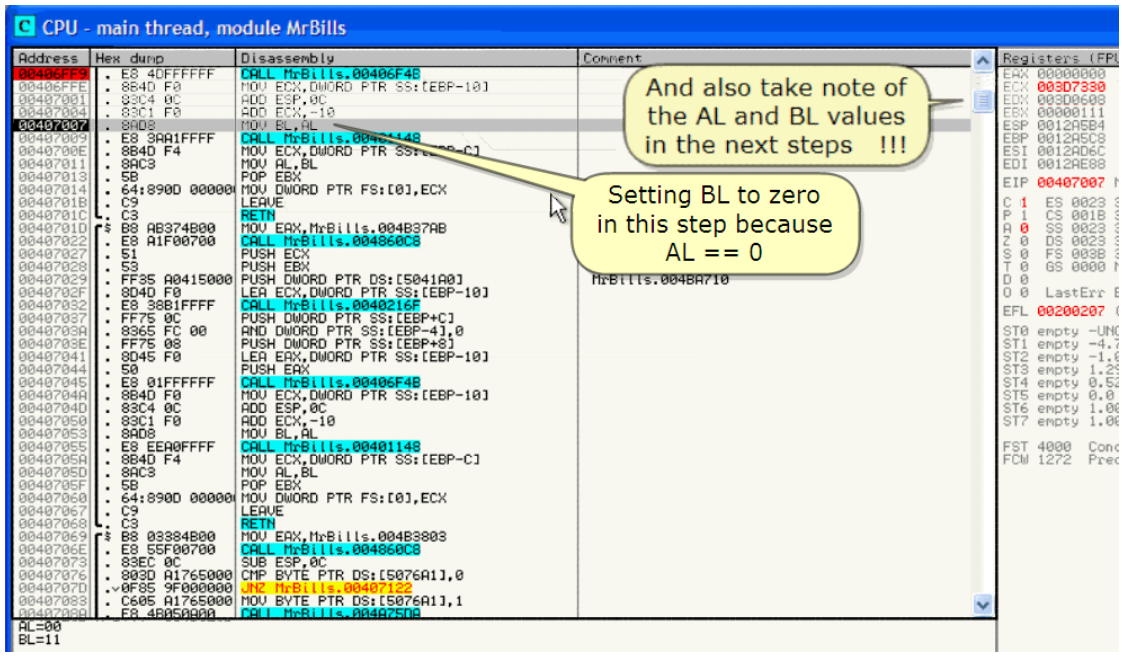
Because after the call : AL == 0

예! 그래서 이 call 에서 al 은 0 으로 set 된다.(미등록)

왜냐하면 call 후에 : AL == 0 이기 때문이다.

F8 to see what happens further

무슨 일이 생기는지 F8 을 눌러서 보자.

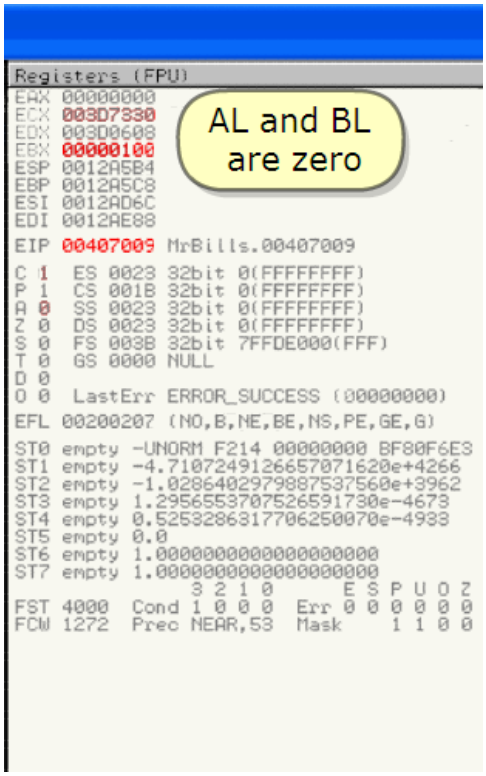


And also take note of the AL and BL values in the next steps !!!

AL 과 BL 값을 적어놓자.

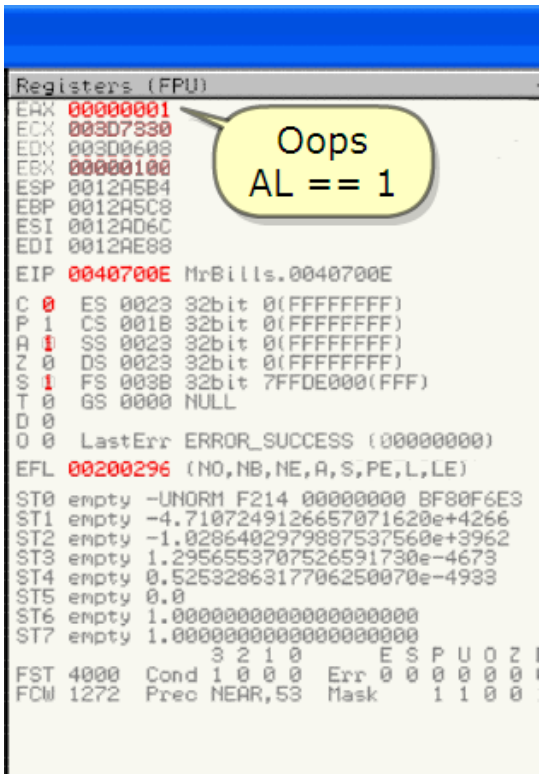
Setting BL to zero in this step because AL == 00

이 step 에서 BL 값을 zero 로 setting 한다. 왜냐하면 AL == 00 이기 때문이다.



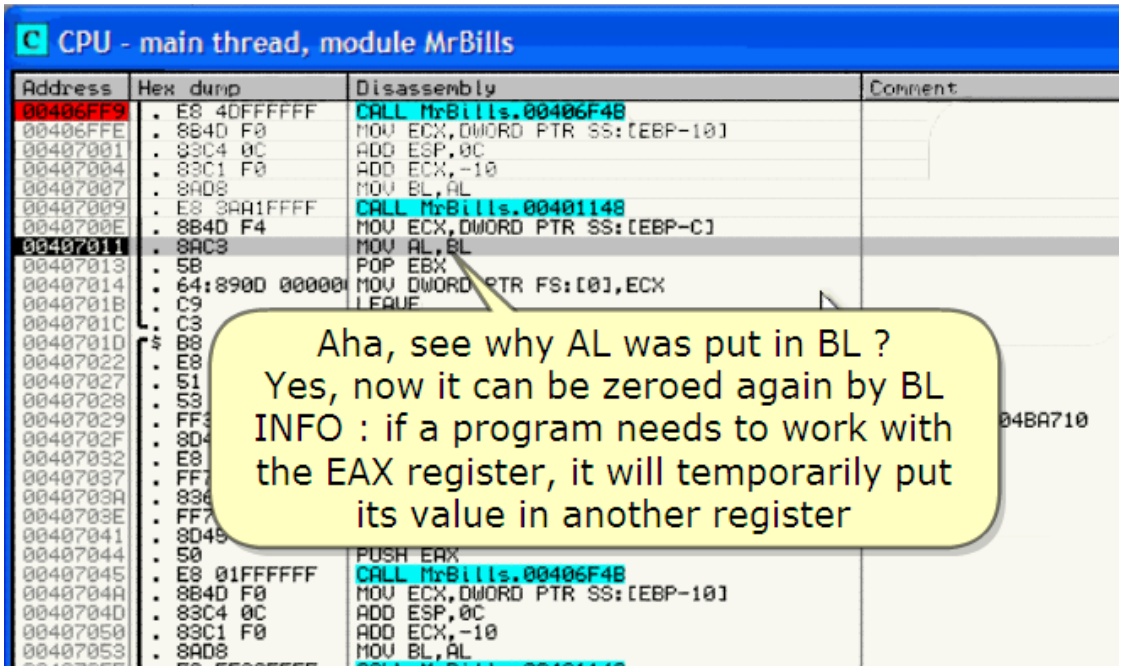
AL and BL are zero

AL and BL 은 zero 다.



Oops AL == 1

웁스 AL == 1 이다.



Aha, see why AL was put in BL?

아하, 왜 BL 이 AL 에 넣어졌나?

Yes, now it can be zeroed again by BL

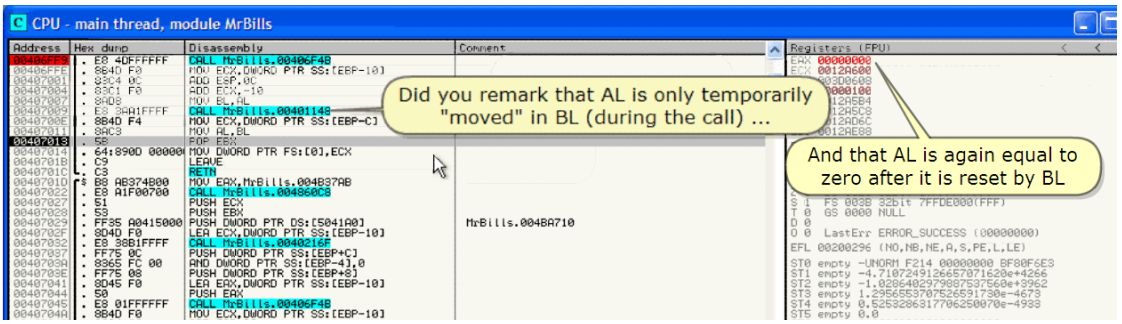
예, 다시 BL 에 의해 zero 가 될 것이다.

INFO : if a program needs to work with the EAX register, it will temporarily put its value in another register

정보 : 만약에 program 이 EAX register 와 함께 일하는 게 필요하다면, 그것은 임시적으로 그것의 값이 다른 register 에 넣어질 것이다.

I'll explain you once again, so you get the feeling of how a program works

다시 한 번 너에게 설명하겠다. 그래서 너는 program 이 어떻게 일하는지 느낌을 얻을 수 있다.

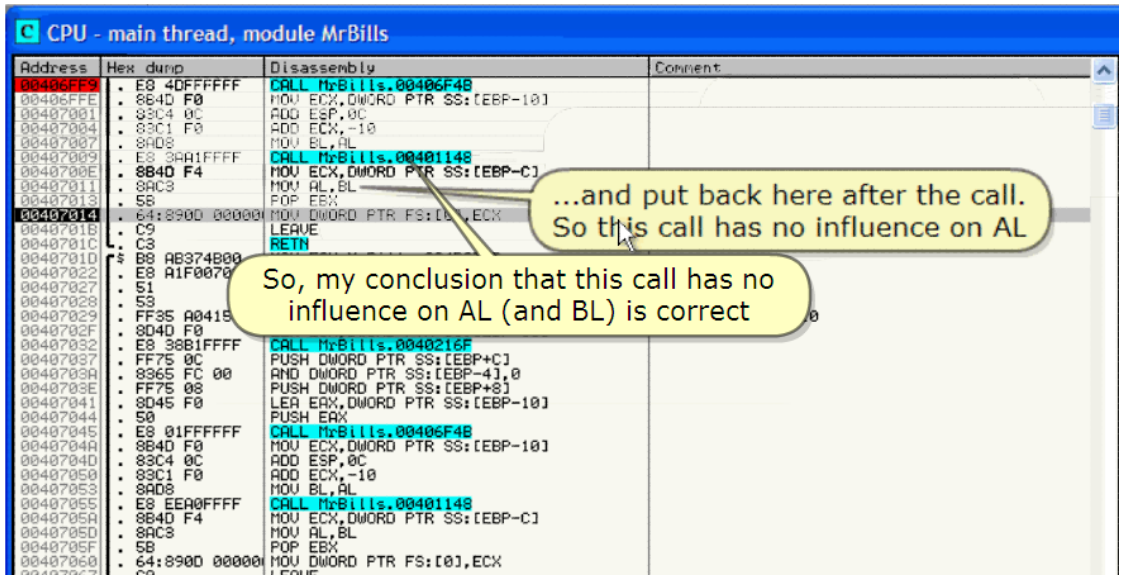


Did you remark that AL is only temporarily "moved" in BL (during the call) ...

주목해. AL 은 임시적으로 BL 로 "moves" 했다. (call 을 하는 동안)

And that AL is again equal to zero after it is reset by BL

그리고 BL 에 의해 reset 되어진 후에 AL 은 다시 zero 와 같아졌다.



...and put back here after the call.

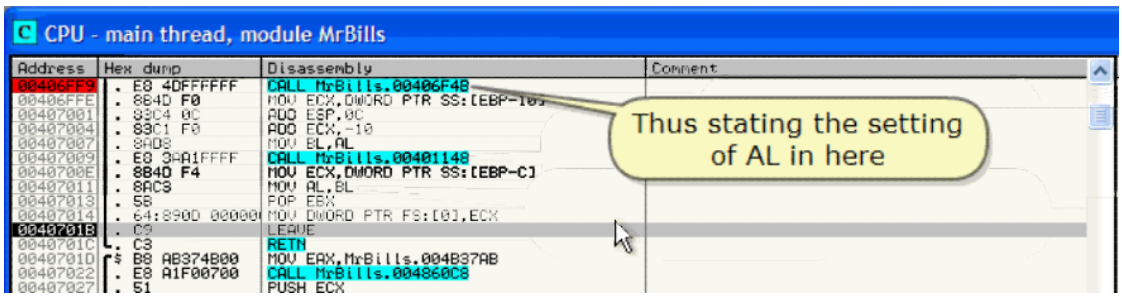
Call 후에 여기에 돌려준다.

So this call has no influence on AL

이 call 은 AL 에 영향을 주지 않는다.

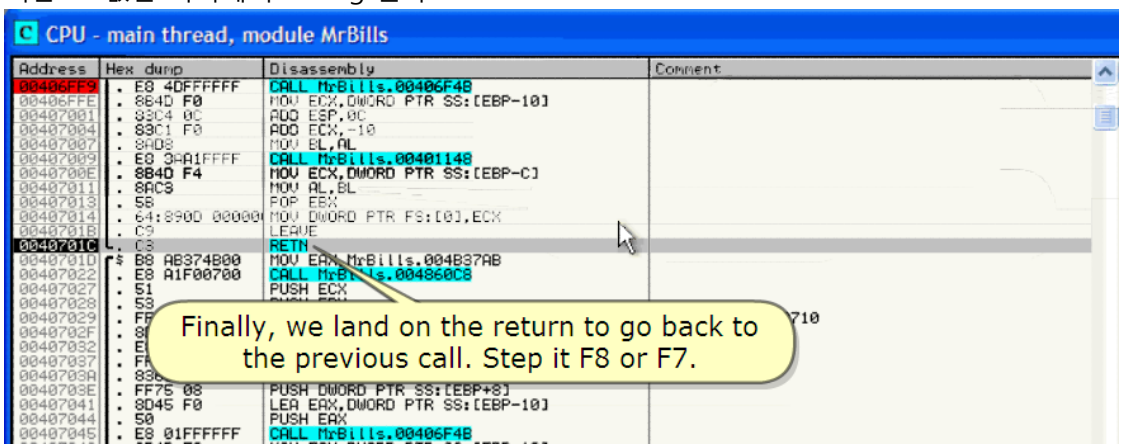
So, my conclusion that this call has no influence on AL (and BL) is correct

그래서, 나의 결론 : 이 call 은 AL 에 정확히 영향을 주지 않는다.



Thus starting the setting of AL in here

처음 AL 값은 여기에서 setting 된다.



Finally, we land on the return to go back to the previous call.

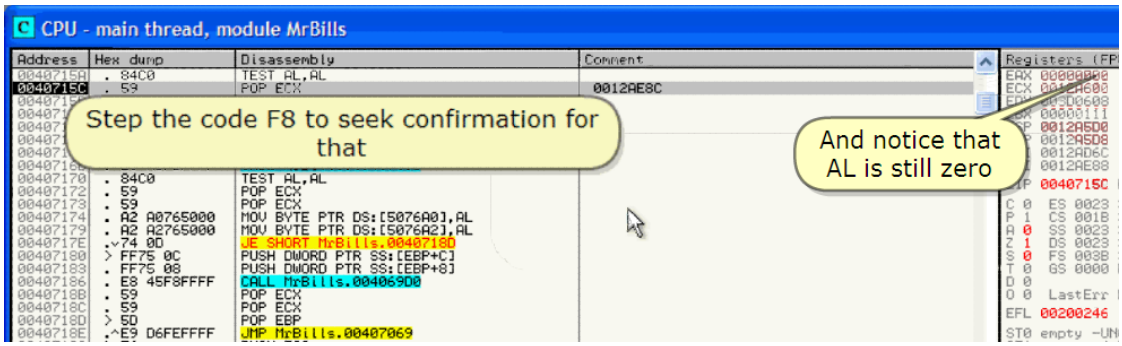
마지막으로, 우리는 이전의 call 로 돌아가기 위한 return 값에 도착한다.

Step it F8 or F7.
F8 이나 F7 을 눌러.

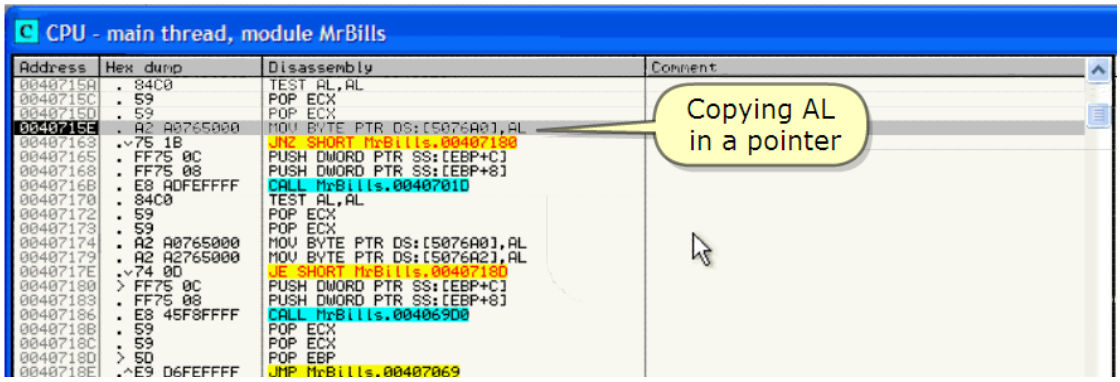
Ok. And so we are back in this call.
Ok. 우리는 call 에서 돌아왔다.
Remember that AL may not be equal to zero when we return to the TEST AL, AL (JNZ to registered or not)
기억해. 우리가 돌아와서 TEST AL, AL 을 할 때(JNZ 등록됐는지 아닌지) AL 은 zero 와 같아지지 않는다.

Now, study hits code here.
What happens with AL here?
이제, 여기 code 를 공부하자.
이 AL 에 무슨 일이 일어났나?

Mmmm, this here is all obvious : AL is never changed, but quite some stuff is carried out according to AL being 1 or 0 (=registered or not)
음, 이것은 명확하다 : AL 은 절대 바뀌지 않았다, 그러나 꽤 많은 재료들은 AL 이 1 이나 0 에 따라 움직였다. (= 등록되거나 아니거나)

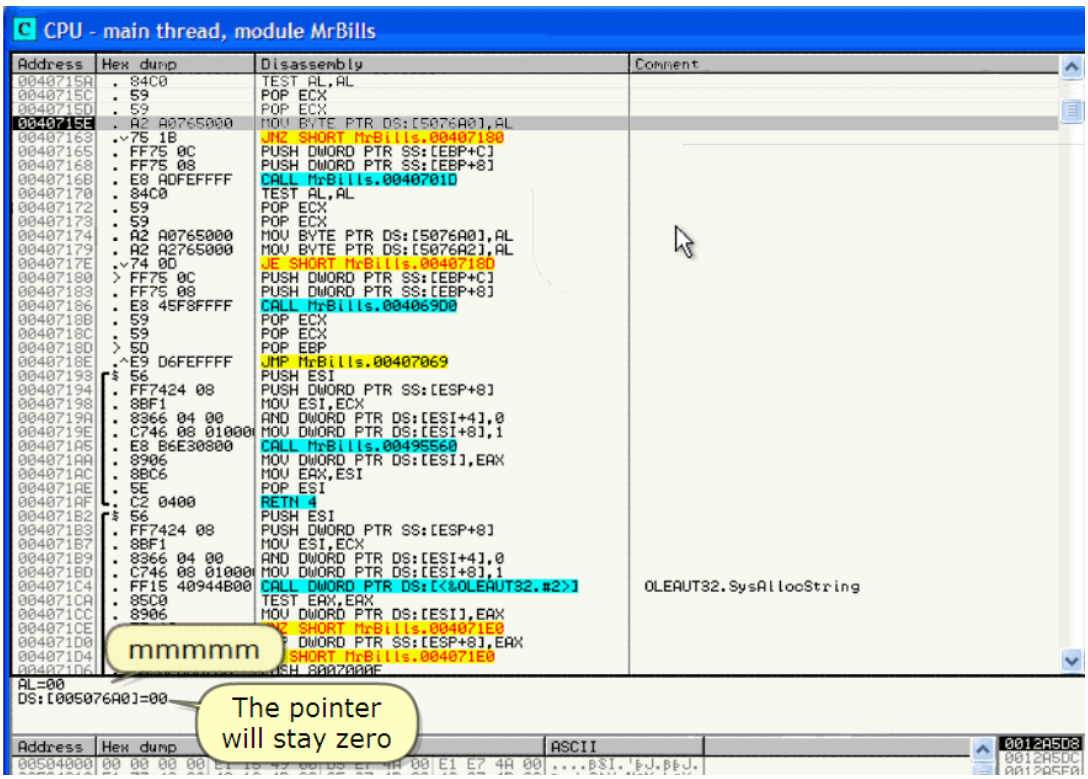


Step the code F8 to seek confirmation for that
And notice that AL is still zero
F8 을 눌러 좀 더 확인하자.
AL 은 여전히 zero 다.



Copying AL in a pointer

AL 를 pointer 에 복사했다.

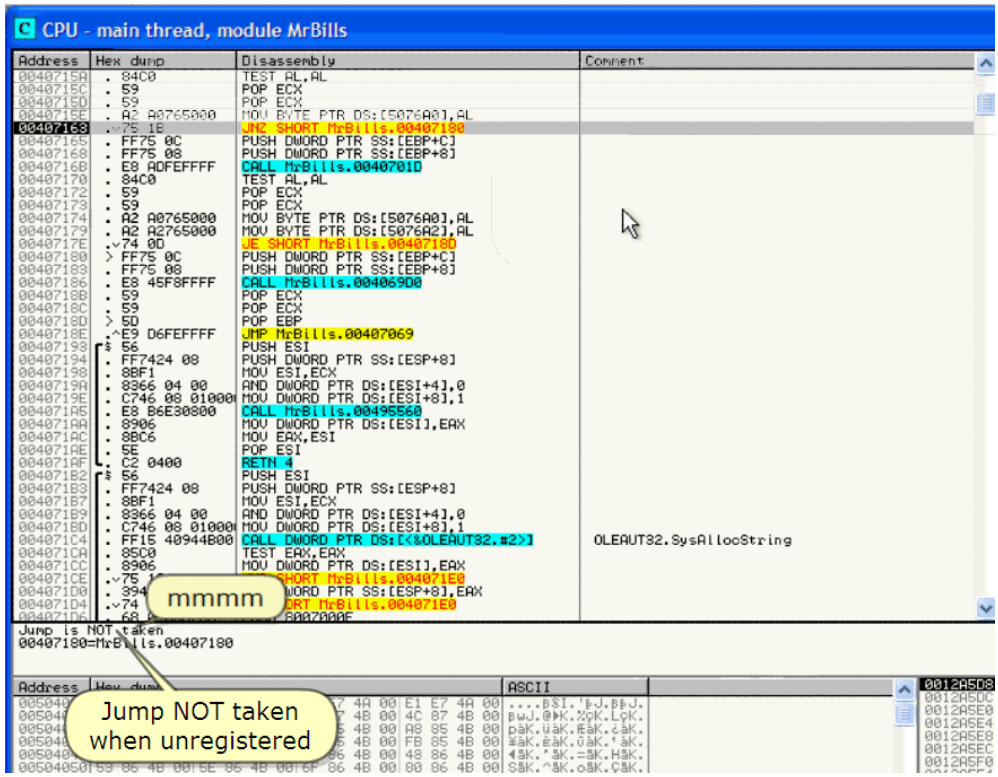


Mmmm

The pointer will stay zero

음

Pointer 는 zero 를 유지할 것이다.



Mmmm

Jump NOT taken when unregistered

음

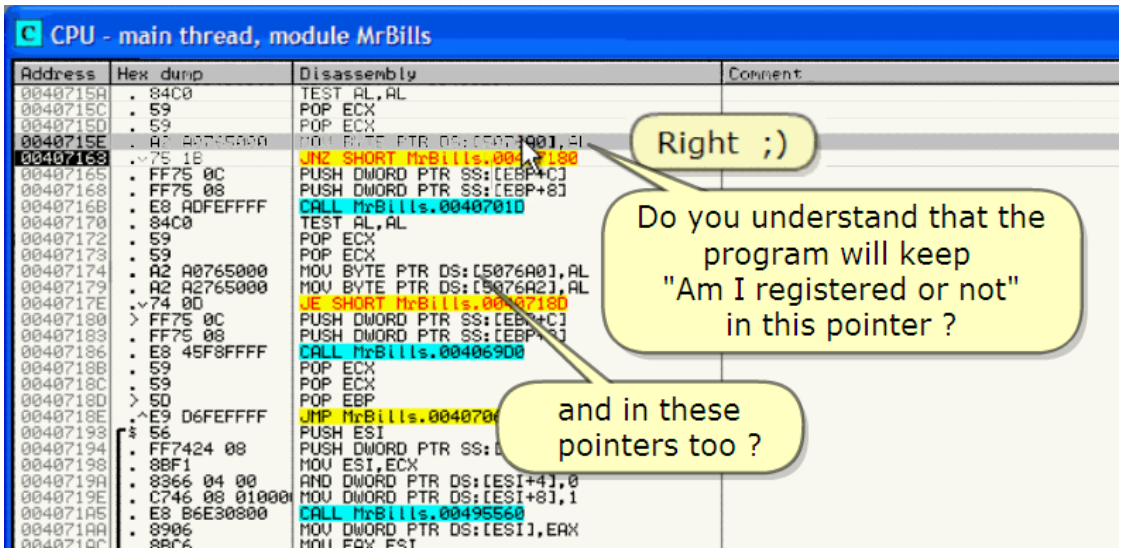
미등록 되었을 때 jump 하지 않는다.

Right ;)

Do you understand that the program will keep "Am I registered or not" in this pointer?

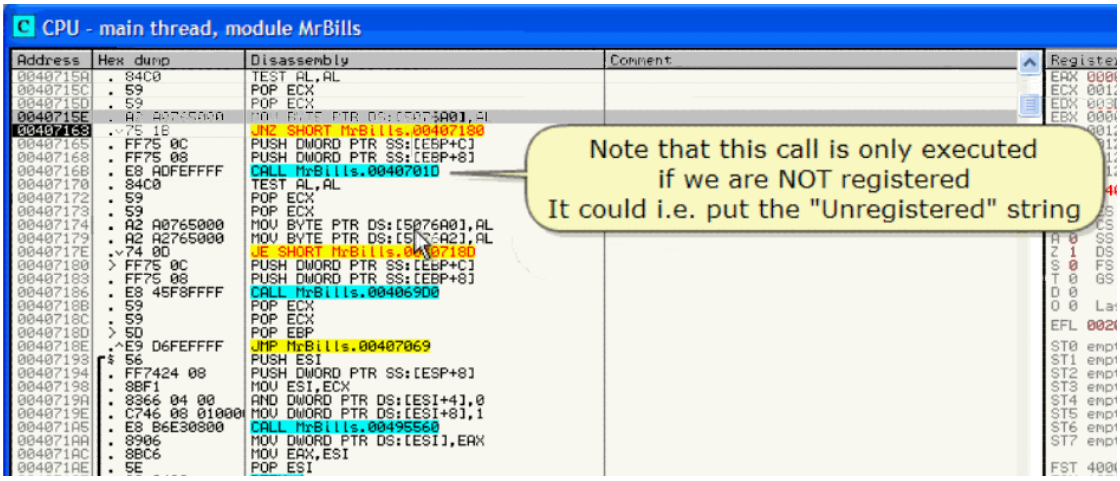
맞아 ;)

Program 은 "등록됐는지 아닌지" 이 pointer 에서 유지할 수 있다는 것을 이해했어?



And in these pointers too?

그리고 이것들도 마찬가지지?



Note that this call is only executed if we are NOT registered

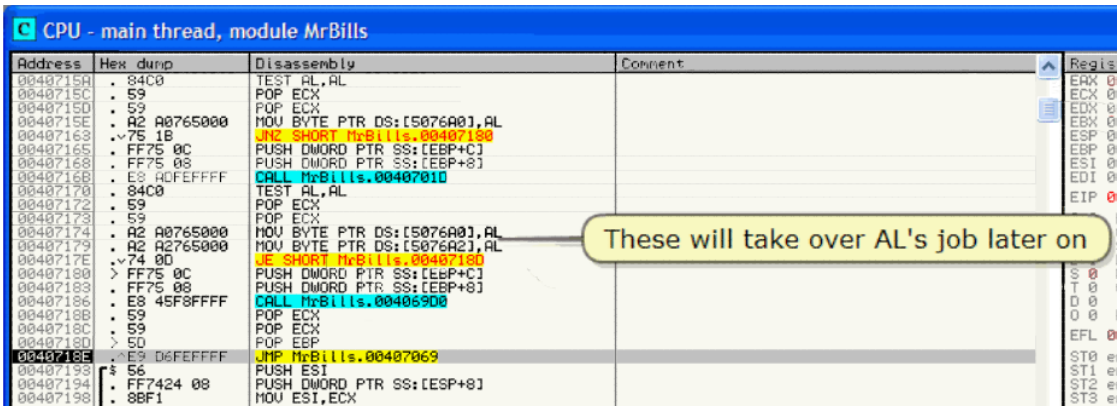
It could i.e. put the "Unregistered" string

등록되지 않았을 때 이 call 은 오직 실행됐다.

"Unregistered" string 을 넣을 것이다.

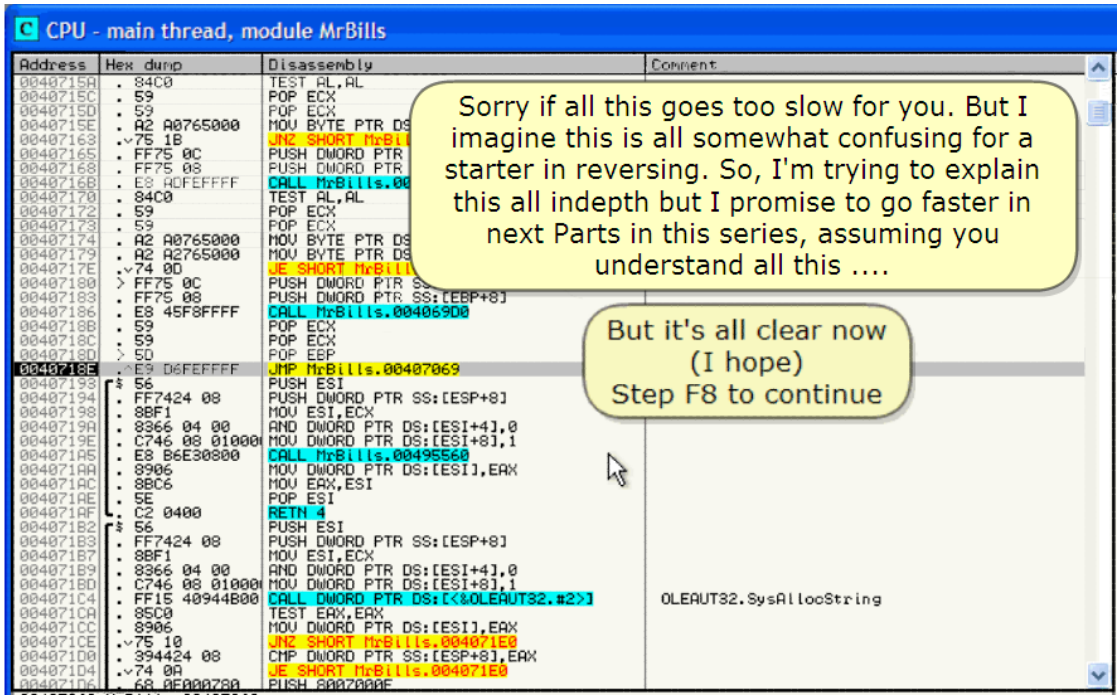
Now, let's step the code F8

F8 을 눌러서 계속 해보자.



These will take over AL's job later on

이것들은 AL job 을 물려받는 것이다.



But it's all clear now(I hope) Step F8 to continue

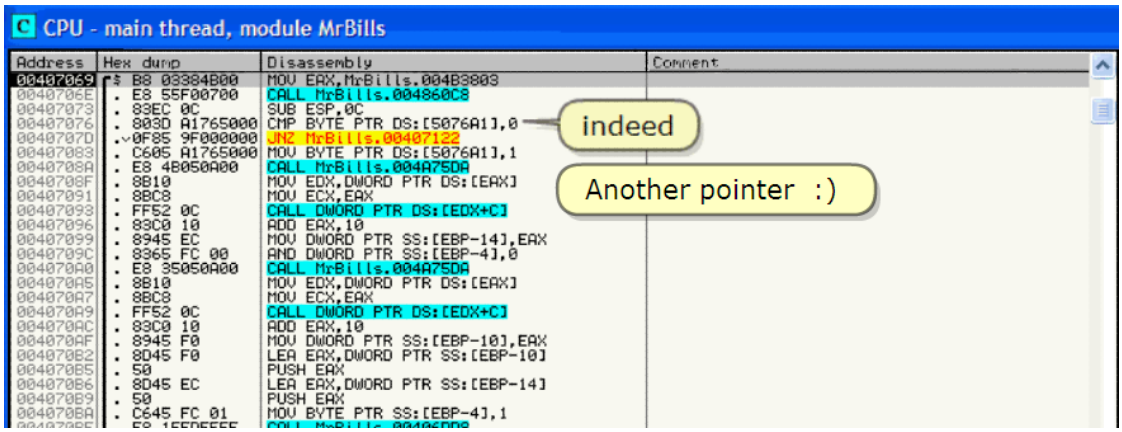
이것은 모두 명확하다. 이제 F8 을 눌러 계속하자.

Sorry if all this goes too slow for you. But I imagine this is all somewhat confusing for a starter in reversing.

미안. 모든 것이 너에게 매우 느려졌다. 이것은 reversing 초보자에게 혼란을 준다고 생각한다.

So, I'm trying to explain this all indepth but I promise to go faster in next Parts in this series, assuming you understand all this ....

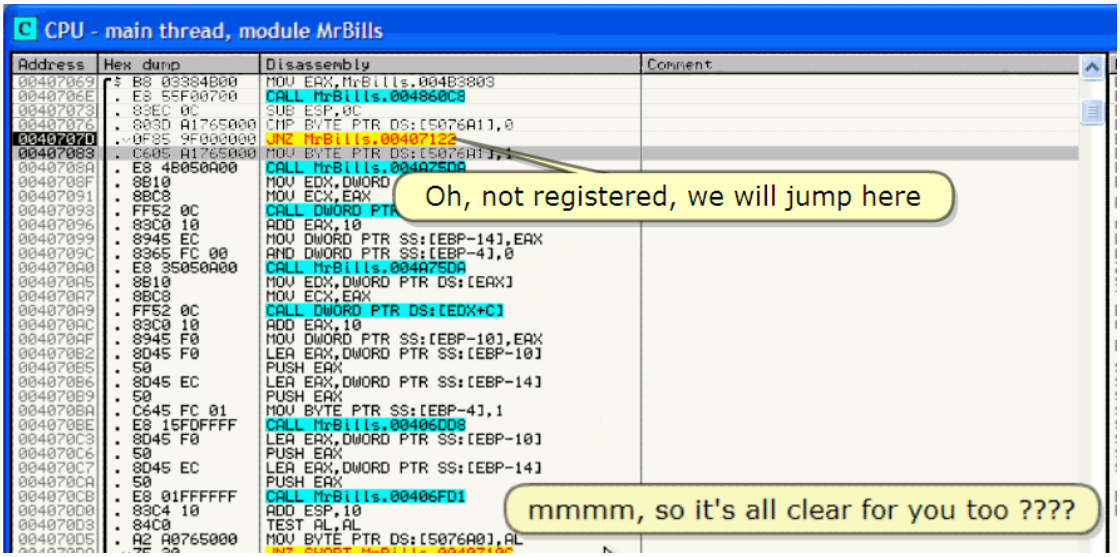
그래서, 나는 너에게 이것의 상세히 설명하려고 한다. 나는 이 series 의 다음 Part 에서 너는 빠르게 이해할 것이라고 약속한다. 네가 모든 것을 이해한다고 생각한다.



Indeed

Another pointer :)

다른 pointer 다.



Mmmm, so it's all clear for you too ????

음, 너에게 모든 것이 명확해졌어?

Oh, not registered, we will jump here

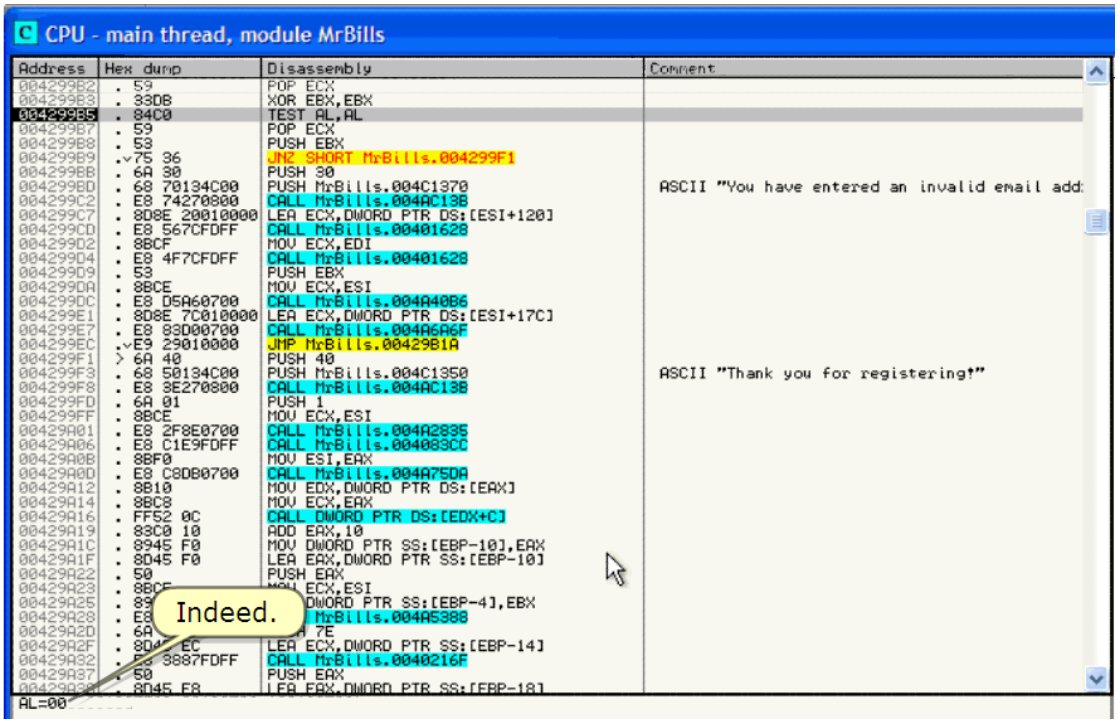
오, 미등록, 우리는 jump 할거야.

And so finally, we return back to successfully register or not.

Continue stepping

마지막으로, 우리는 성공적으로 등록했든지 아니든지 원래 자리로 돌아왔다.

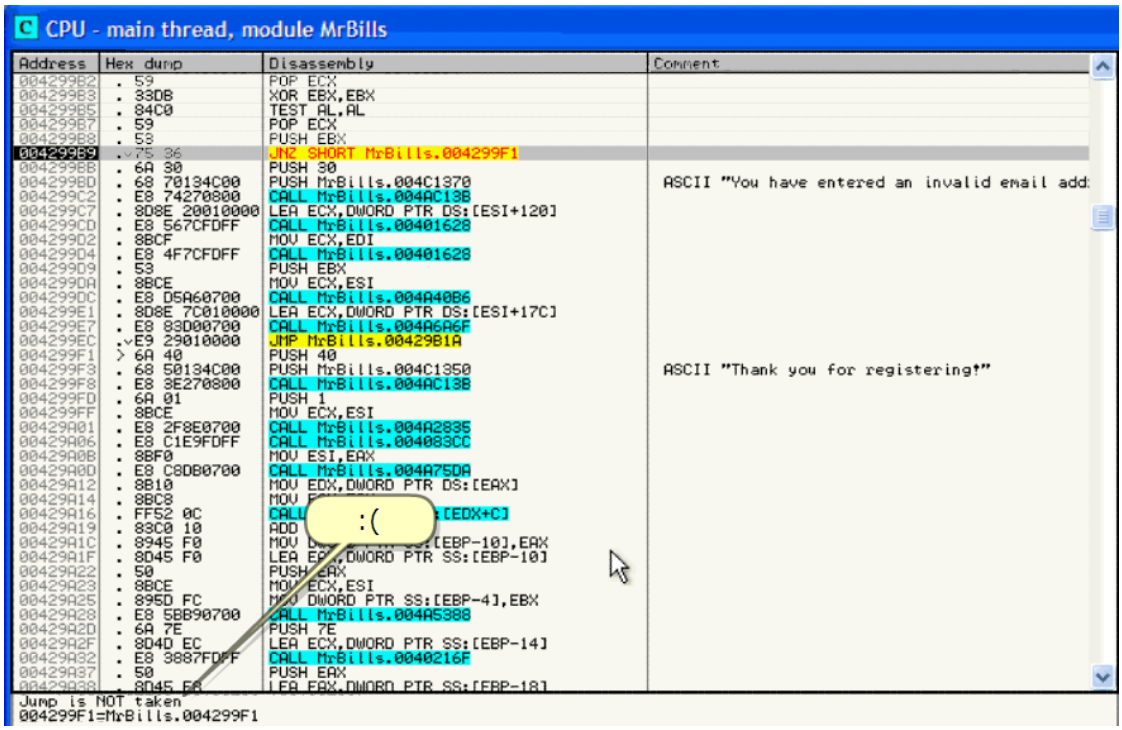
계속해보자.



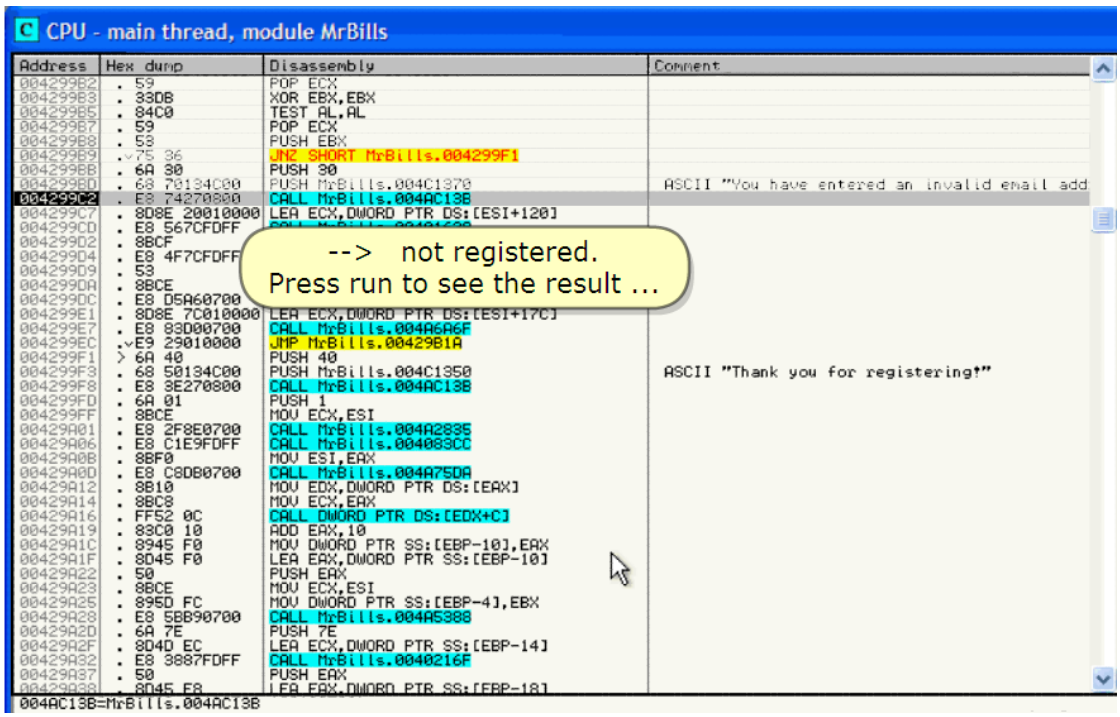
Indeed.

And so we know why we will continue to the Badboy!

왜 우리가 계속하면 Badboy 로 가는지 안다.



:(

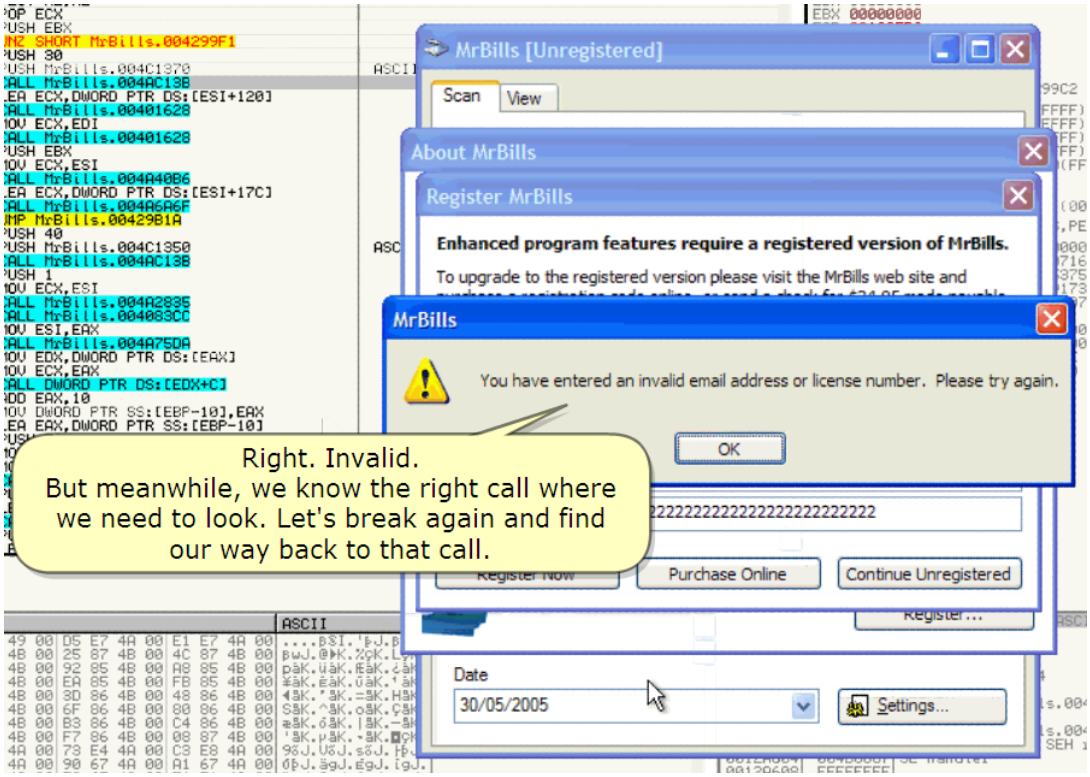


--> not registered.

Press run to see the result ...

--> 미등록

결과를 보기 위해 Run 을 눌러



Right. Invalid.

But meanwhile, we know the right call where we need to look.

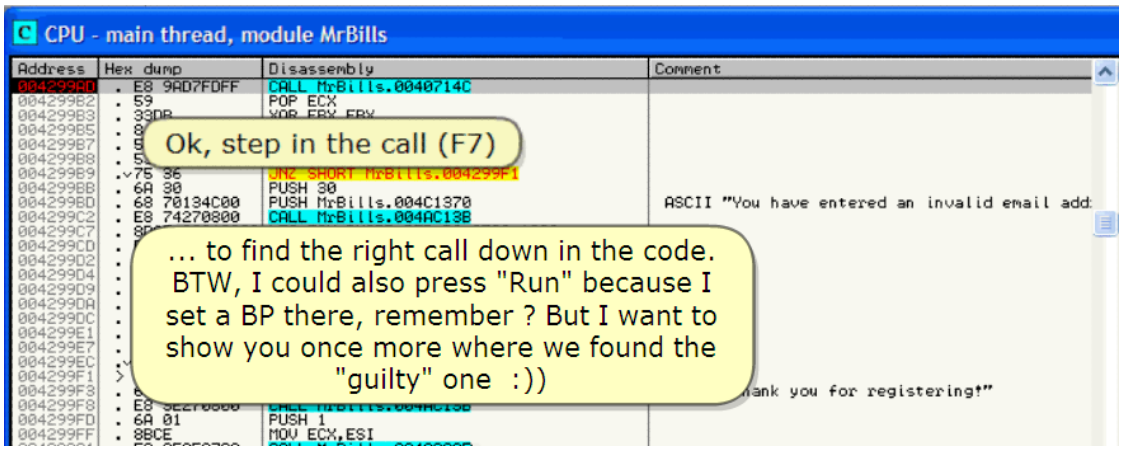
Let's break again and find our way back to that call.

정확히. 일치하지 않아.

그러나 그 동안, 우리는 우리가 보기 위해 필요한 정확한 call 을 알았다.

다시 멈추자. 그리고 찾았던 방법으로 call 로 가자.

5. Patching the program



Ok, step in the call (F7)

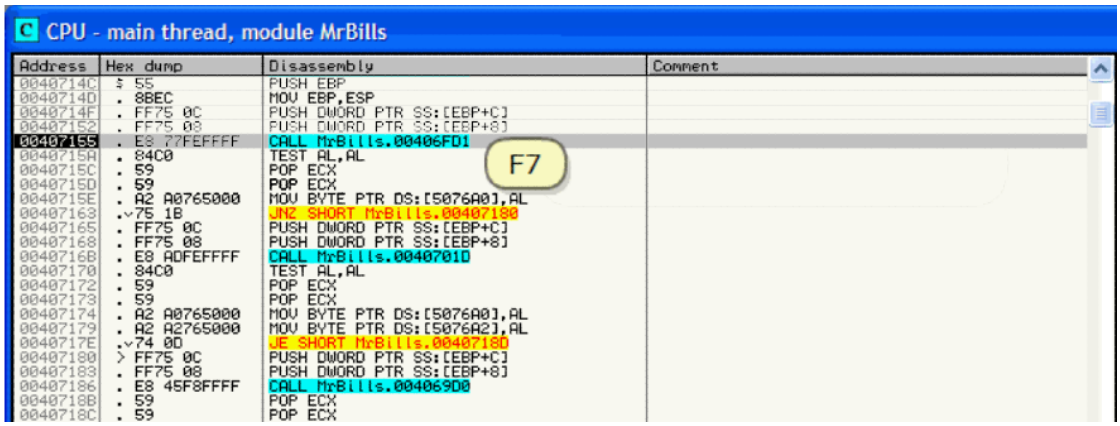
... to find the right call down in the code.

Ok, F7 로 call 안으로 들어가자.

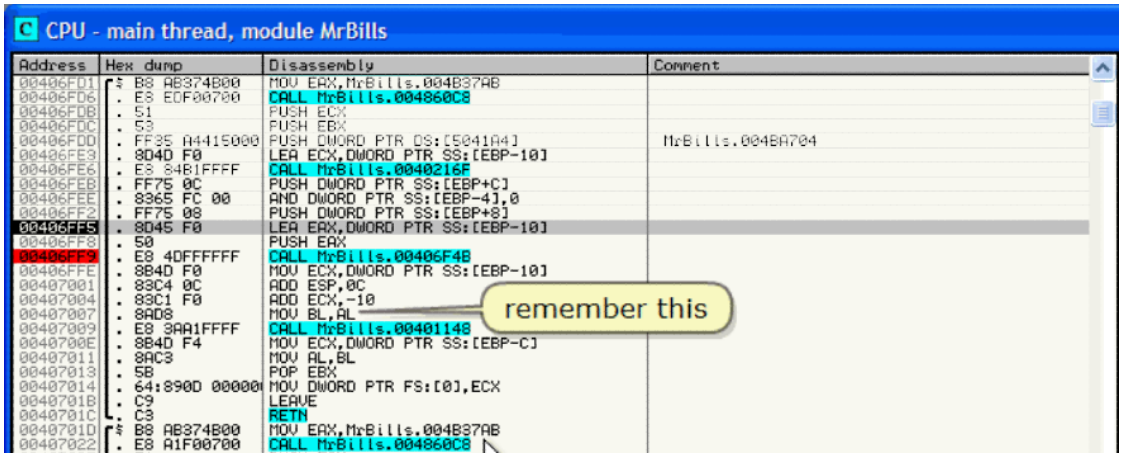
Code 의 Call 에서 바로 찾을 수 있다.

BTW, I could also press "RUN" because I set a BP there, remember? But I want to show you once more where we found the "guilty" once :))
By the way, "RUN'을 누른다. 왜냐하면 나는 BP 를 set 했다. 기억해? 그러나 나는 너에게 우리가 찾았던 의심스러운 곳을 한 번 더 보여주겠다.

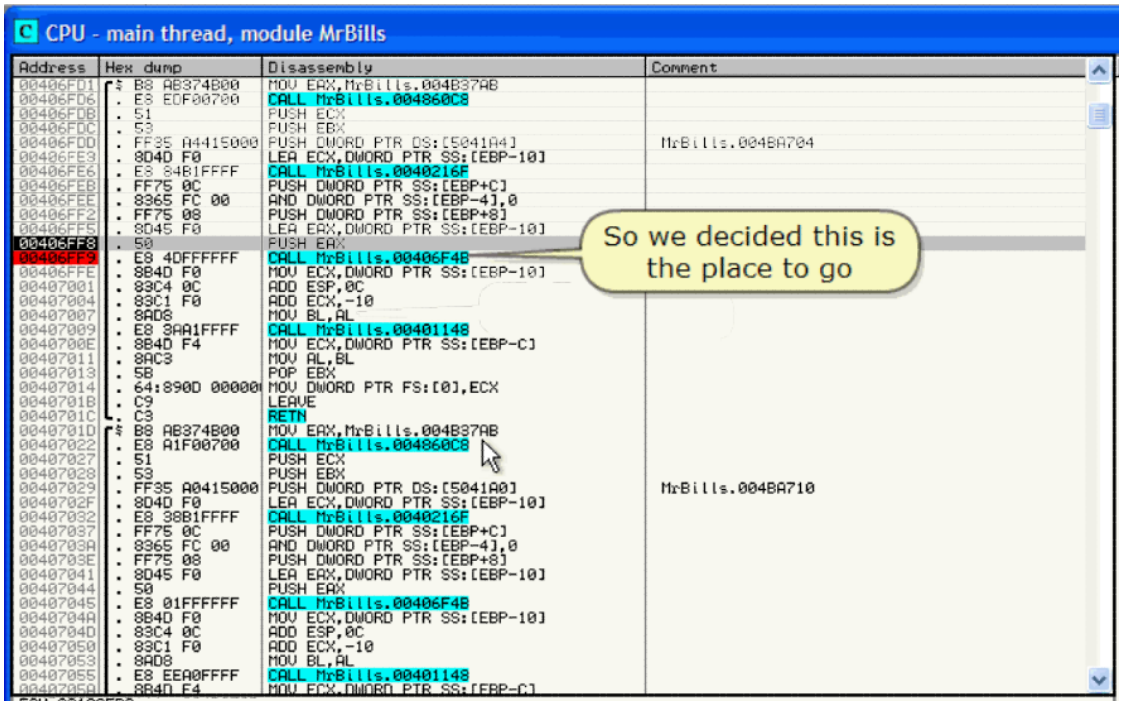
Remember the call we entered before
기억해. 우리는 전에 왔던 call 에 들어왔다.



F7
So, here we are back.
우리는 여기로 돌아왔다.
Continue stepping F8 till the call or press run
F8 을 눌러 call 까지 계속 해.



Remember this
기억해 이곳?

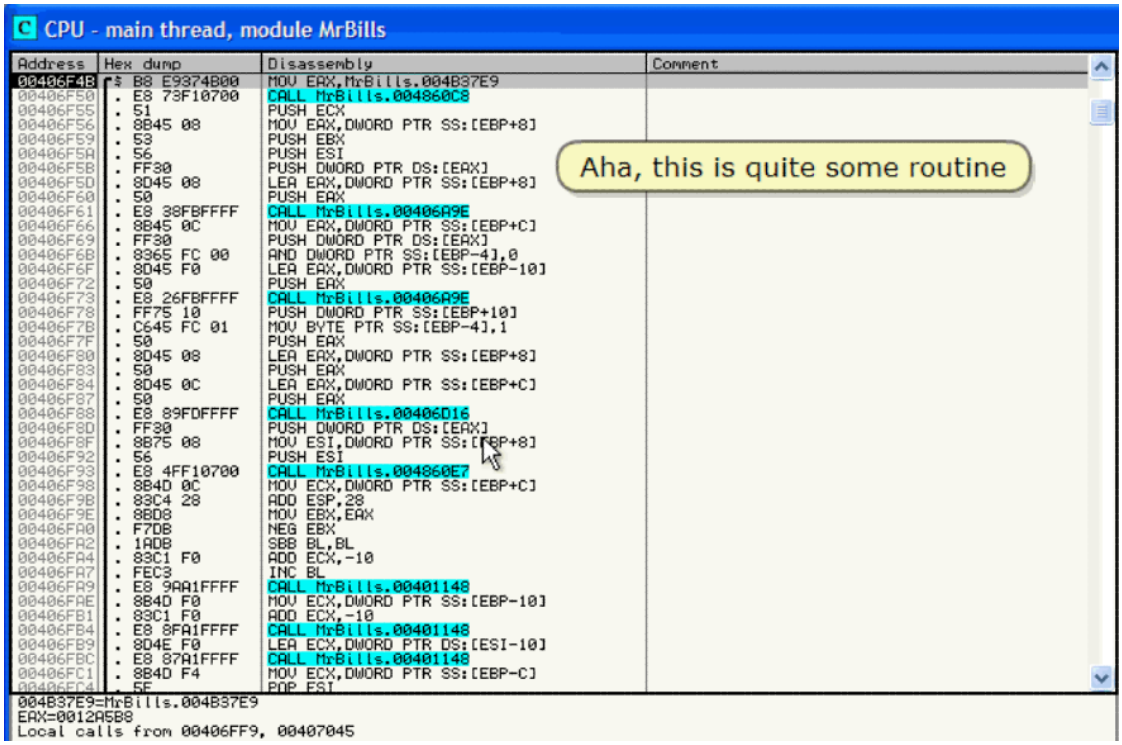


So we decided this is the place to go

이곳에서 결정한다. 들어가 보자.

So step with F7 in the call

F7 을 눌러 들어가자.



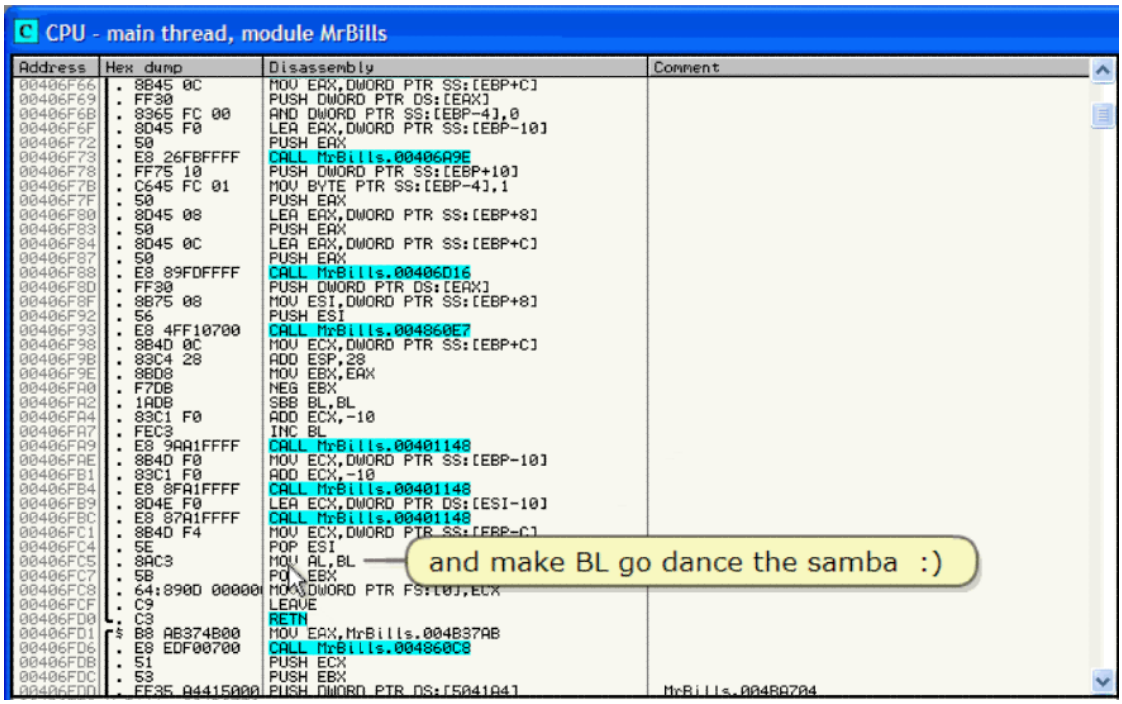
Aha, this is quite some routine

아하, 이것은 꽤 약간 되는 routine 이다.

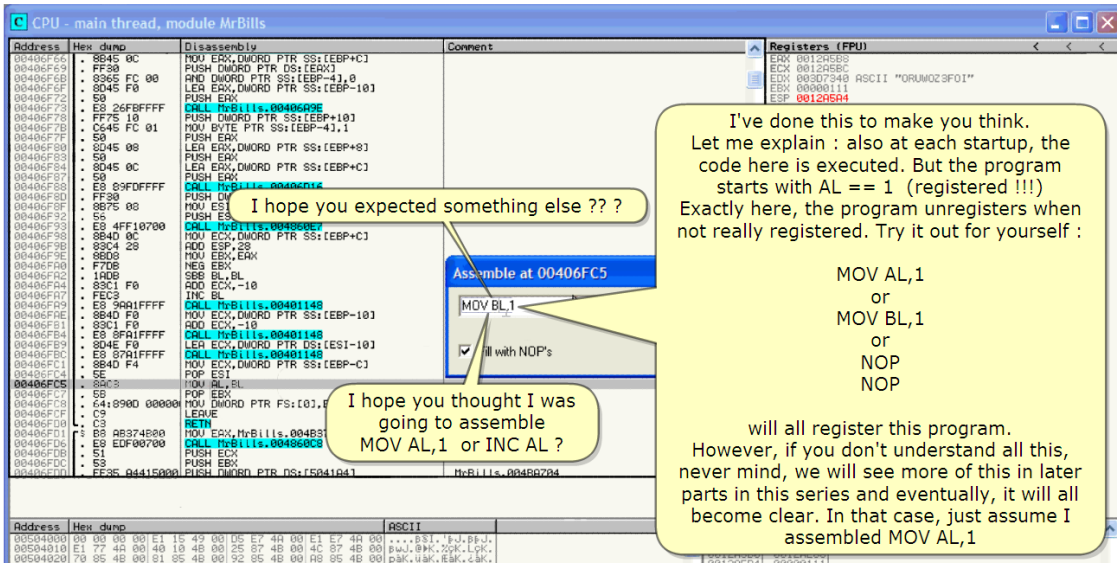
Let's look down, before the return, where AL is set
Scroll down
밑을 보자. 돌아가기 전의 AL 이 어디에서 set 되는지
Scroll 내려

Mmmm, quite a lot of code.
Code 가 꽤 많다.
For this first time, let's not dig too deep in right away.
첫번째, 정상적인 방법으로 더 이상 깊이 들어가지 않는다.
I suspect somewhere in one of these calls is the verification of the serial.
나는 여러 call 들 중에서 serial 을 검증하는 call 하나를 의심한다.
But that is for other Parts, today, let's patch AL here
다른 part 에서 하기로, 오늘은 AL 을 여기에서 patch 하겠다.

Which explains also the difference with "Advanced level patching" where we would dig deeper till the end :) (See in a later Part in this series)
"진보된 level patching"은 다르다라고 설명했다. 우리는 끝날 때까지 깊게 들어갈 수 있다.(이 series 의 다음 Part 에서 보기로 하자)



And make BL go dance the samba :)
BL 을 samba 춤 추게 만들자.



I hope you expected something else ???

다른 걸 예상했어?

I hope you thought I was going to assemble MOV AL, 1 or INC AL?

네 생각에 MOV AL, 1 이거나 INC AL 로 assemble 할 거라 생각했겠지?

I've done this to make you think.

네가 그런 생각을 하게 만들었어.

Let me explain : also at each startup, the code here is executed. But the program starts with AL == 1 (registered !!!)

설명 해줄께 :각 startup 은, code 가 이곳에서 실행됐다. 그러나 program 은 AL == 1 일 때 실행된다. (등록 !!!)

Exactly here, the program unregisters when not really registered. Try it out for yourself :

정확히 이곳이다, 정말 등록되지 않았을 때 program 은 미등록 됐다. 너는 이렇게 할 수 있다.

MOV AL, 1

Or

MOV BL,1

Or

NOP

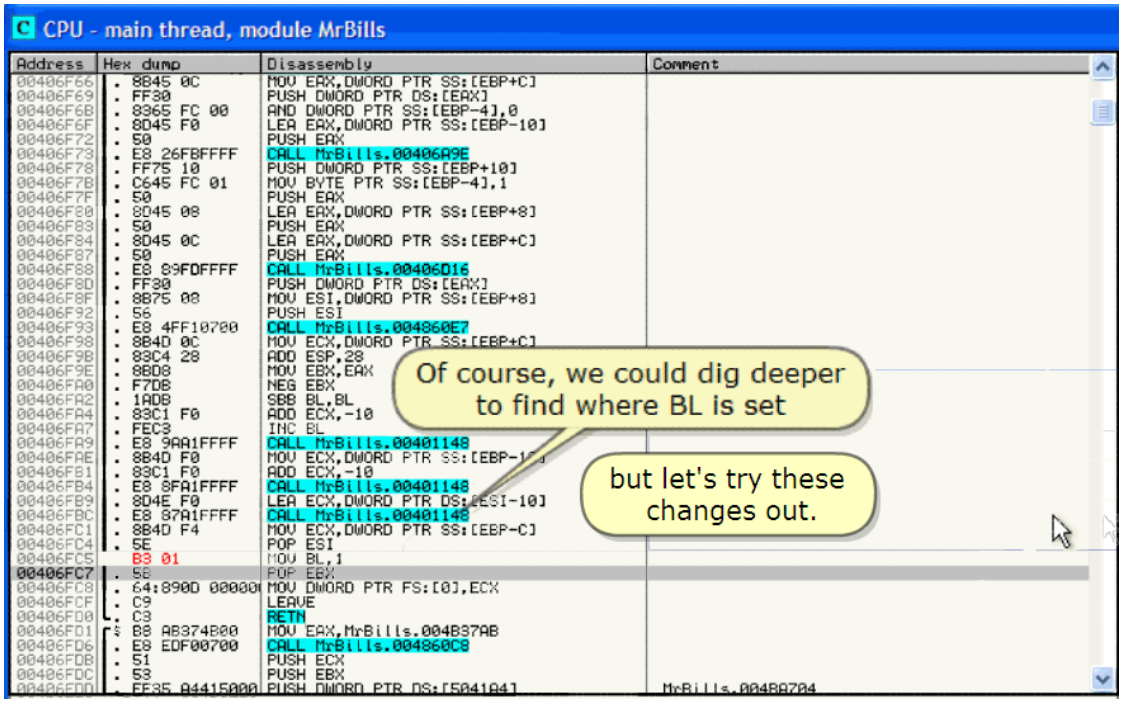
NOP

Will all register this program.

이 program 에 등록할 수 있다.

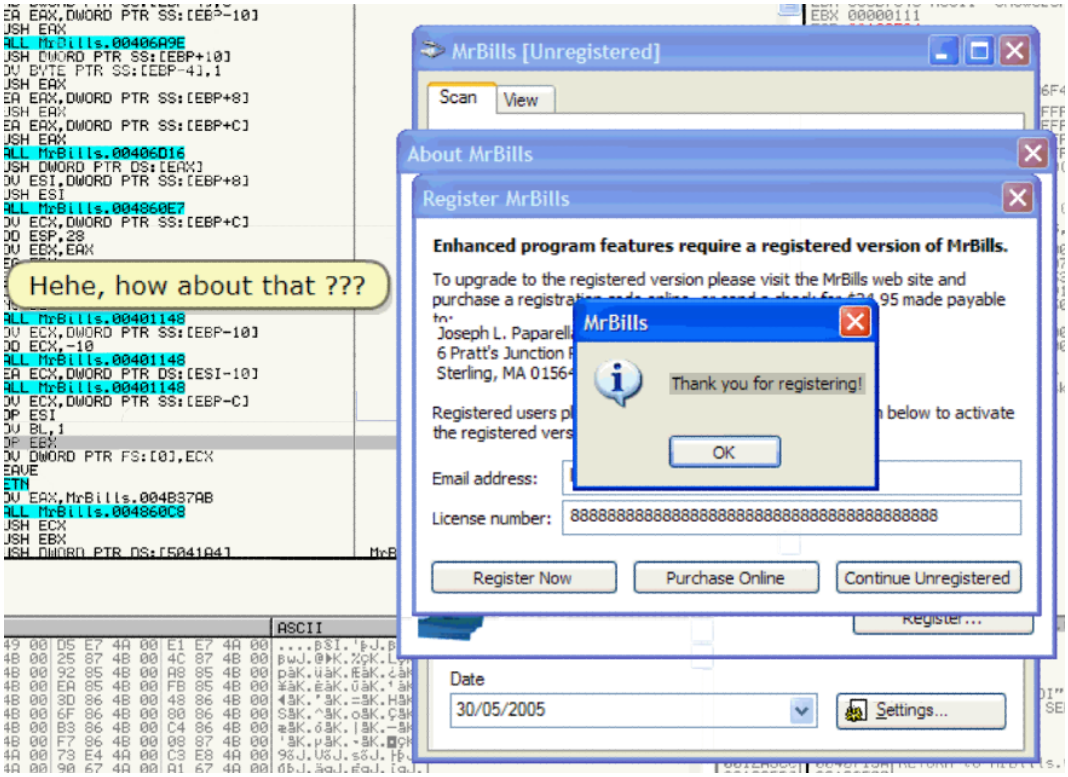
However, if you don't understand all this, never mind, we will see more of this in later parts in this series and eventually, it will all become clear.

그러나, 네가 이것에 대해서 이해가 안됐다면, 걱정하지 마. 우리는 이 series 의 뒤 part 에서 좀 더 볼 거야. 결국 그것은 명확해질 거야.
It that case, just assume I assembled MOV AL, 1
이번 경우는, MOV AL, 1 로 assemble 했을 거라 추정했다.



Of course, we could dig deeper to find where BL is set
But let's try these changes out.
물론, 우리는 BL 이 어디에서 set 되는지 좀 더 깊게 찾을 수 있다.
그러나 여기에서 바꾼 것으로 끝내자.

For now, give it a go (or F9)
이제, 실행해 보자.(F9)

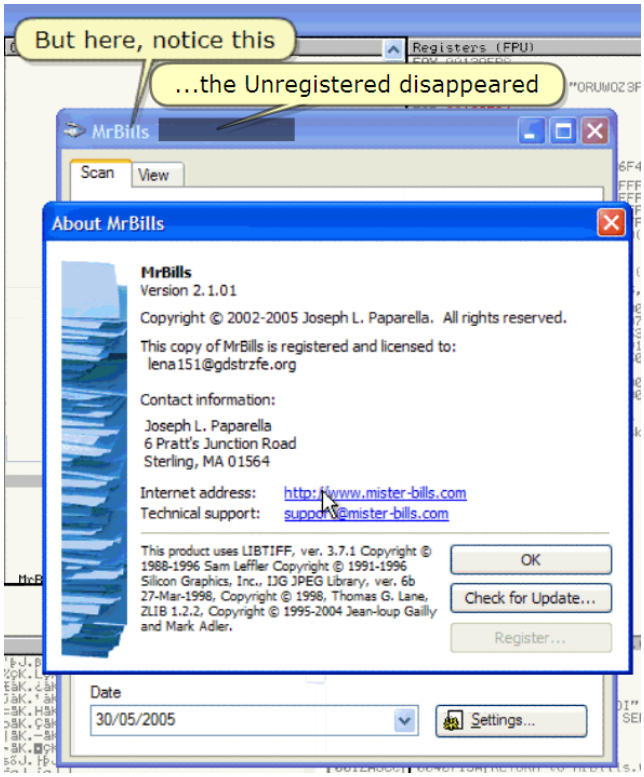


Hehe, how about that ???

헤헤, 어때 ???

Now notice this

이것을 알려준다.



But here, notice this

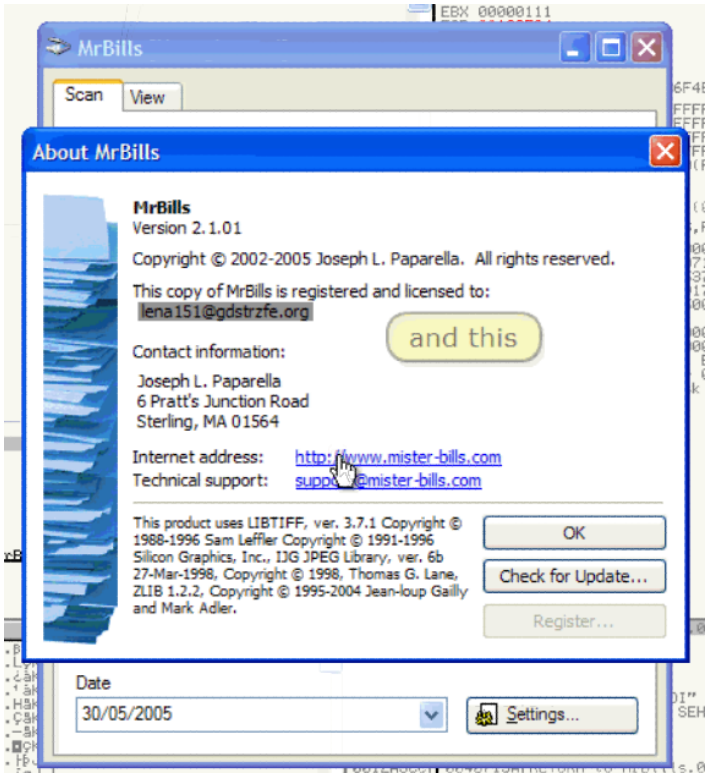
... the Unregistered disappeared

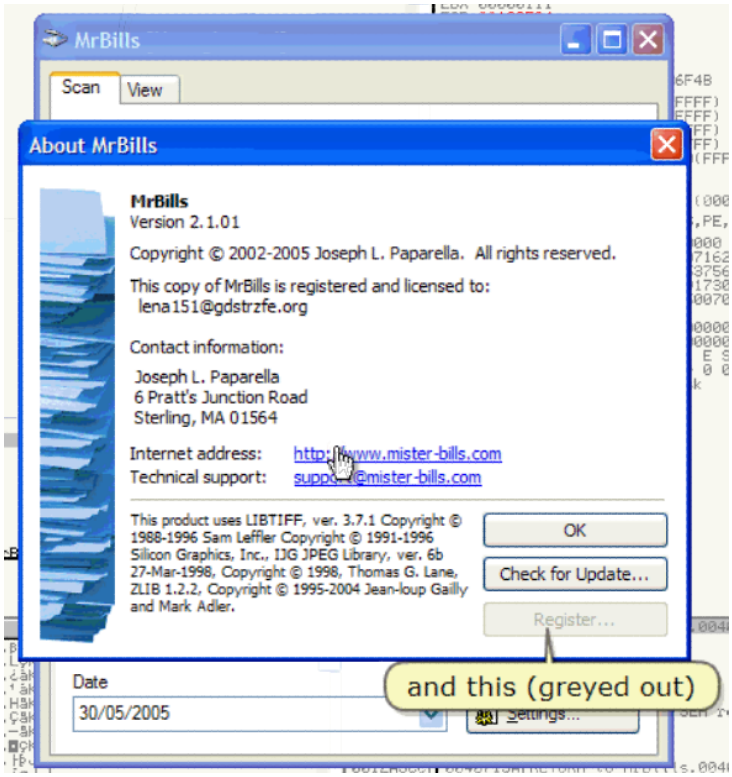
여기도, 알린다.

미등록 되어서 사라졌다.

And this

이것도





And this(greyed out)

이것도(회색으로 바뀌었다)

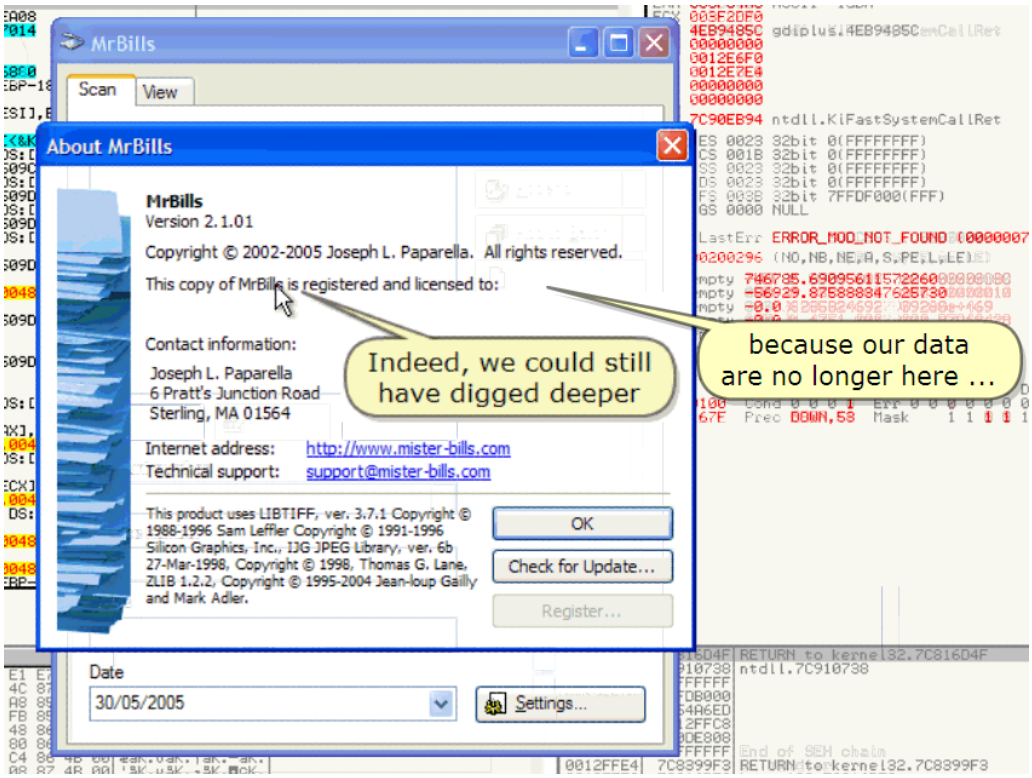
So, save the changes

그래서, 바뀐 것들을 저장하자.

... under a new name
다른 이름으로 저장해
And try it out
Test 해보자.

6. Testing the patched program

And run the soft
Let's verify the program
부드럽게 실행하자.
Program 을 검증하자.



Indeed, we could still have digged deeper
정말, 우리는 깊게 팔 수 있었다.
Because our data are no long here ...
왜냐하면 우리의 data 는 이곳에 더 이상 없다.
But see this
이곳을 봐
And this !!!
그리고 여기도
The "Unregistered" has gone !!!

미등록이 없어졌다 !!!

7. Conclusion

In this part 7, the primary goal was to study the behaviour of a program's registration scheme and patching it at intermediate level.
이번 part 7 에서, 가장 중요한 목표는 program 을 등록하기 위한 행동과 중간 level 로 patch 를 배우는 것이다.
I hope you understood everything fine and I also hope someone somewhere learned something from this. See me back in part 08 ;)
모든 것을 잘 이해했기를 바란다. 그리고 나는 누구든지 어느 곳에서든지 이것에서 무엇이든지 배웠기를 바란다. Part 08 에서 돌아올 것이다

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.