12.Guiding a program by multiple patching

2012 년 1 월 29 일 일요일
오후 7:48

Hello everybody.
모두들 안녕.
Welcome to this Part 12 in my series about reversing for newbies/beginners.
나의 초보자 reversing series Part 12 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.

1. Abstract

In this Part 12, we will reverse a "real" application to learn something about multiple patching a program to guide it to success.
이 Part 12 에서, 우리는 multiple patching 에 대하여 guide 하고 success 배우기 위하여 "real" application 을 reverse 할 것이다.

This is because indeed, the best practice is found in real applications.
Real application 에서 가장 좋은 연습을 찾았다.
In previous parts, we have only talked about easy and simple patching.
이전 parts 에서, 우리는 쉽고 간단한 patching 을 이야기 했다.
I hope you are ready for some "a little more difficult" patching.
네가 "a little more difficult" patching 에 대하여 준비가 됐다고 희망한다.
For better comprehension and if you are a newbie, I advise you to first see the previous part in this series before seeing this movie.
좀 더 좋게 이해하고 네가 초보자라면, 너에게 이 movie 를 보기 전에 이 series 의 이전 part 를 먼저 보라고 조언을 한다.
The goal of tutorial is to teach you something about a program's behaviour.
이 tutorial 의 목표는 너에게 이 program's behaviour 를 가르치는 것이다.
In my search not to harm authors, I found an old version of Techscheduler (version 6.01) which is no longer available for download.
나의 연구는 제작자에게 해가 되지 않는다. 나는 Techscheduler 의 old version 을(version 6.01) 찾았다. 그것은 더 이상 download 되지 않는다.
However, because it could be misused, I only included the main executable (not the install exe !) for your research in the shown techniques.
그러나, 우리에게 잘못 알려준다. 나는 오직 너의 연구를 위해 이 보여주는 technique 에서 main executable 만 포함했다.(install exe 는 없다)
This makes the program useless for "wrong" purposes. Talking a look in the specialized media, I also found this application to be "cracked" already.
나는 program 을 "wrong" 목적으로 사용하지 않는다. 특화된 media 를 가져, 나는 이미 이 application 의 "cracked" version 을 찾았다.
Here, this application is only chosen because it is ideal for this tutorial in reversing and it is targeted for educational purposes only.
여기, 이 application 은 오직 선택됐다. 왜냐하면 이 reversing tutorial 에 이상적이고 target 은 오직 교육적인 목적이 있다.
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 얻은 새로운 지식을 긍정적인 방향으로 이용하길 희망한다. 문제가 있다면, Part1 를 참고하기 바란다.

2. Tools and Target

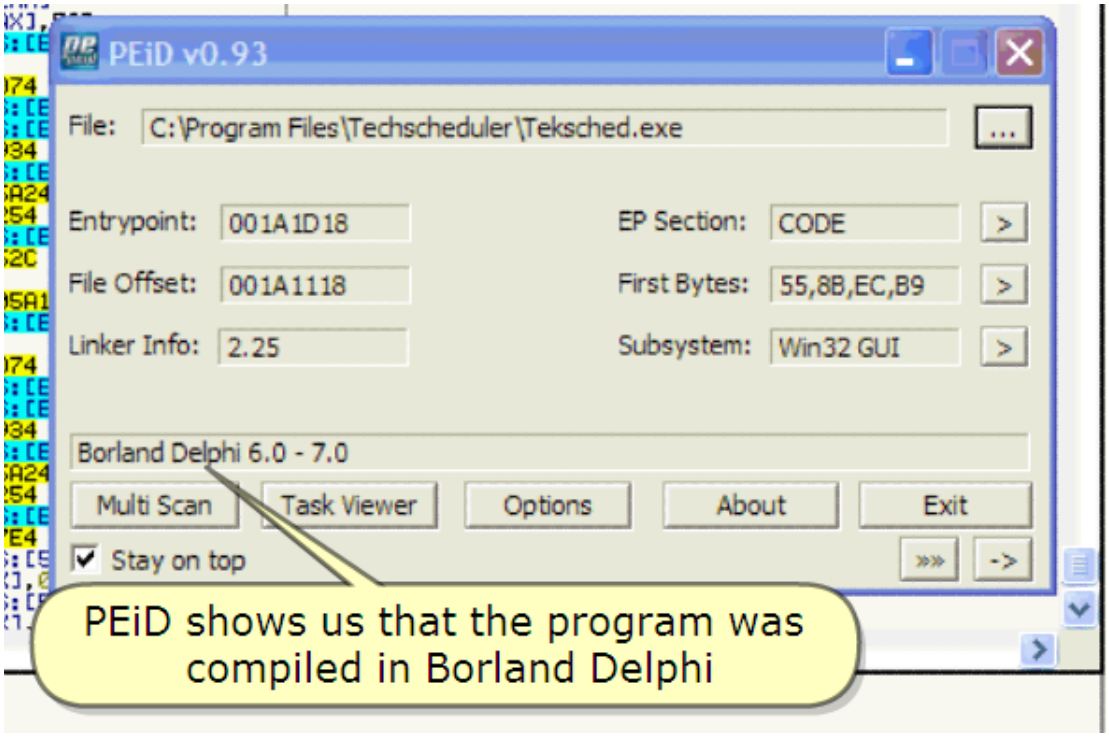
이것도 똑같음
The tools for today are : Ollydebug and... your brain.
오늘의 tools 은 : Ollydebug 와 너의 두뇌다.

The first can be obtained for free at
먼저 무료로 얻을 수 있다.
http://www.ollydbg.de

Again, the brain is your responsibility ;)
다시 한 번 말하지만, 너의 두뇌는 너의 책임감이다 ;)
The target is a program called Techscheduler 6.01, I have included the main-exe-only in this package for research.
Target program 은 Techscheduler 6.01 로 불린다. 나는 오직 너의 연구를 위해 이 package 에 main-exe 만 포함했다.

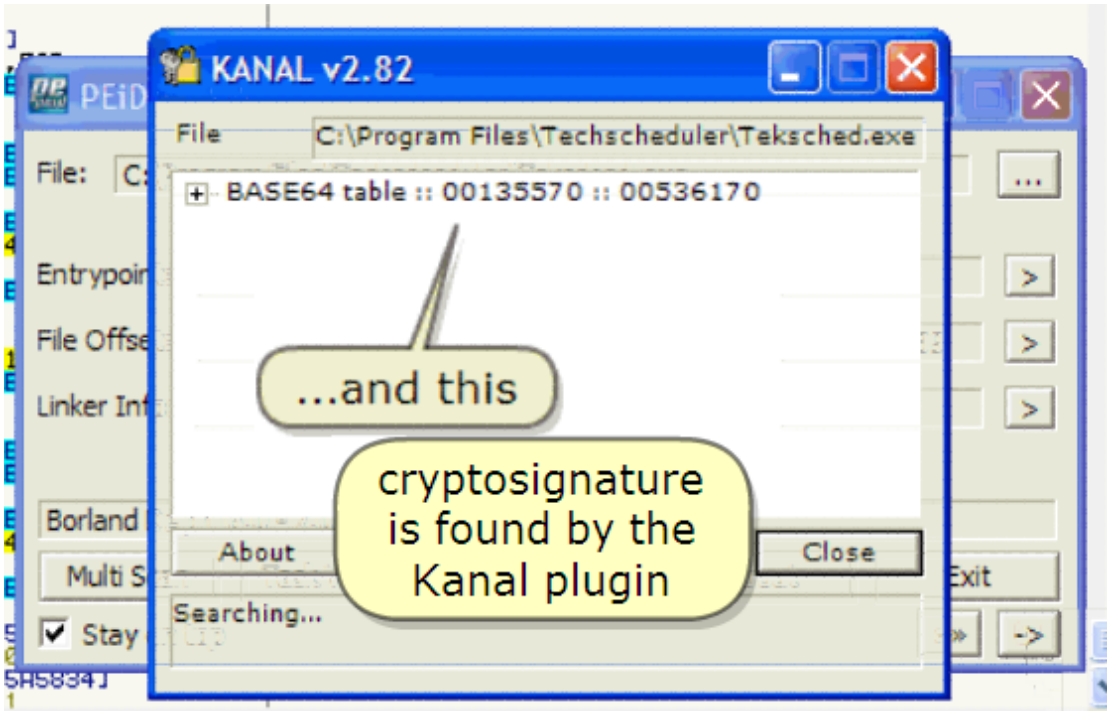
3. Behaviour of the program

Because you know meanwhile about the importance of decent study of your target and because you've seen how to do it in my previous Parts in this series,
너의 target 의 적절한 공부는 중요하다. 그리고 너는 이 series 이전 part 에서 어떻게 하는지 봐왔다.
I will only show you the things we will need to take care of while searching for the patches.
오직 너에게 Patche 를 위해 찾는 동안 우리는 돌봐 줄 것들을 필요한 것들을 보여주겠다.
Please do some preliminary study of the target on your own.
자신의 목표 중 일부 예비 연구를 수행해.
As you can see, I have already opened the target program in PEiD.
네가 볼 때, 나는 PEiD 에 이미 target program 을 열어놨다.



PEiD shows us that the program was compiled in Borland Delphi

PEiD 는 우리에게 Borland Delphi 에서 program 은 compile 된 것을 보여준다.

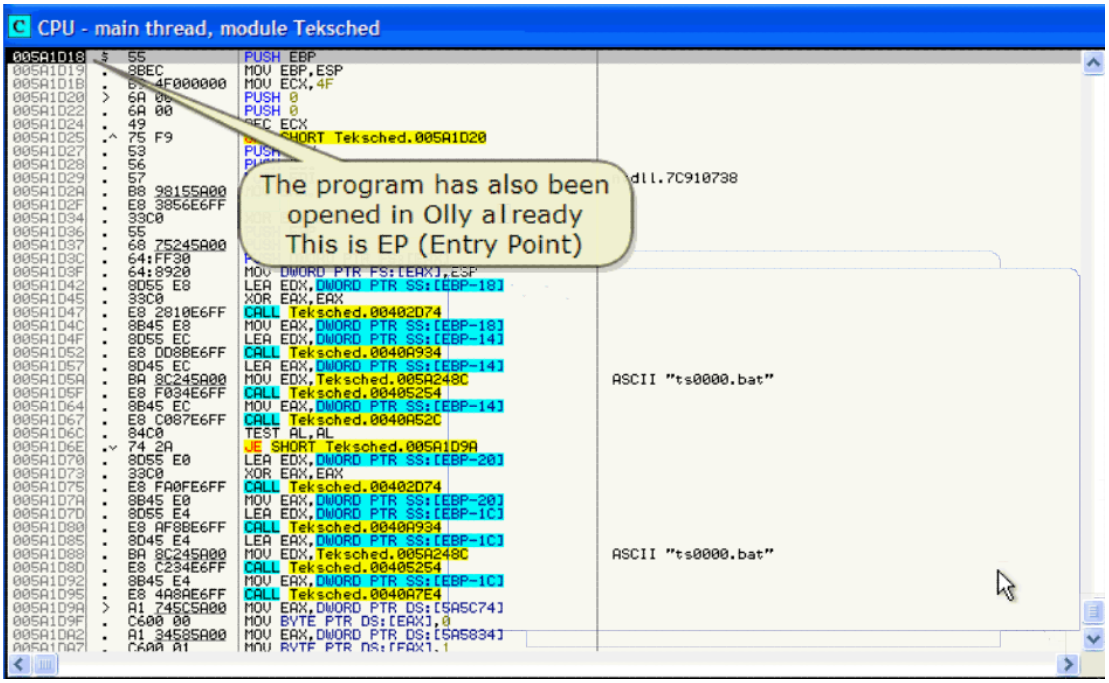


...and this cryptosignature is found by the Kanal plugin

그리고 Kanal plugin 에 의하여 cryptosignature 를 찾았다.

We can close PEiD now.

이제 우리는 PEiD 를 닫을 수 있다.



The program has also been opened in Olly already

이미 Olly 에서 program 을 열어놨다.

This is EP(Entry Point)

이것은 EP 다.(Entry Point)

번역 주)프로그램은 실행 될 때, main 이나 WinMain, 혹은 DLLMain 에서 시작하는데, 이 위치가 RVA 로 저장되어 있다. 바로 이 위치를 EP(Entry Point)라고 한다.

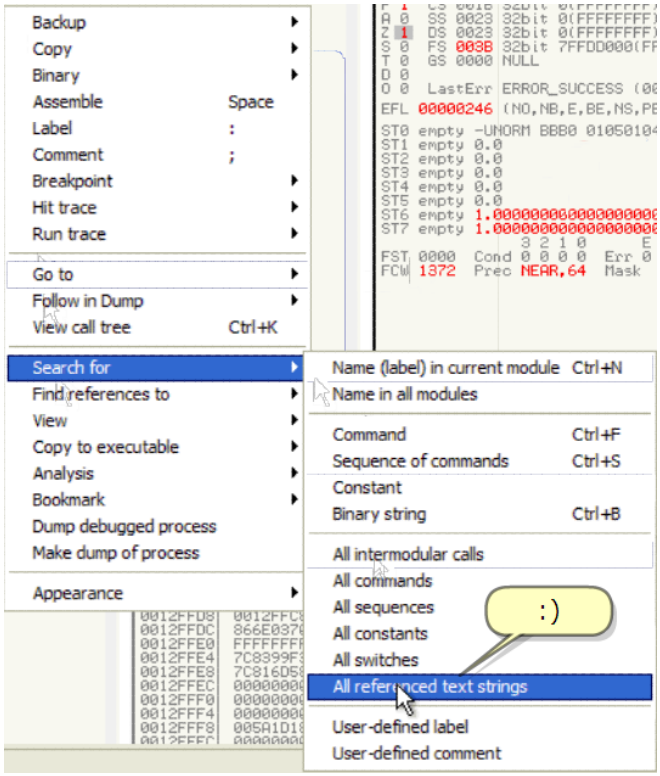
RVA : Relative Virtual Address 의 약자로서 Offset 과 같은 개념이라고 봐도 무방하다.

VA : Virtual Address 의 약자로써, 실제 물리적 메모리 주소에 매핑한 가상 메모리 주소를 말한다.

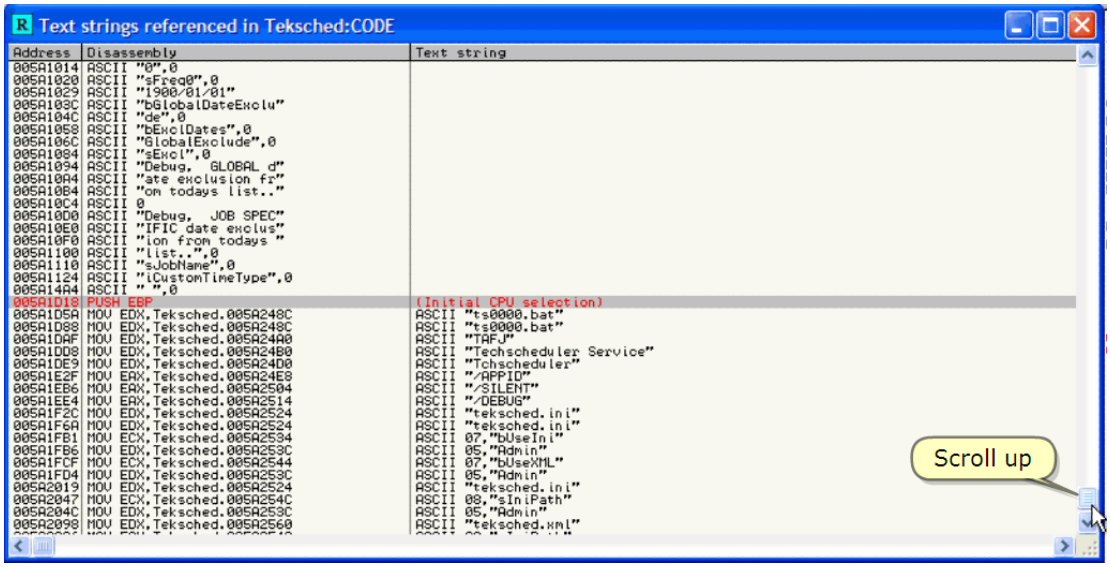
INFO : In programs, there are some strings that are very often used for registration schemes that also can give good clues on how to proceed further. Just follow …

Program 에서는 그곳에 약간의 strings 이 있다. Strings 은 등록 scheme 를 위해 매우 자주 사용된다. Registration scheme 는 우리에게 미래에 어떻게 진행할지 좋은 단서를 준다. 이제 따라가보자.

4. Finding the patches

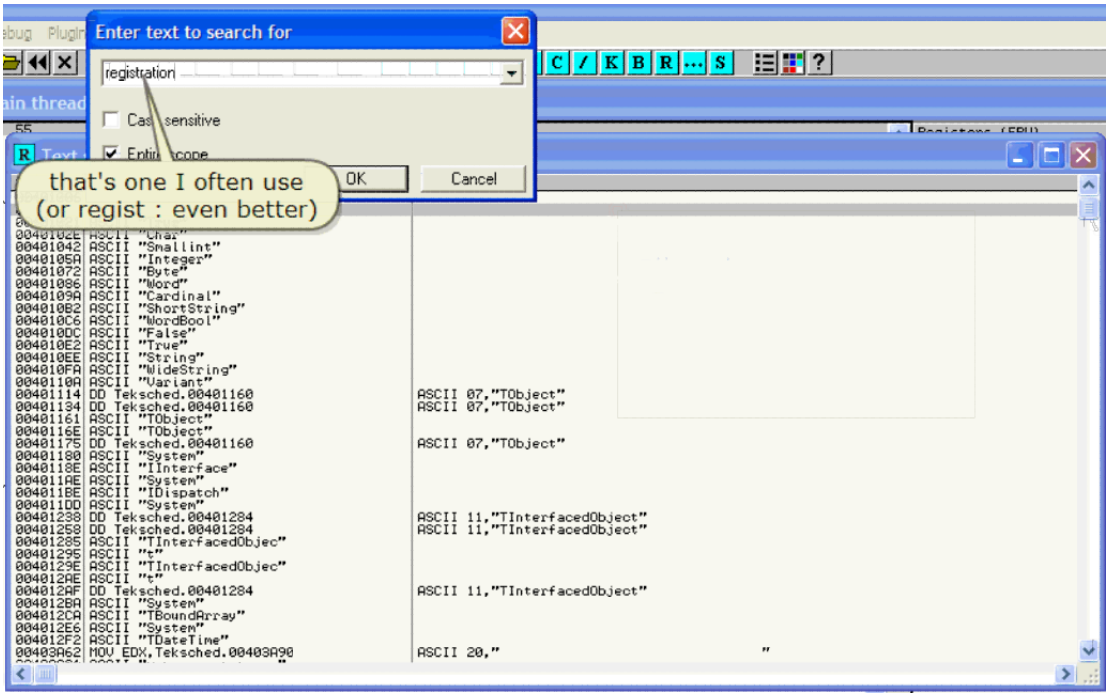


:)



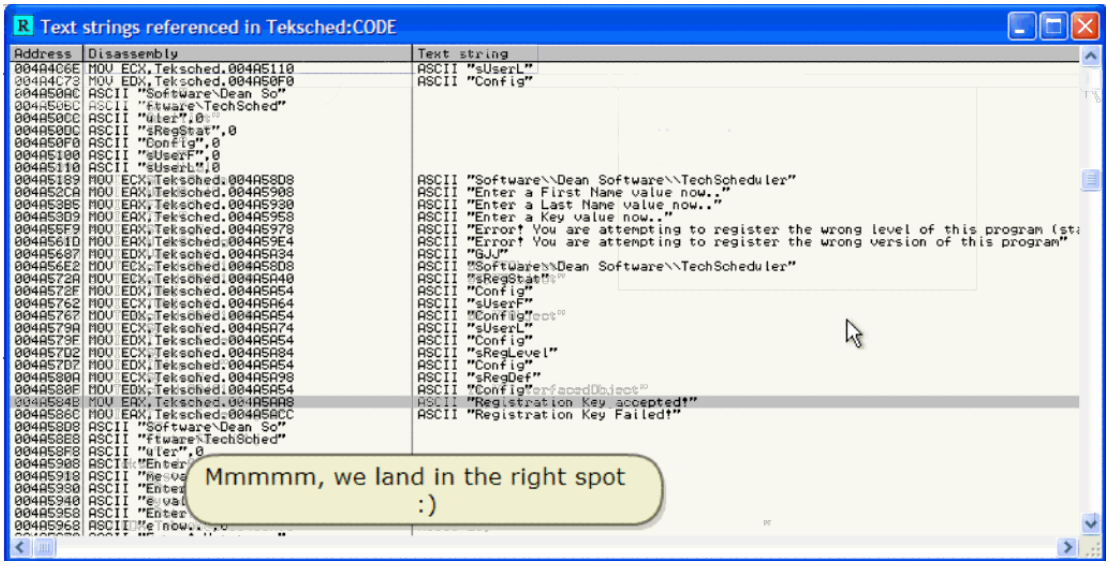
Scroll up

번역 주)string 검색할 때는 제일 처음부터 검색하라고 했다. 기억이 나지 않는다면 이전 강의를 살펴보세요.



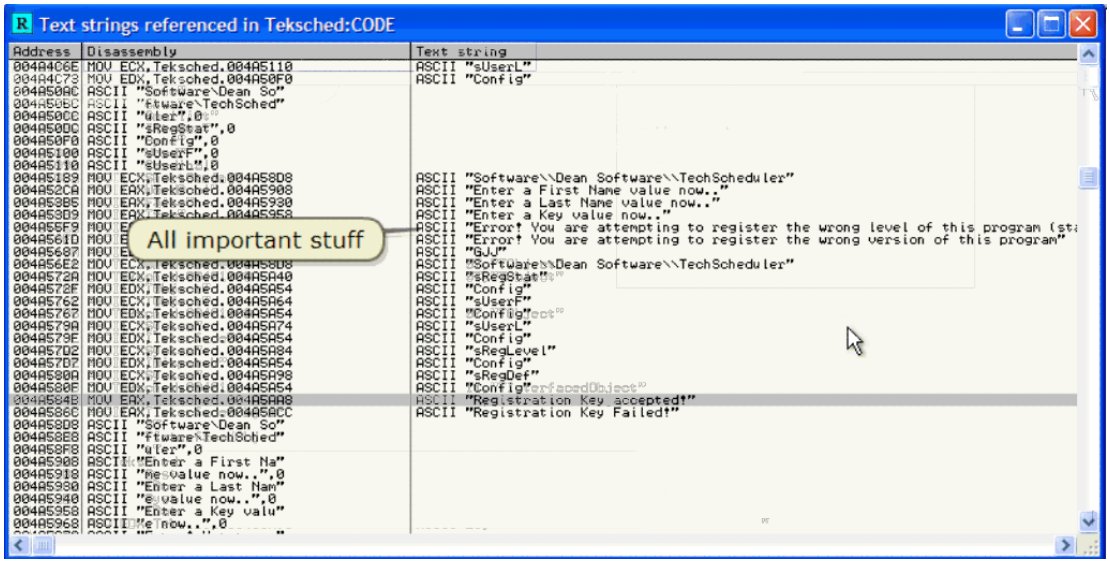
That's one I often use(or regist : even better)

이것은 자주 사용된다.(또는 regist : 이것보다 좋다)



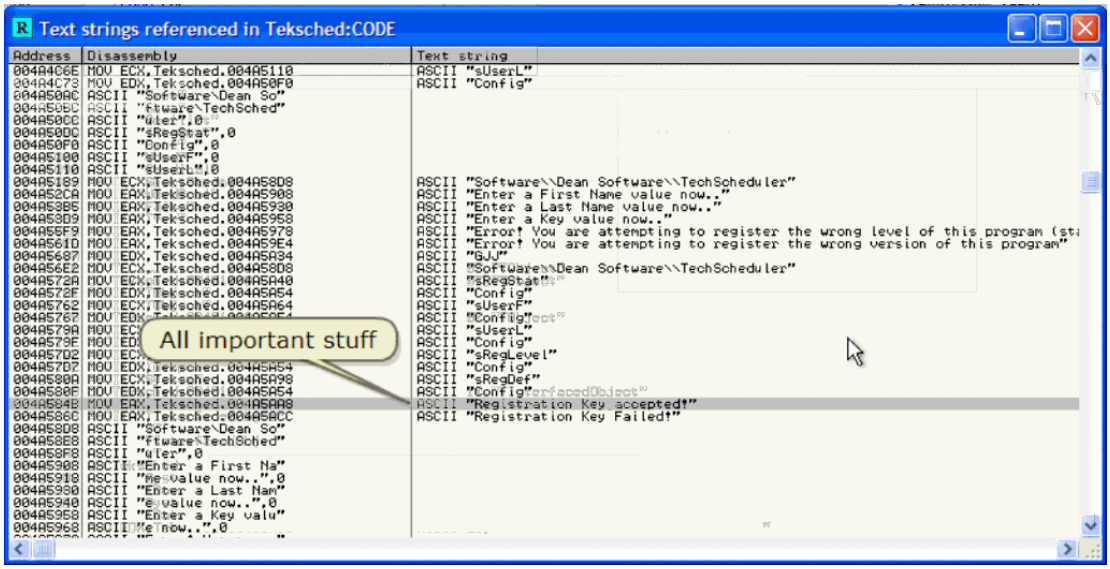
Mmmm, we land in the right spot :)

음, 우리는 좋은 지점에 도착했다 :)



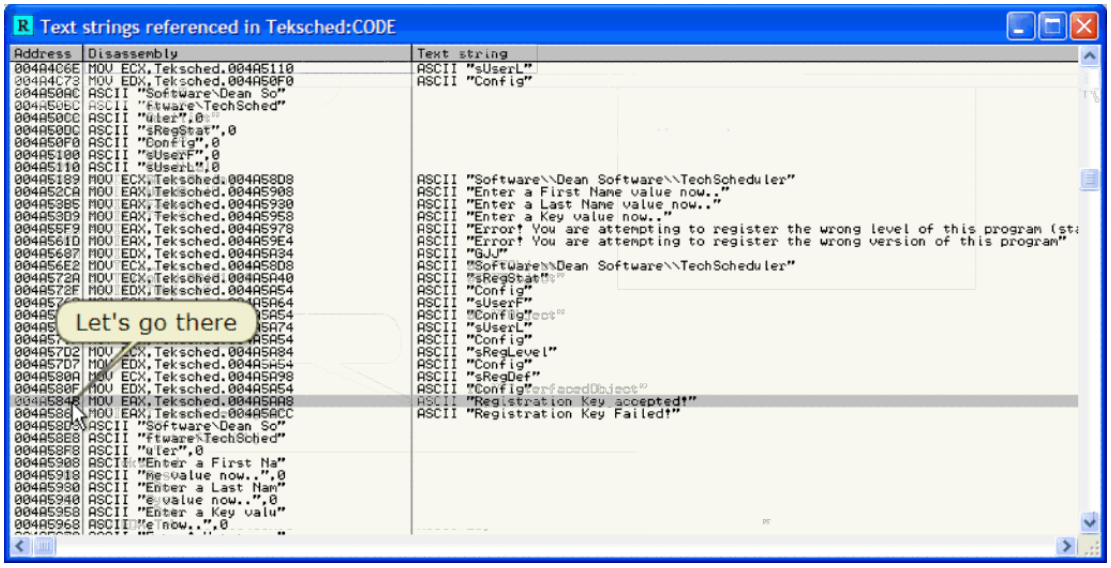
All important stuff

모두 중요한 재료



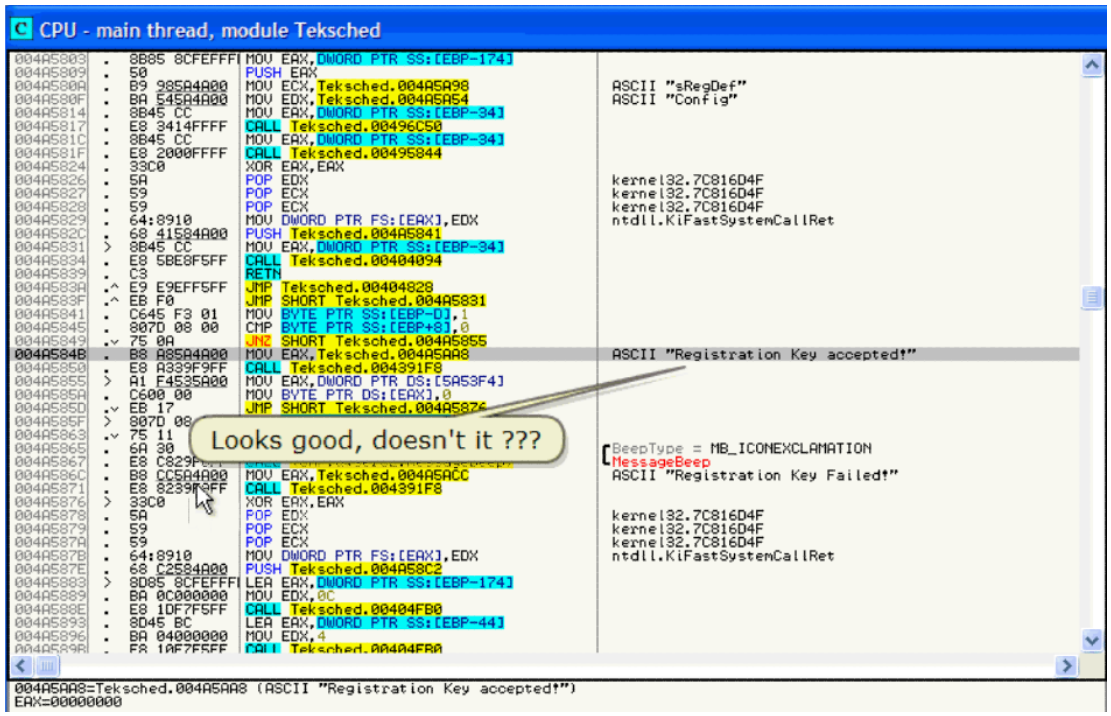
All important stuff

모두 중요한 재료



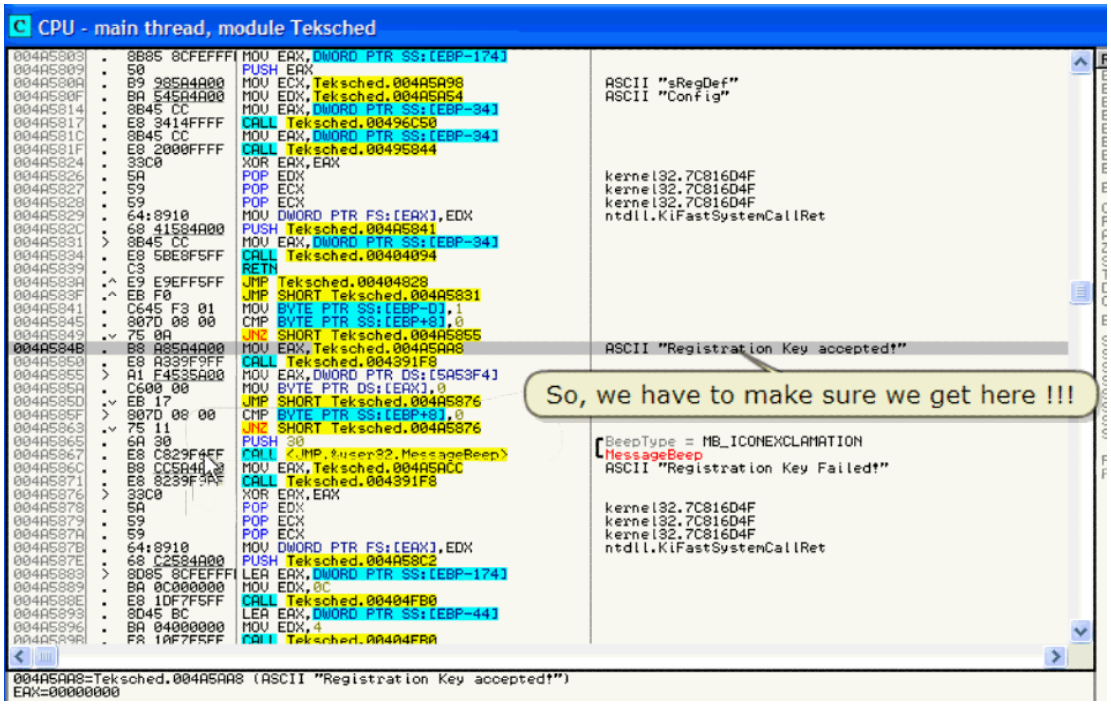
Let's go there

여기로 가자.



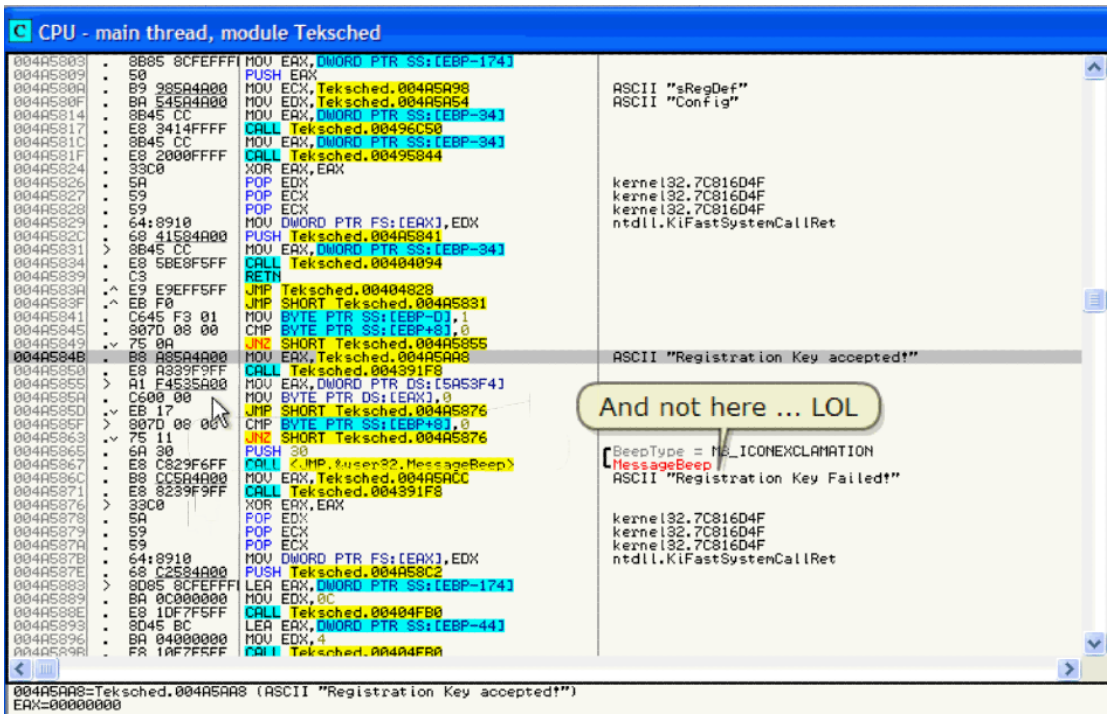
Looks good, doesn't it ???

좋아, 그렇지 ???



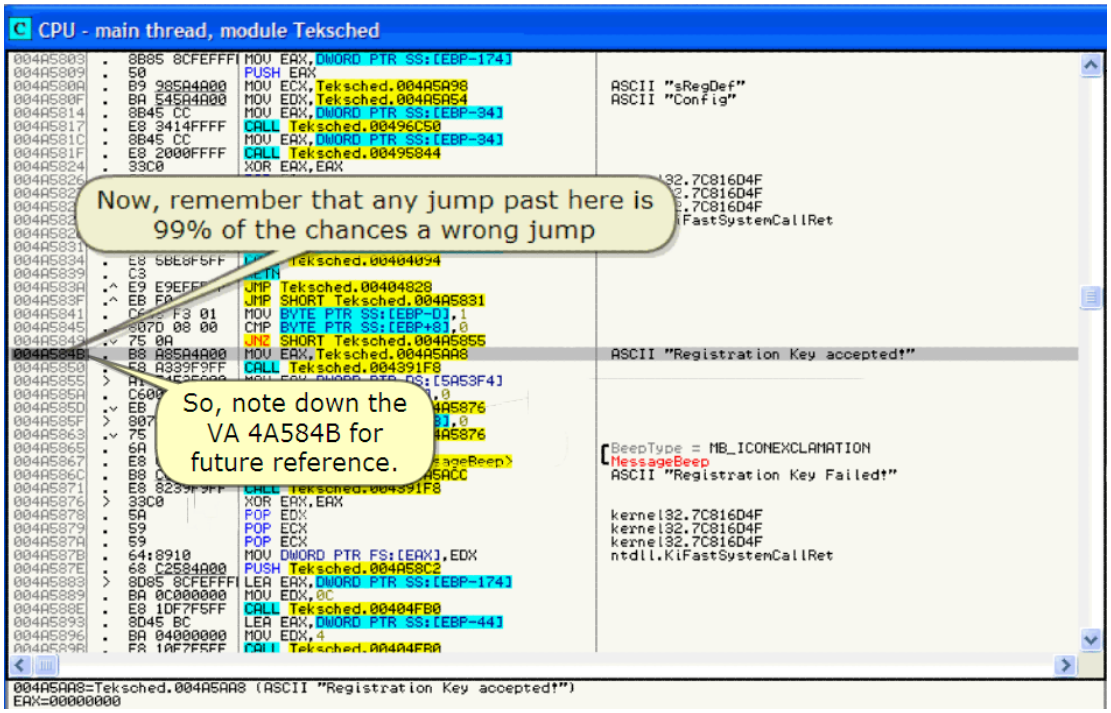
So, we have to make sure we get here !!!

그래서, 우리는 여기를 얻을 수 있게 만들어야 한다. !!!



And not here ... LOL

여기는 오지 않아야 한다.

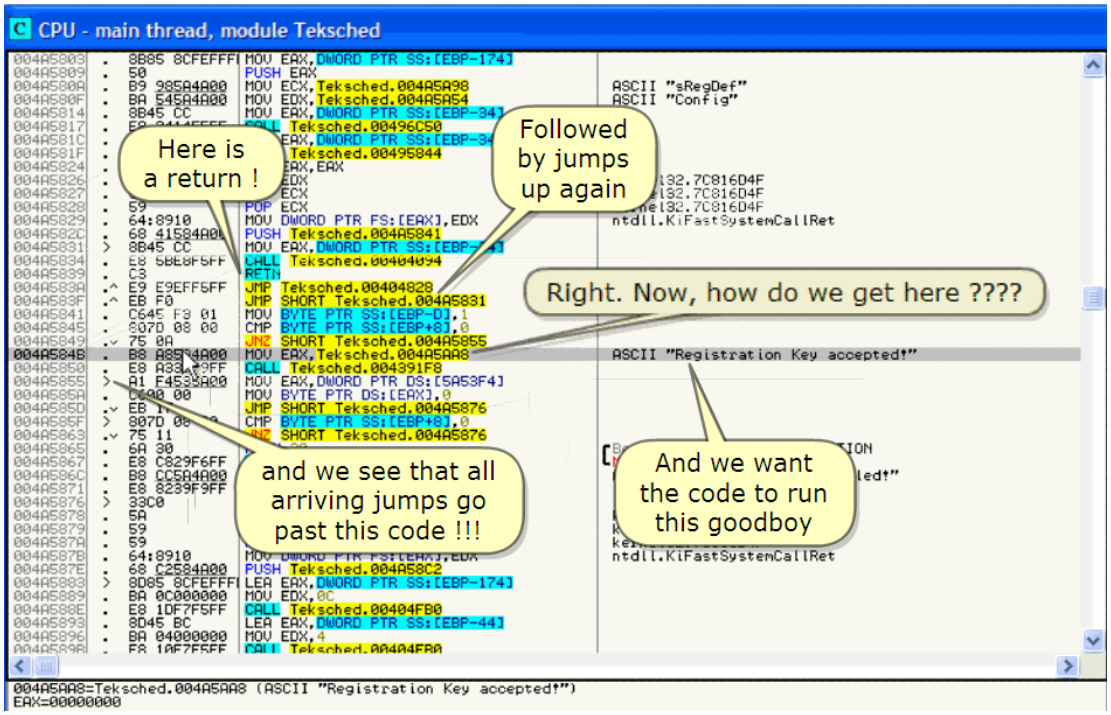


Now, remember that any jump past here is 99% of the chances a wrong jump

So, note down the VA 4A584B for future reference.

이제, 기억해. 여기를 지나가는 모든 jump 는 99% 잘못된 jump 다.

그래서, 미래의 참조를 위해 VA 4A584B 를 적어놓자.



Right. Now, how do we get here ????

좋아. 이제, 여기를 어떻게 얻겠어 ????

Here is a return !

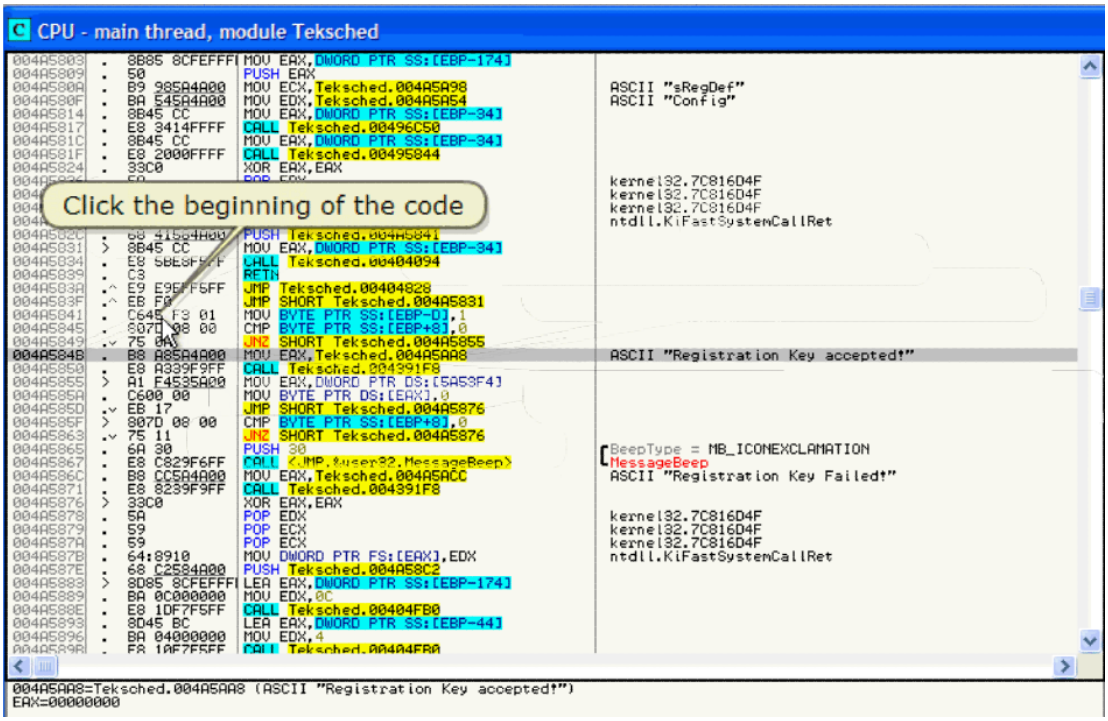
이곳은 return!
Followed by jumps up again
다시 Jump 에 의해 역행한다.
And we want the code to run this goodboy
이 goodboy 로 실행하기 위한 code 를 원한다.
And we see that all arriving jumps go past this code !!!
그리고 모든 도착하는 jumps 는 이 code 를 지나가는 걸 봐.

2012.02.04 03:00~

Mmmm, no jumps to this piece of code …
음, 이 code 조각으로 jump 하는 것은 없다.
However, there are other ways of arriving here.
그러나, 여기에 도착하는 다른 방법이 많다.
A call for example could do the trick or a jump[register+something] would make it hard for Olly to trace how we got here
Call 을 trick 하거나 Olly 가 여기에 어떻게 오는지 추적을 대비해 jump[register+something]을 어렵게 만든다.

Another important remark is the following possibility
또 다른 중요한 발언은 따라갈 가능성이다.

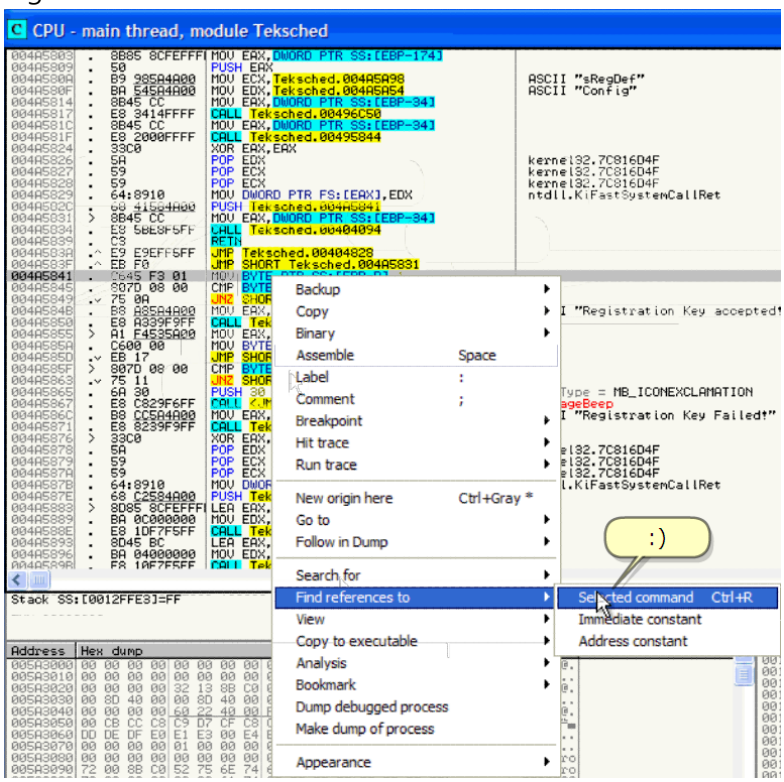
A push followed by a return == a (unconditional) jump
Return 에 뒤따라 PUSH 를 한다 == a (unconditional) jump
Anyway, see how we can search for it
게다가, 이것을 어떻게 찾는지 보자.



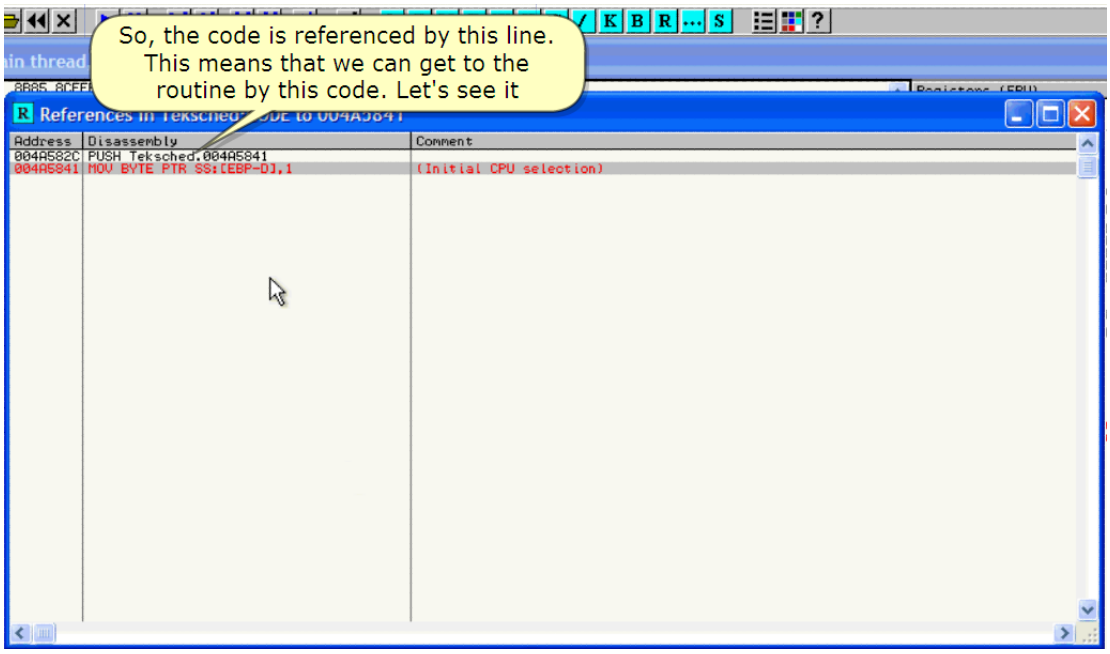
Click the beginning of the code

Code 의 시작을 click 해

Rightclick



:)



So, the code is referenced by this line.

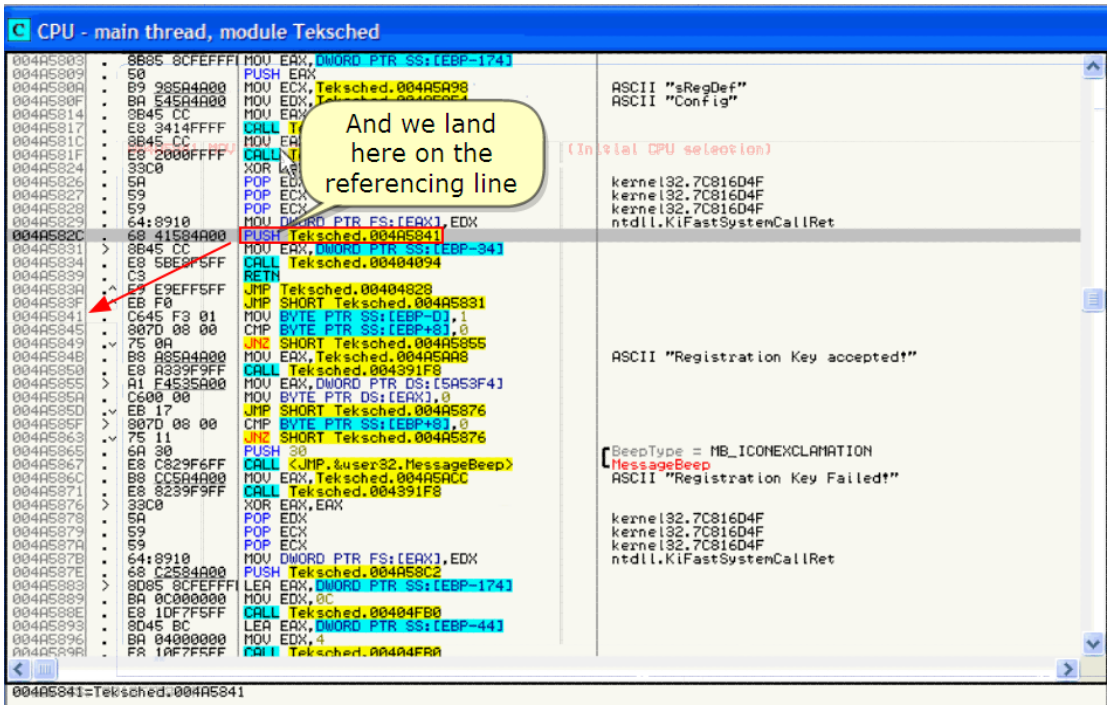
그래서, code 는 이 line 에 의해 참조됐다.

This means that we can get to the routine by this code.

우리는 이 code 에 의해 routine 으로 갈 수 있다.

Let's see it

이제 보자



And we land here on the referencing line

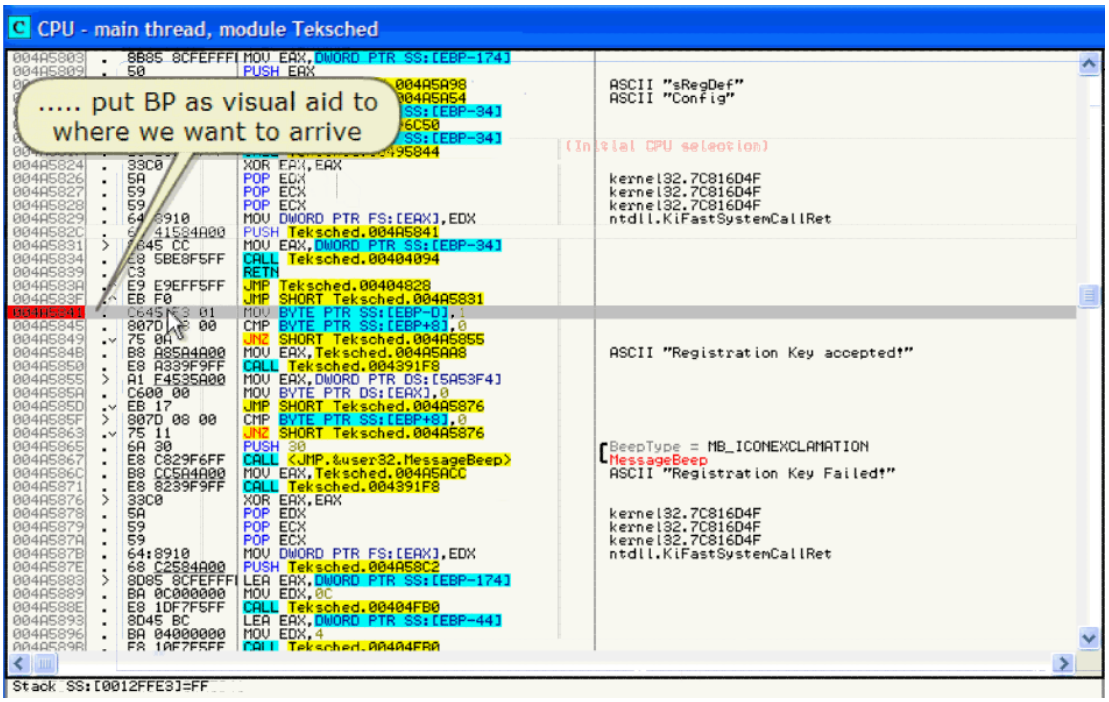
참조하고 있는 line 에 도착했다.

See once more what I mean :

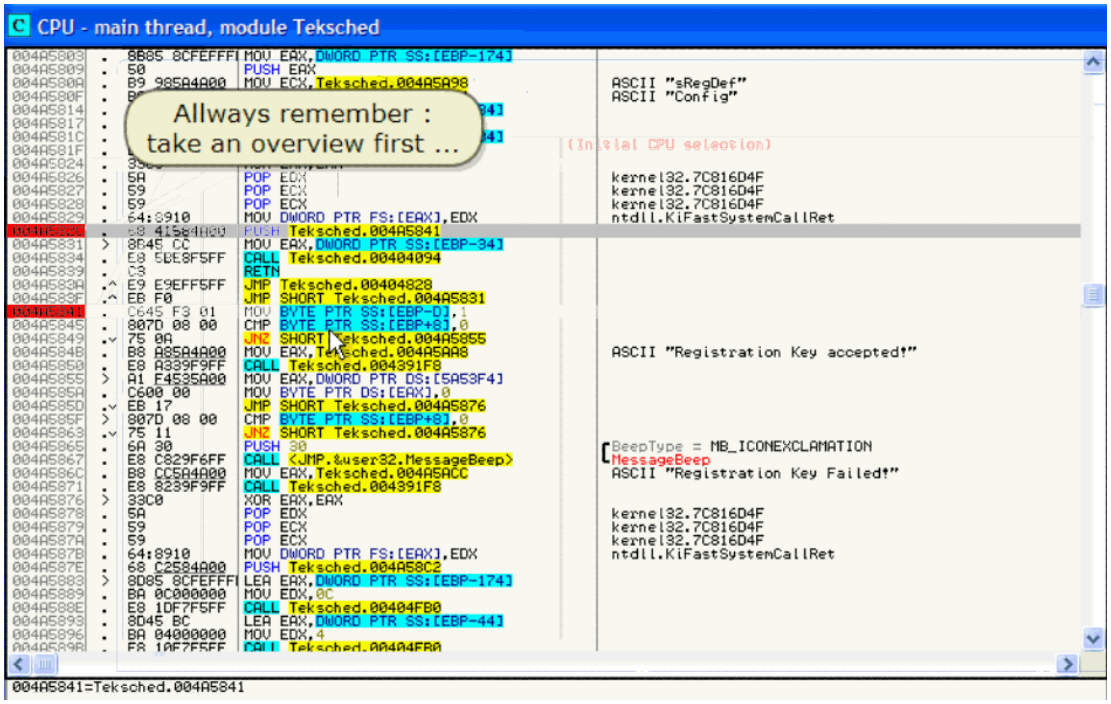
내가 말하는 게 무엇인지 한 번 더 보자.
Indeed: PUSH 4A5841
Followed by a RETURN equals JMP 004A5841
RETURN 에 의해서 JMP 004A5841 과 같아진 것을 따라간다.
번역 주)PUSH 004A5841 후에 Return 을 하면 Stack 에 004A5841 이 들어가고 return 에 의해 004A5841 이 EIP 에 들어가게 되므로 JMP 004A5841 과 같다는 말이다.
PUSH + return ---> JMP 004A5841
Which will bring us to the line we want to go
우리가 원하는 line 으로 우리를 데려간다.

INFO :
Remember this for the future.
미래를 위해 이것을 기억해.
If you want to search for the end of a routine and you find a RETURN then see if there are no PUSH xxxxx
In front that are irrelevant for other purposes
네가 routine 의 끝을 찾기 원한다면 그리고 RETURN 을 찾아 봐. 그 후 만약 PUSH xxxx 가 없다면 이 앞에 있는 것은 다른 목적들과 상관이 없다.

Ok, now that this is clear ...
Ok, 이제 이것은 명확하다 ...



.... Put BP as visual aid to where we want to arrive
우리가 도착하기를 원하는 곳에 시각화를 위해 BP 를 넣어라.



Yep, here too

예, 여기도

Always remember : take an overview first ...

항상 기억해 : 개요를 먼저 살펴봐...

So, scroll down to see what happens next

그래서, 다음에 어떤 일이 발생하는지 보기 위해 scroll 내려.

Mmmm, nothing important it seems

Probably the registering of the program if the key was accepted

Scroll up

음, 지금 보는 것은 중요하지 않다.

아마 key 가 받아 들여진다면 program 은 등록된다.

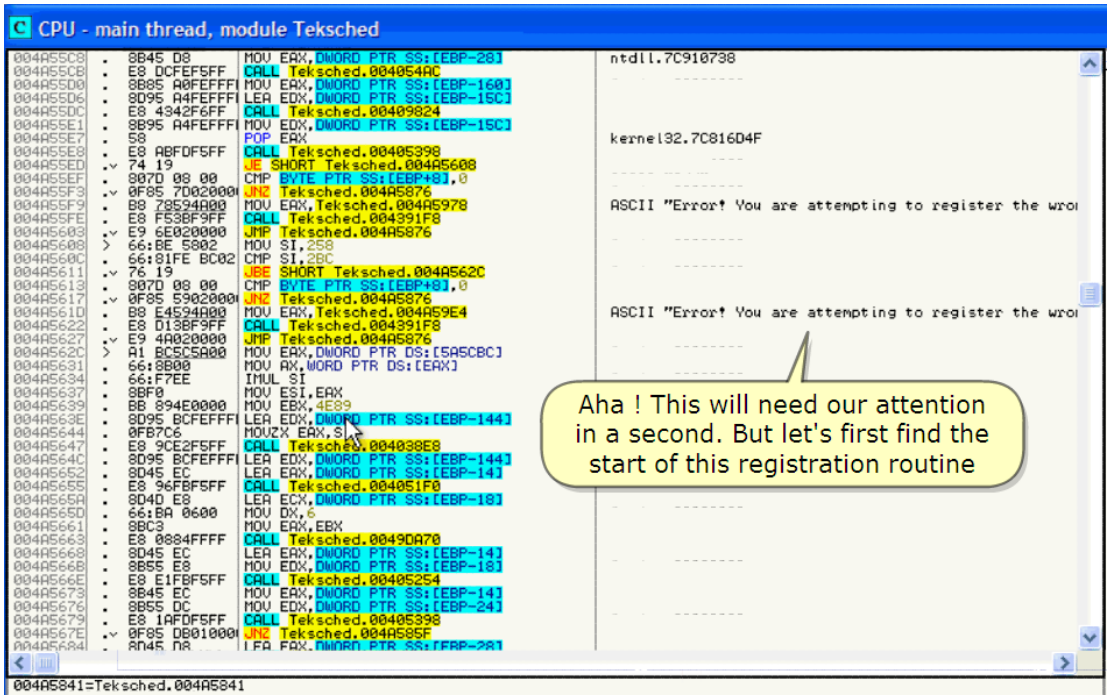
Scroll 올려.

And let's see what happens before all this

이 모든 것 다음에 어떤 일이 발생하는지 봐.

Delphi code can also be recognized by calls and calls and calls :)

Delphi code 는 call 과 call 과 call 에 의해 알려졌다.

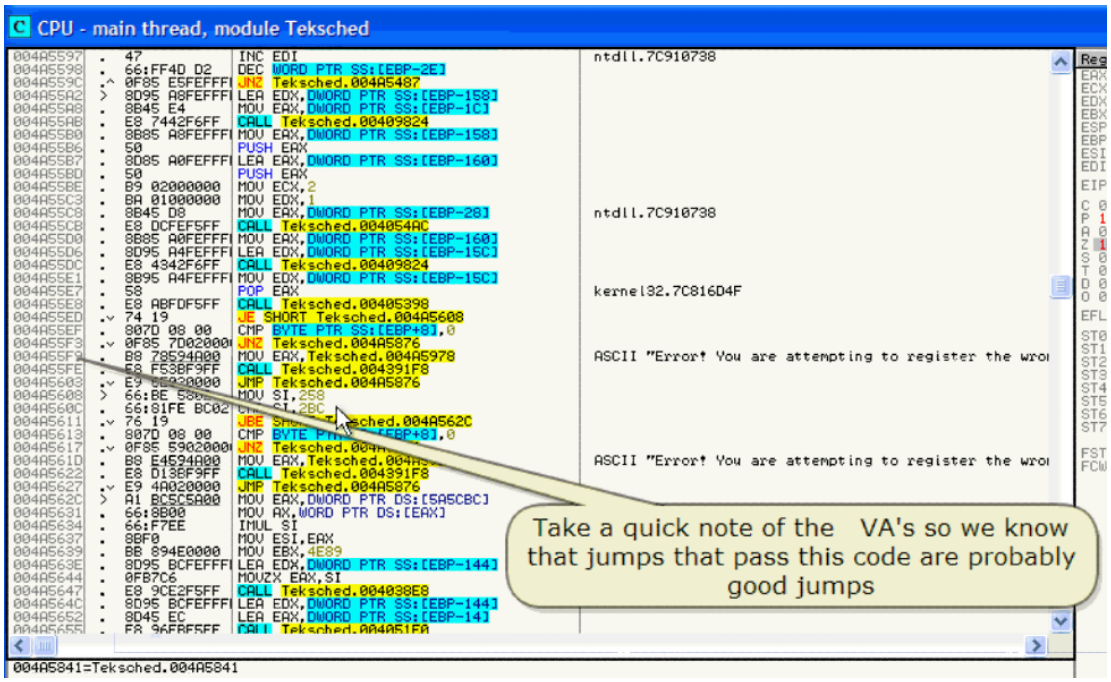


Aha! This will need our attention in a second. But let's first find the start of this registration routine

아하! 이것은 두번째로 우리의 주의를 필요로 할 것이다. 먼저 이 registration routine 의 시작부분을 찾아보자.

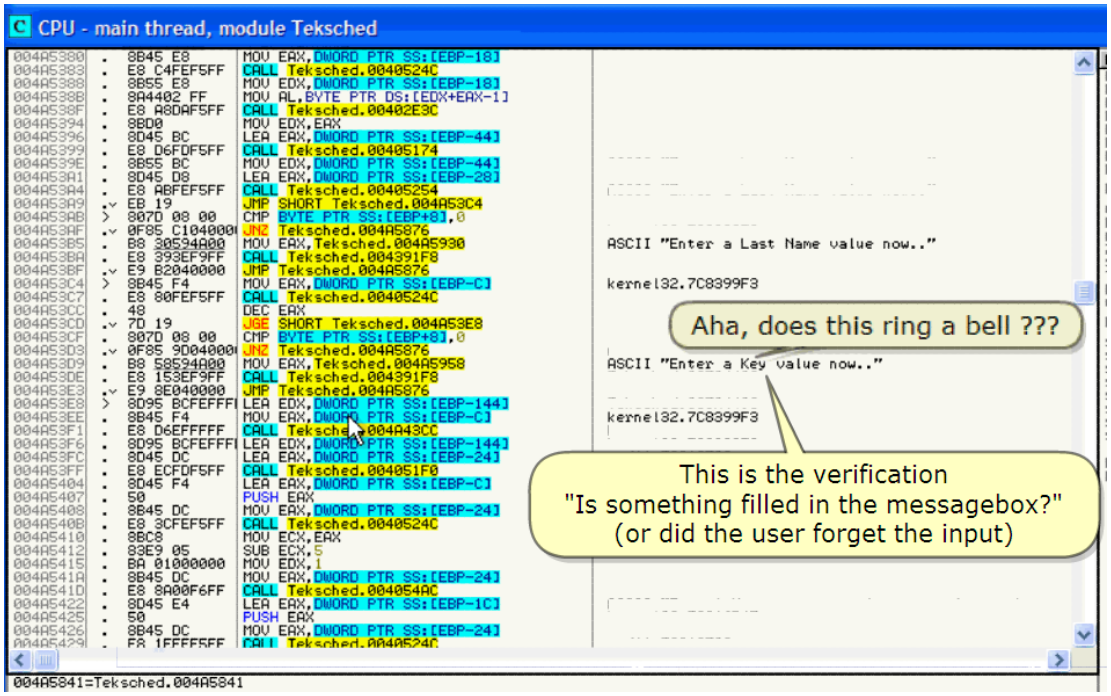
More BadBoys

Badboy 가 많다.



Take a quick note of the VA's so we know that jumps that pass this code are probably good jumps

빠르게 VA's 를 적어. 그래서 그 jump 는 이 code 를 피해야 good jump 로 간다.



Aha, does this ring a bell ???

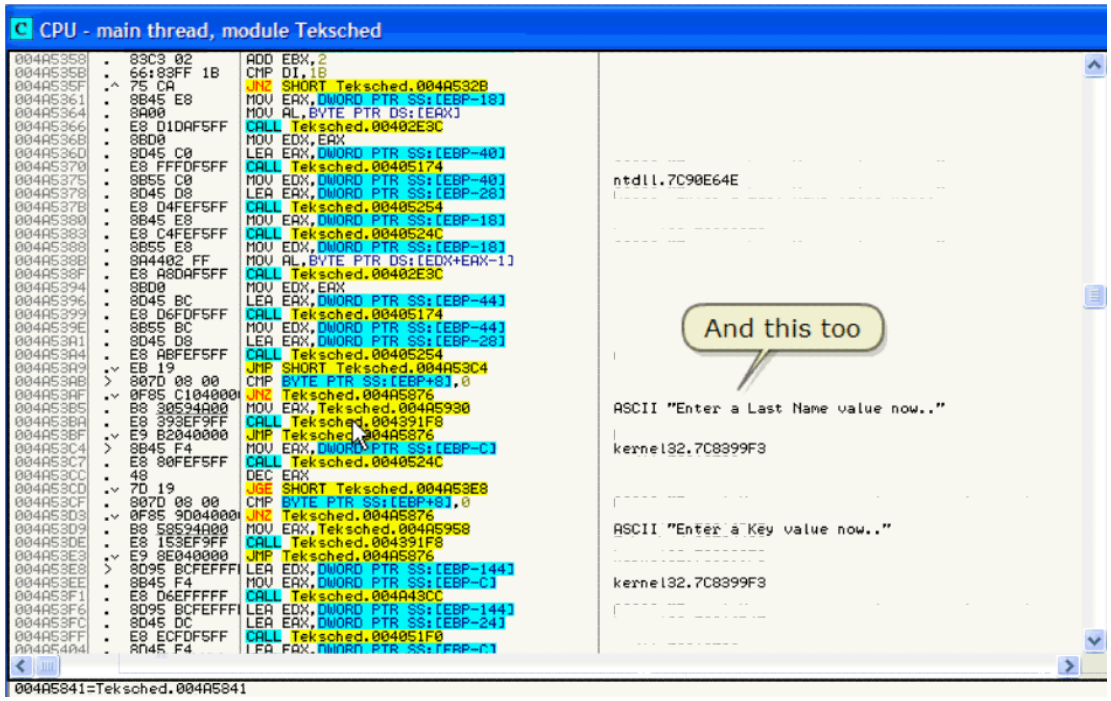
빙고?

This is the verification

"Is something filled in the messagebox?"(or did the user forget the input)

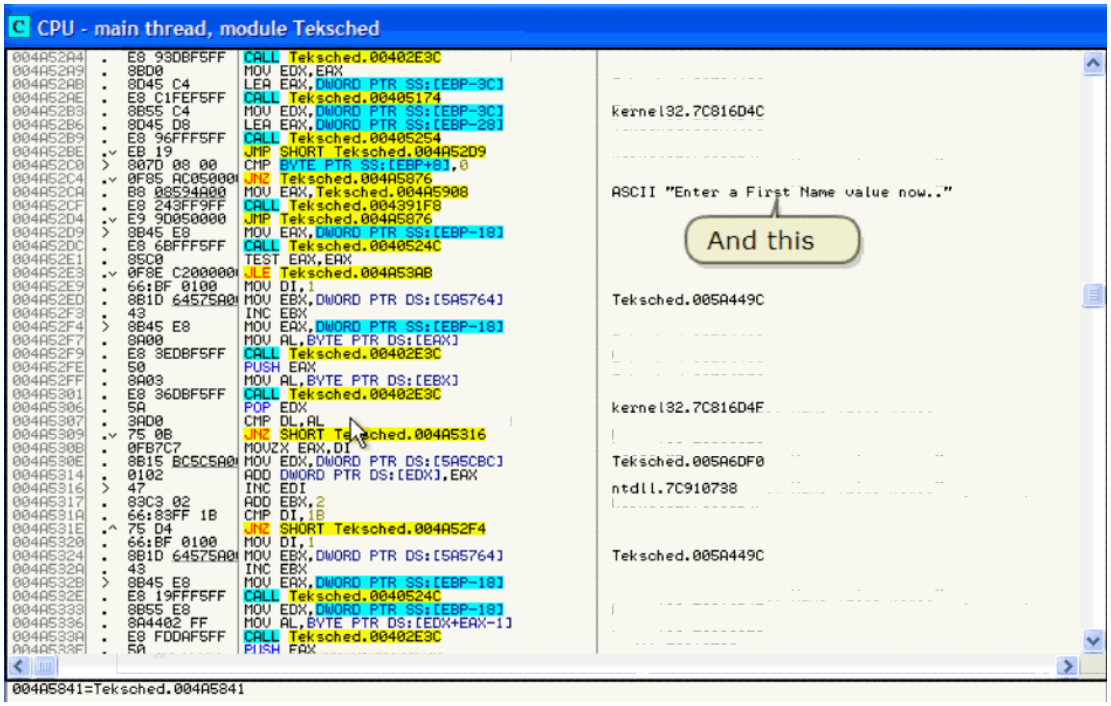
이것은 검증하는 것이다.

"messagebox 에 무언가 채워졌어?"(아니면 user 는 input 을 잊었다)

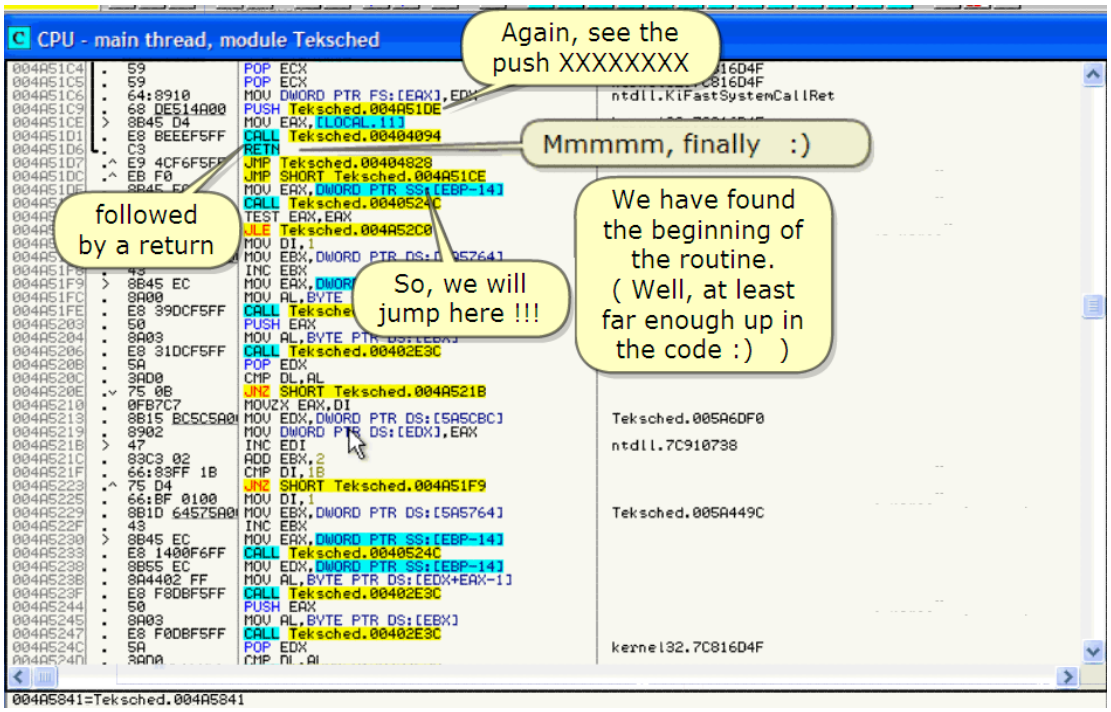


And this too

이것 또한



And this



Mmmmm, finally :)

음, 마지막으로 :)

We have found the beginning of the routine. (Well, at least far enough up in the code :))

우리는 routine 의 시작하는 부분을 찾았다. (좋아, code 에서 적어도 이 정도면 충분하다. :))

Again, see the PUSH XXXXXXXXXXXX

다시, PUSH XXXXXXXXXXXX 봐.

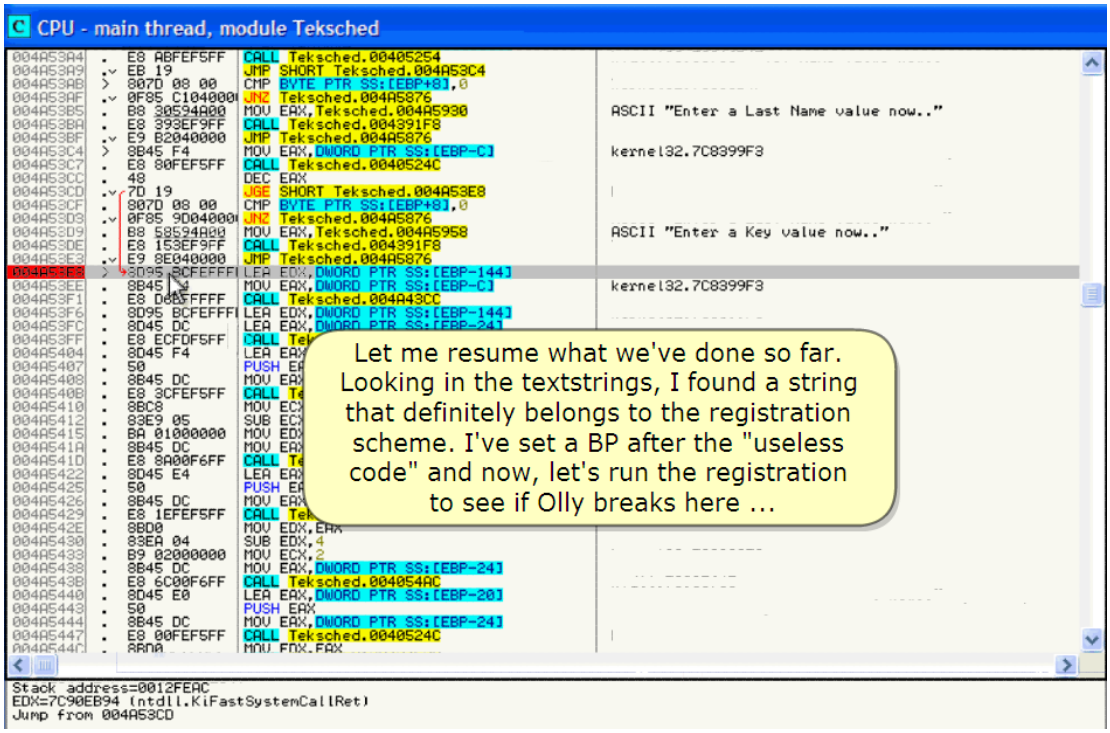
Followed by a return

Return 을 따라가.
So, we will jump here !!!
그래서, 우리는 여기로 jump 할 것이다!!!

Do you understand that effectively, this is NOT the beginning of the registration routine?
효과적으로 이해했어? 이것은 registration routine 의 시작이 아니다.
But I'm assuming that I'm far enough up in the code because I've passed the "Is something entered?" code already.
그러나 그것은 이 code 에서 충분하다고 생각한다. 왜냐하면 나는 이미 "무언가로 들어갔어?" code 를 지나쳤다.

Scroll down again
Scroll 다시 밑으로 내려.

...and place a BP after all the code
... 이 code 뒤에 BP 를 설치 해.
To verify "Is first name entered ?"
검증하기 "첫번째 이름 입력됐어?"
To verify "Is last name entered ?"
검증하기 "마지막 이름 입력됐어?"
To verify "Is key value entered ?"
검증하기 "key value 입력됐어?"



Let me resume what we've done so far.

내가 지금까지 한 일을 재개해

Looking in the textstrings, I found a string that definitely belongs to the registration scheme.

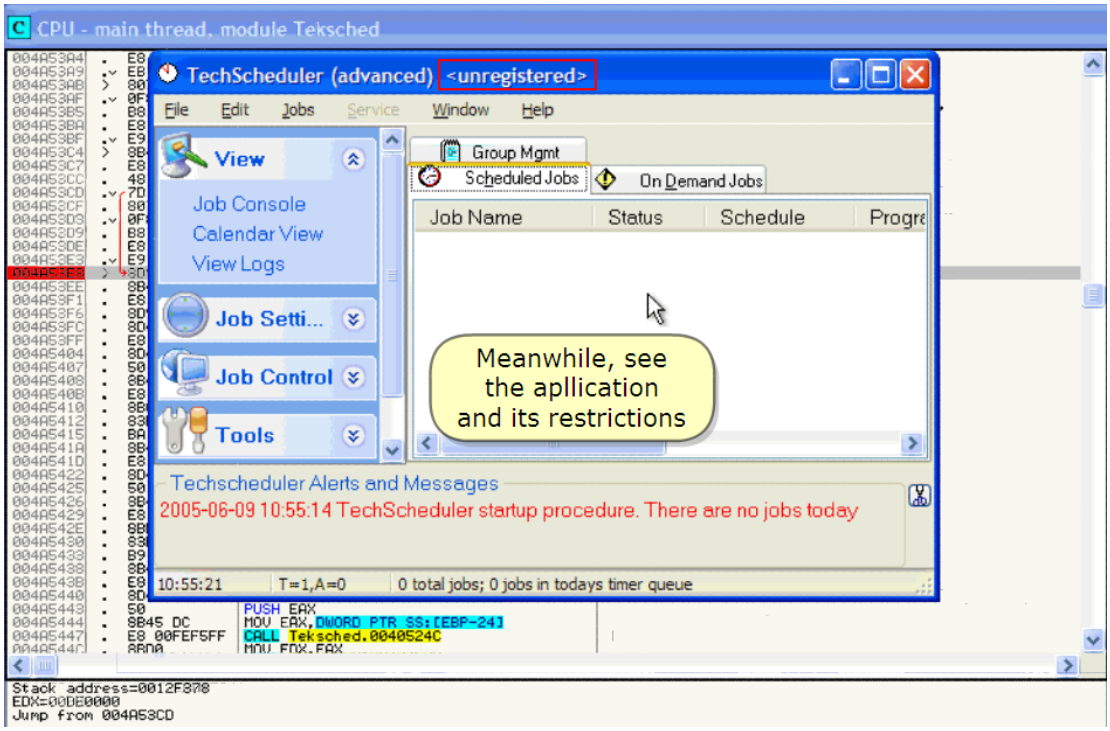
Textstrings 에서 봐. 나는 분명히 registration scheme 이 위치해 있는 string 을 찾았다.

I've set a BP after the "useless code" and now, let's run the registration to see if Olly breaks here ...

나는 "쓸모없는 code" 후에 BP 를 set 했다. Olly 가 여기에서 멈춰 있을 때 보기 위해 Registration 을 실행 해.

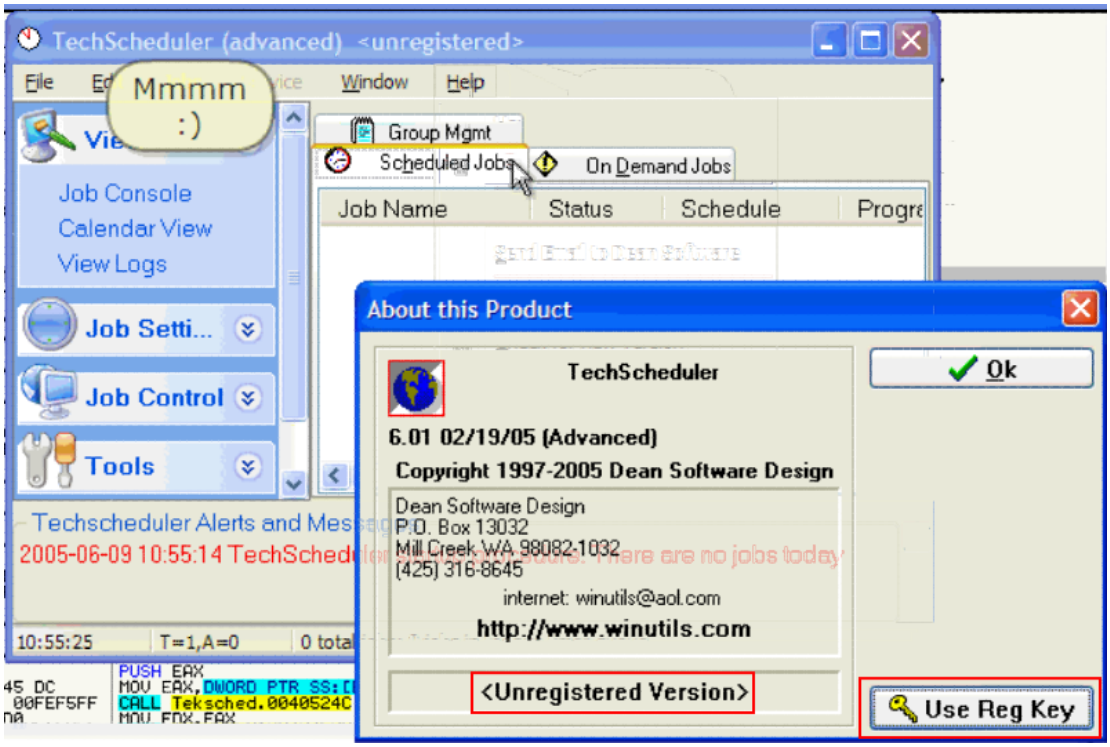
And run

실행



Meanwhile, see the application and its restrictions

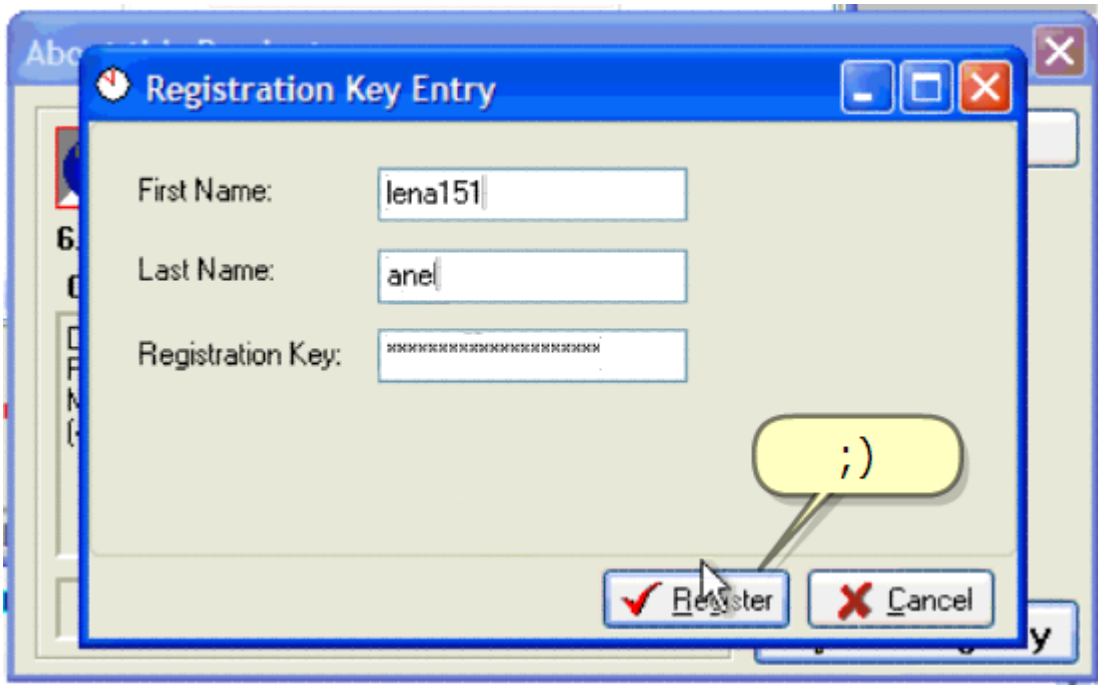
Application 을 봐. 그리고 그것은 제한적이다.



Mmmm :)

:(

;)

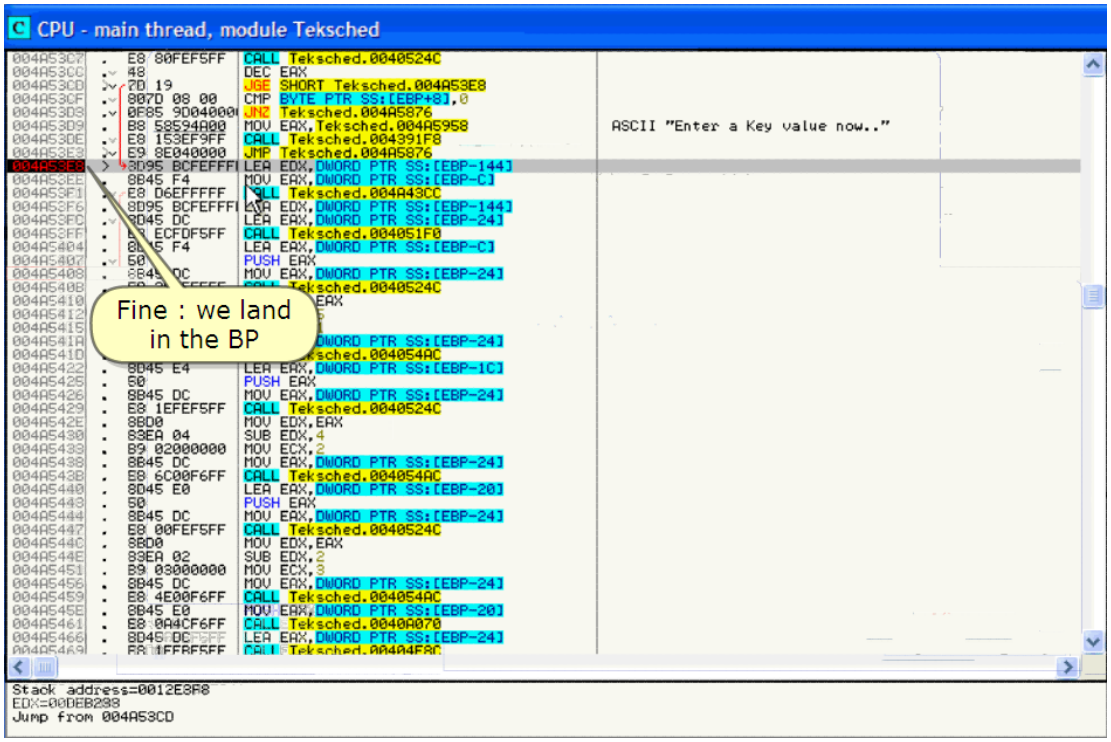


That's also me, but backwards :)(anel)

그것도 나지만, 거꾸로

번역 주)lena -> anel

;)



Fine : we land in the BP

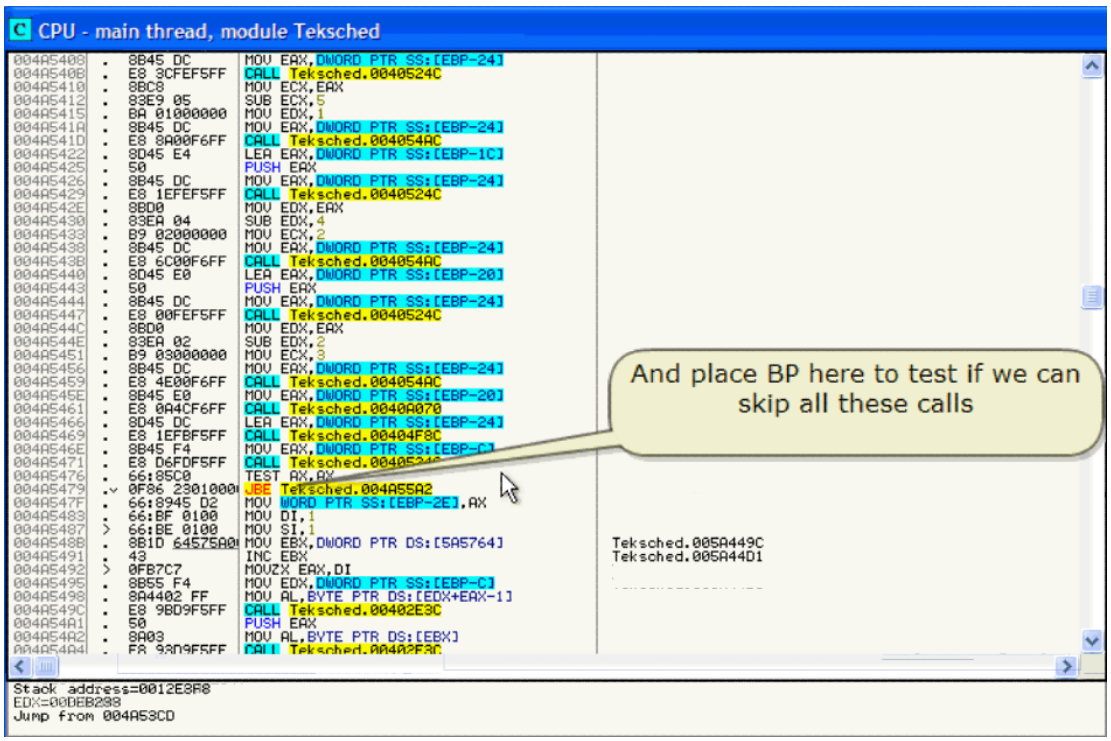
좋아 : BP 에 도착했다.

Remove BP

BP 제거.

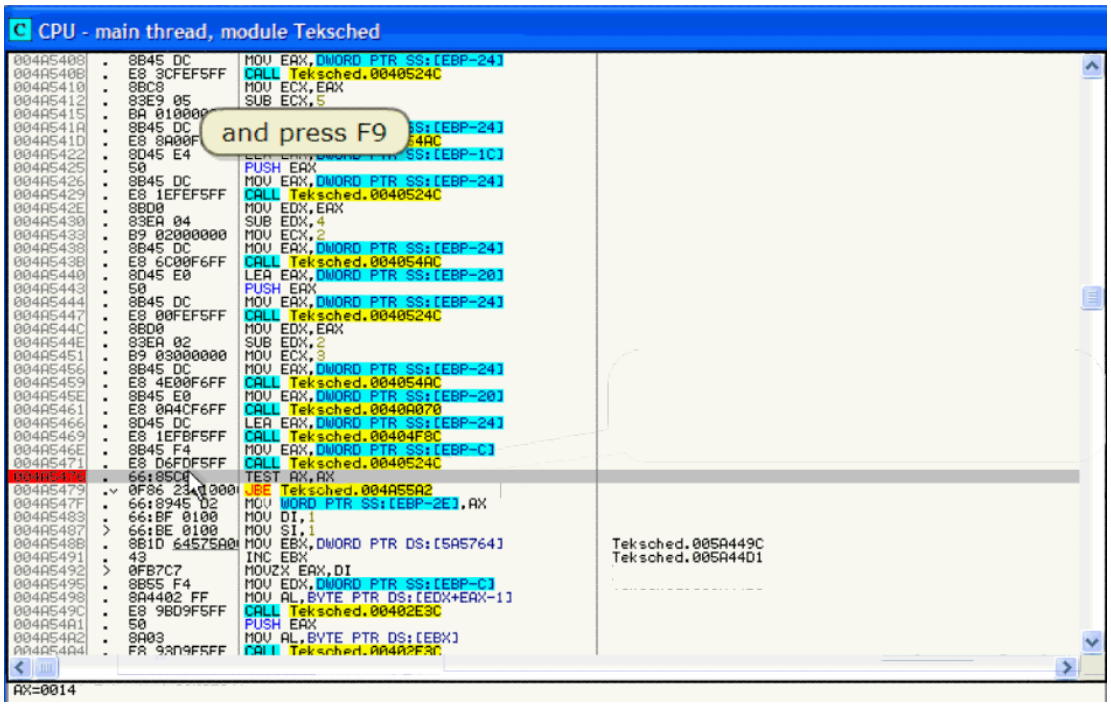
Scroll down

Scroll 내려.



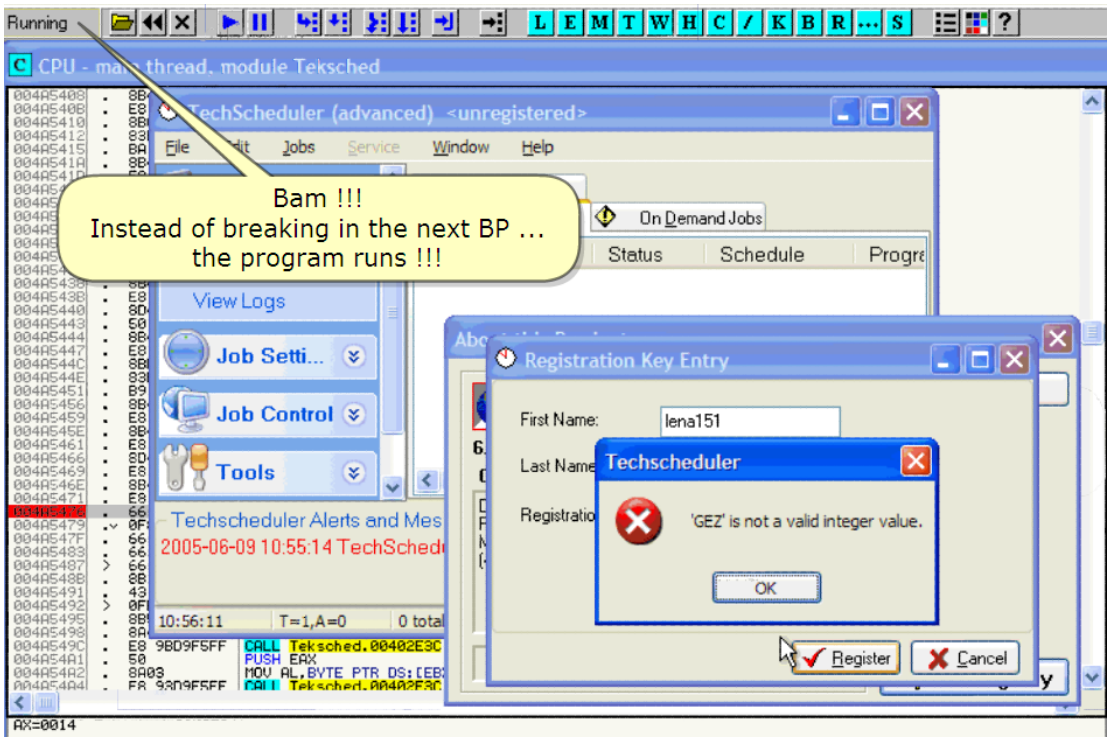
And place BP here to test if we can skip all these calls

그 calls 을 skip 할 것인지 test 하기 위해 여기에 BP 설치 해.



And press F9

F9 를 눌러.

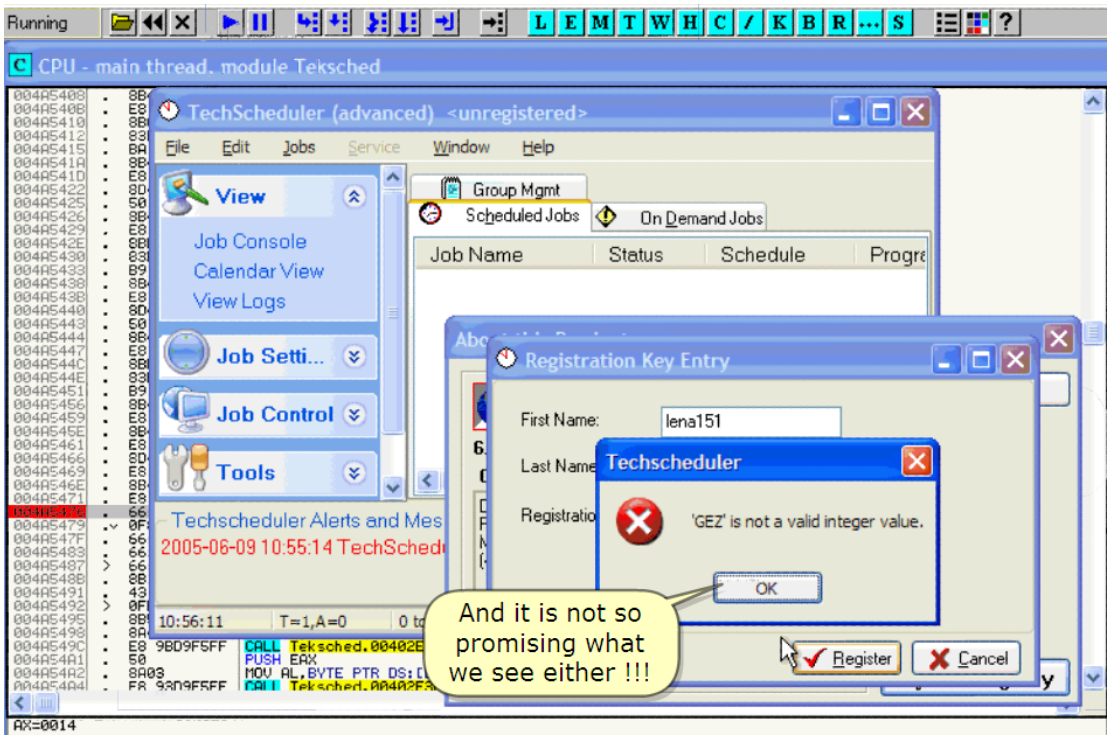


Bam !!!

오 !!!

Instead of breaking in the next BP ...

다음 BP 대신에 멈췄다.
The program runs !!!
Program 실행중 !!!

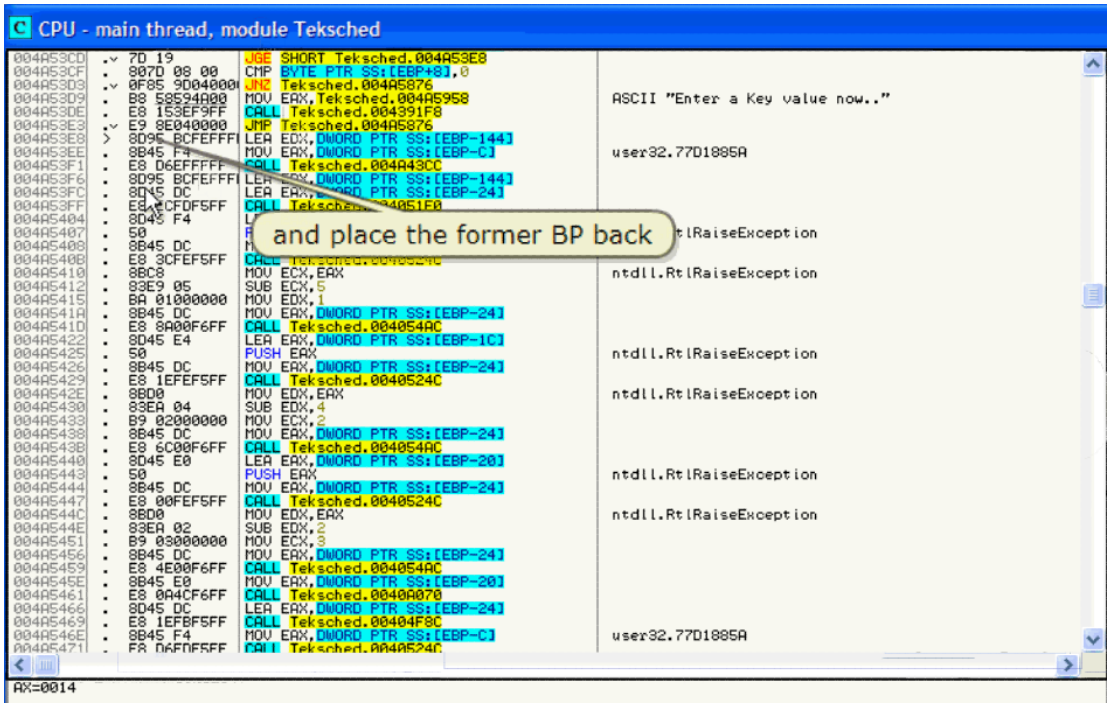


And it is not so promising what we see either !!!
우리가 보는 것 또한 유망하지 않다.

INFO :
This is suspicious for the program detecting some mistake somewhere.
이것은 어딘가에 몇 가지 실수를 감지하는 program 이 의심된다.
Always be suspicious for code integrity checks but it is not in this case.
항상 code 의 진실성 check 가 의심된다. 이 case 에서는 그렇지 않다.
You can check that by removing the BP's and verifying.
너는 BP 를 삭제하는 것과 검증하는 것을 check 할 수 있다.
You get the same error message.
너는 똑같은 error message 를 얻는다.
In this case, it's the program's bizarre way of telling us we used a bad serial.
이 경우, 이것은 program's 은 우리에게 전하는 기이한 방법이다. 우리는 bad serial 을 사용했다.
BTW, programs can also check for software BP's !!!!
By the way, programs 은 software BP's 를 위해 check 할 수 있다. !!!!
We will see all this more in detail in later parts in this series.
우리는 이 series 의 나중 part 에서 상세하게 이 모든 것을 좀 더 볼 수 있다.

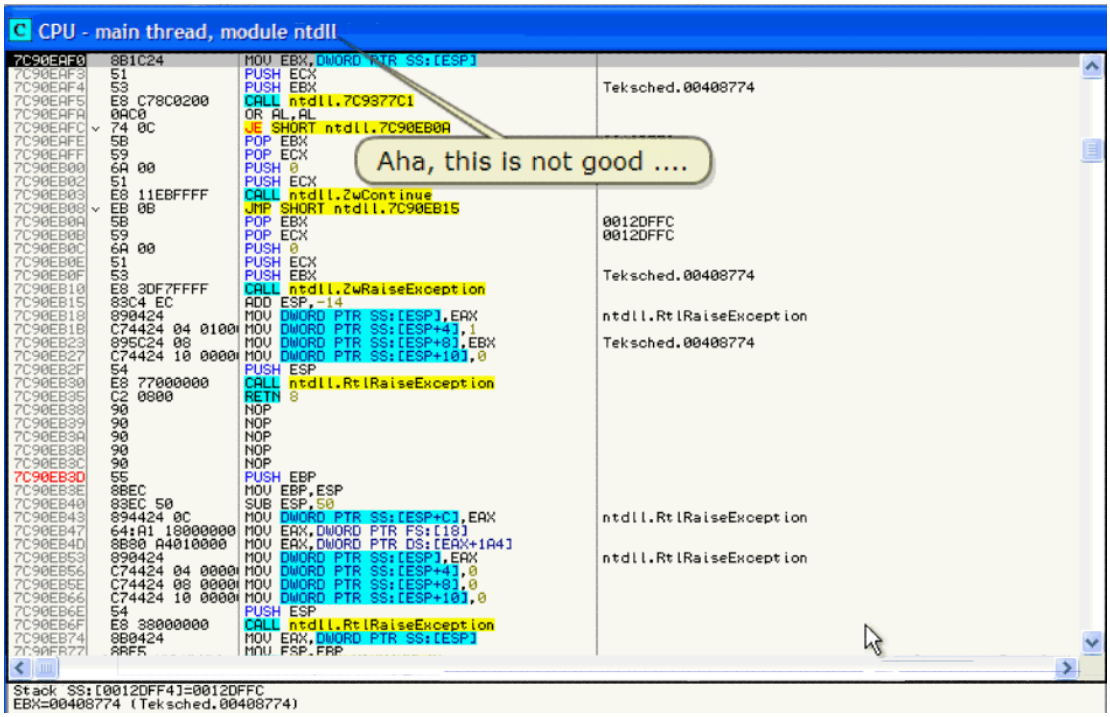
Let's find out where the program has detected the mistake ....
Program 이 어디에서 잘못 발견했는지 찾아보자.

Return to Olly
Olly 로 돌아가자.
(click in Olly)
(Olly 에서 click 해)
Scroll up ...
Scroll 올려 ...



And place the former BP back
옛날 BP 에 설치 해.
번역 주)어디에서 잘못됐는지 찾기 위해 예전에 BP 를 설치했다 삭제한 곳에 다시 BP 를 설치합니다. 그리고 어디에서 잘못됐는지 trace 합니다.

If we run the registration again, Olly will break in the BP again
다시 registration 을 실행한다면, Olly 는 다시 BP 에서 멈출 것이다.
:)
And step F8 to see what is going on where ...
어떻게 되는지 보기 위해 F8 을 눌러.



Aha, this is not good ....

아하, 이것은 좋지 않은데 ...

Right, press <-> to see what caused this

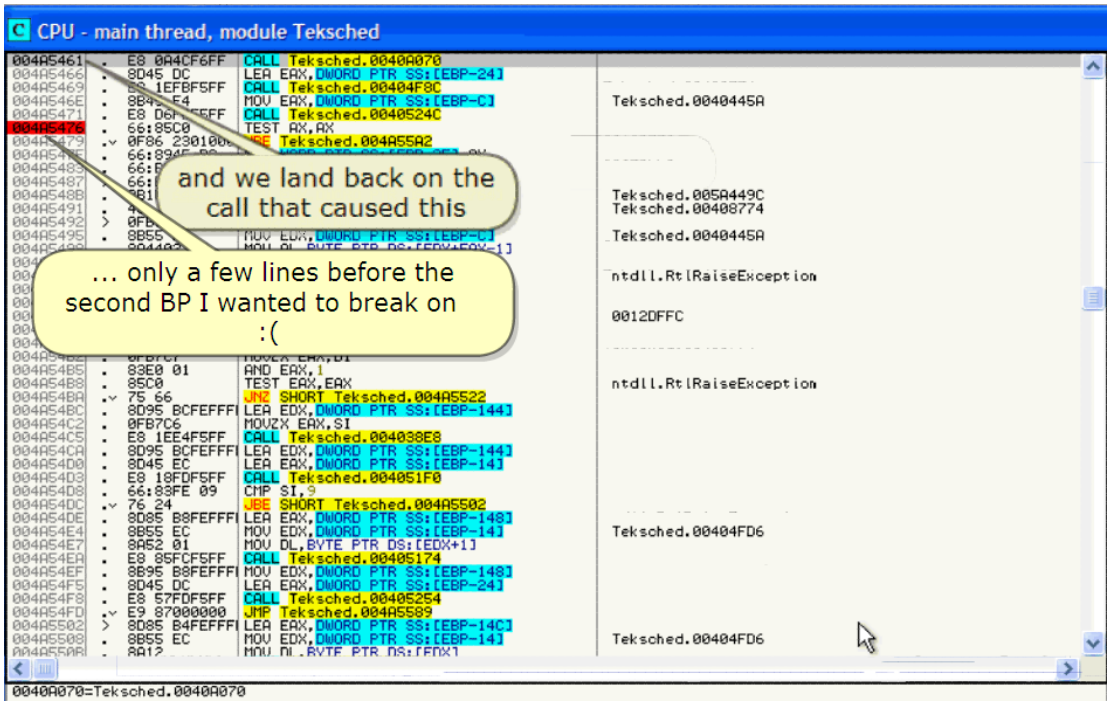
좋아, 눌러 <-> 이것이 어디에서 발생했는지 보기 위해

번역 주)마이너스(-): 자기가 code 를 실행하고 있는데 갑자기 다른 곳으로 왔다면 -를 눌러. 왔던 곳으로 돌아갈 수 있다.

INFO :

Pressing the minus sign on your keyboard lets you return on your steps(without effectively undoing them of course)

너의 keyboard 의 Minus sign 을 눌러. 네가 진행해 왔던 곳으로 돌아가게 할거야(효과적으로 그것들을 바로 잡지는 않는다)

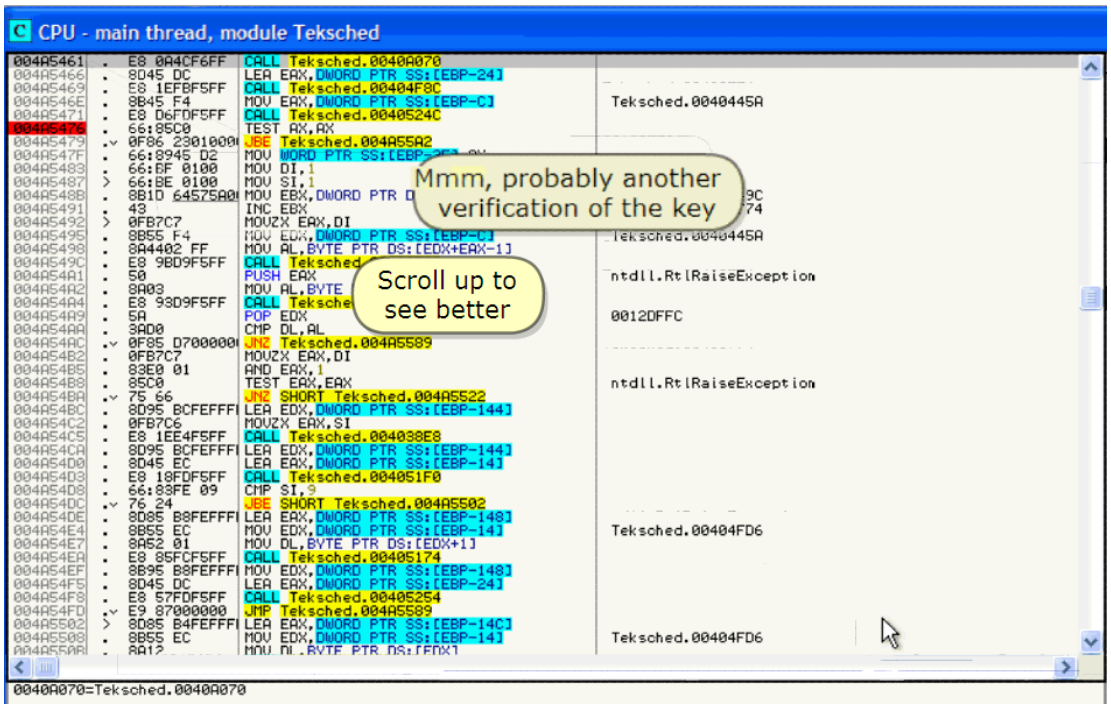


And we land back on the call that caused this

발생했던 Call 로 돌아올 것이다.

... only a few lines before the second BP I wanted to break on :(

내가 원할 때 멈출 수 있는 2 번째 BP 전에 오직 몇 line 이 있다.



Mmmm, probably another verification of the key

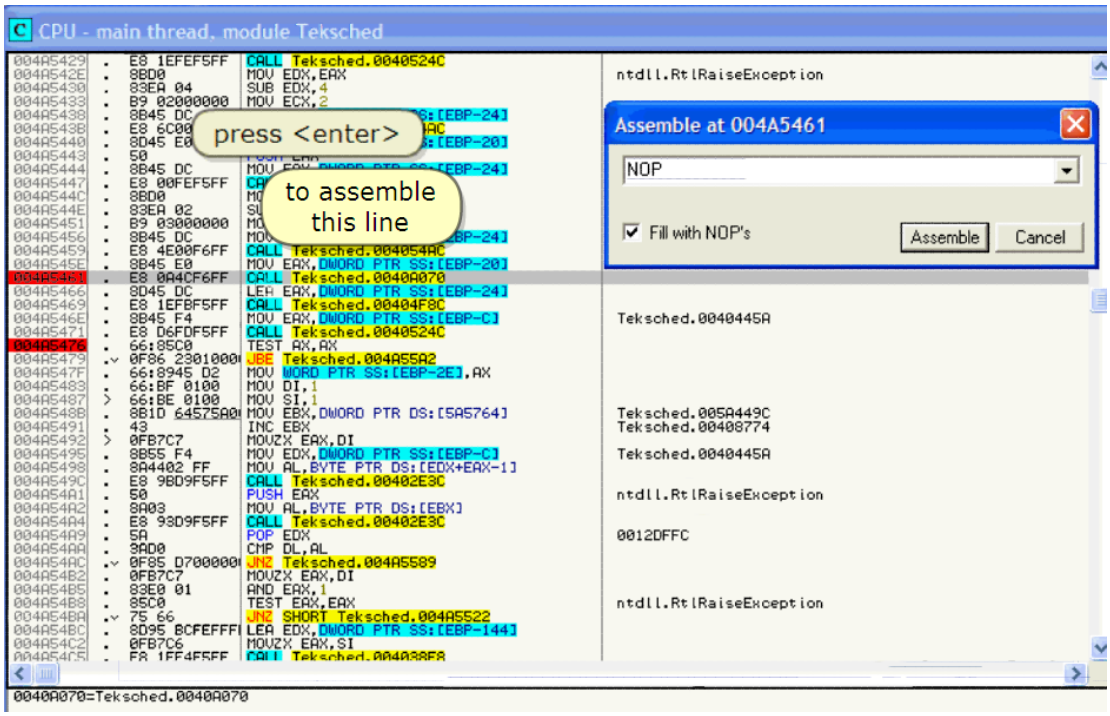
Scroll up to see better

음, 아마 다른 key 검증이 있을 것이다.

좀더 보기 좋게 Scroll 올려.

Mmmm,
Let's just NOP the call and return to see what happens.
음,
Call 을 NOP 하고 무슨 일이 일어나는지 보기 위해 return 해.

INFO :
In later parts, we will dig deeper into this kind of problems.
나중 part 에서, 우리는 이 종류의 문제에 깊이 파고들 것입니다.
This means to step IN the call that caused this of course :)
물론 이것이 발생했던 Call 로 들어가자.



Press <enter>
To assemble this line
Enter 를 눌러.
이 line 을 assemble 하기 위해

And remove the BP in the changed code
바뀐 code 에서 BP 를 삭제 해.
And set it just above
이 위를 바로 set 해.
Right
좋아

INFO :

This is only to avoid having to re-enable the BP's at restart(see previous parts)

이것은 restart 할 때 오직 BP 를 재사용을 회피할 때 사용한다.(이전 part 를 봐)

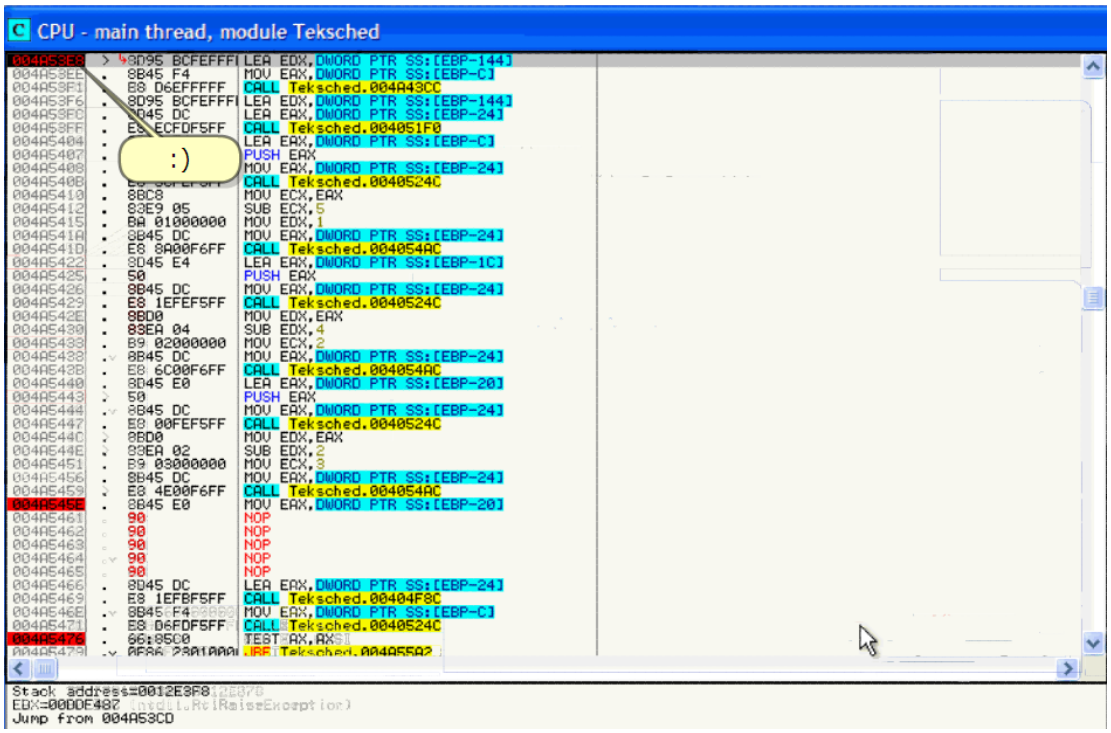
번역 주)restart 해서 BP 가 쓸모 없을 때 사용한다.

Run because the program is paused

실행 해. 왜냐하면 program 이 멈춰있다.

Yes, we were already in the call that causes this

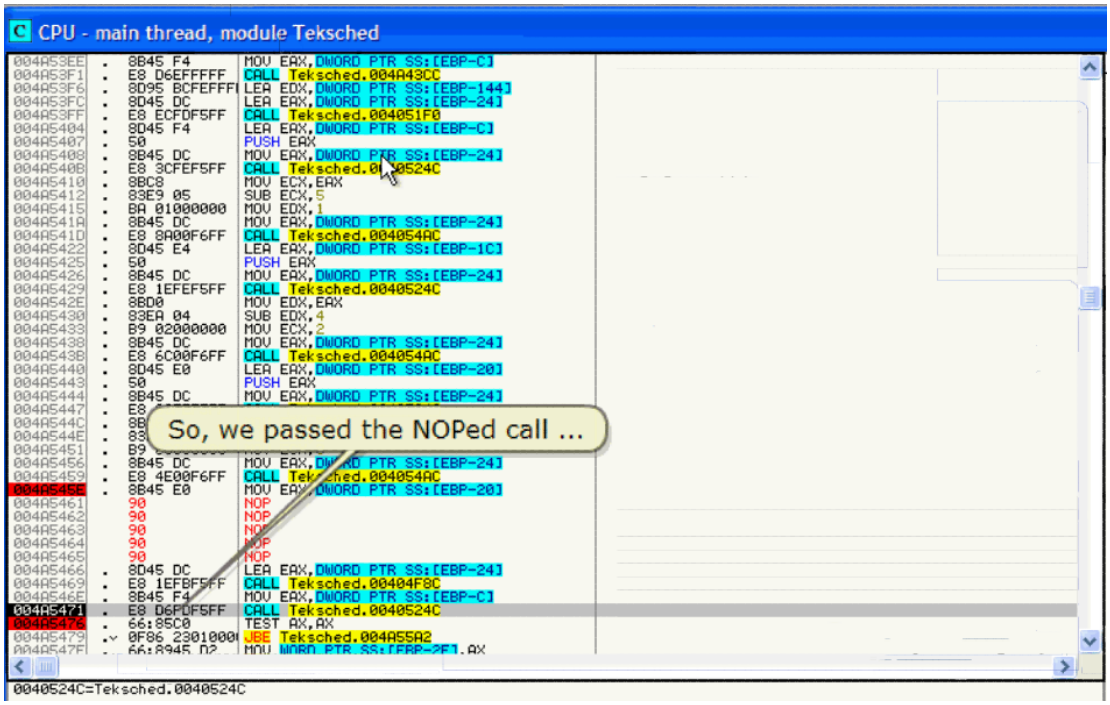
예, 우리는 이미 발생한 call 에 있다.



:)

Ok. Now let's step F8 to see if there are no errors by NOP'ing the call

좋아. 이제 NOP'ing 된 call 에 의해 아무 error 가 없는지 보기 위해 F8 을 눌러.

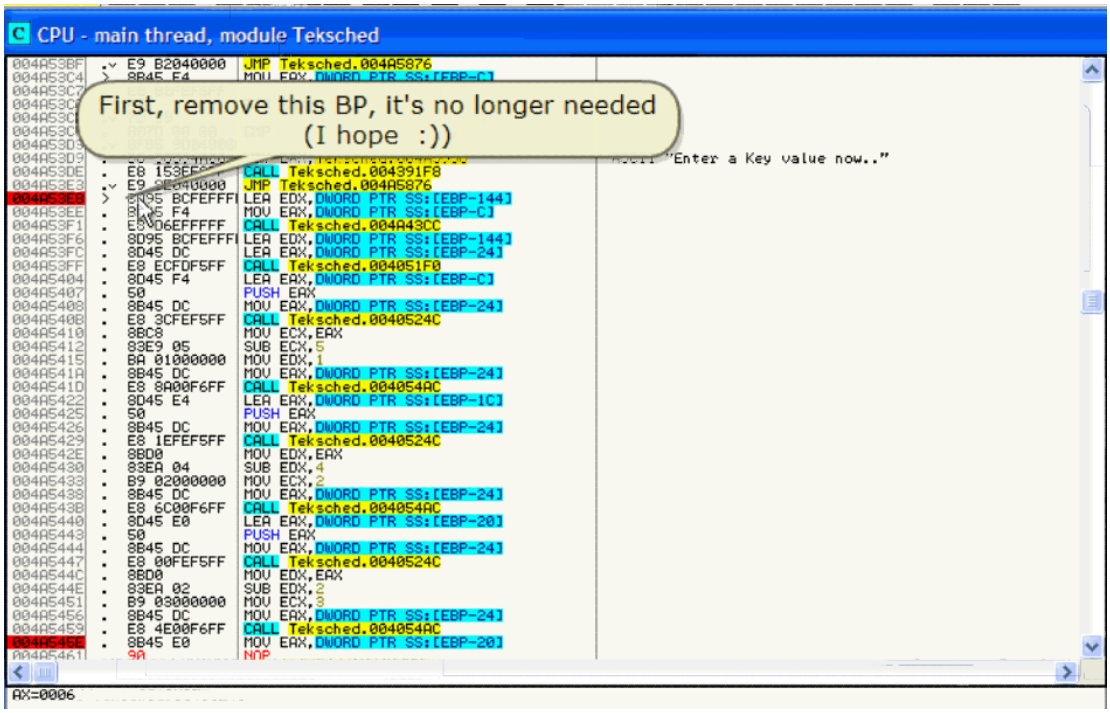


So, we passed the NOPed call ...

그래서, NOP 된 call 을 피한다.

Mmmm, ok, scroll up

음, ok scroll 올려.

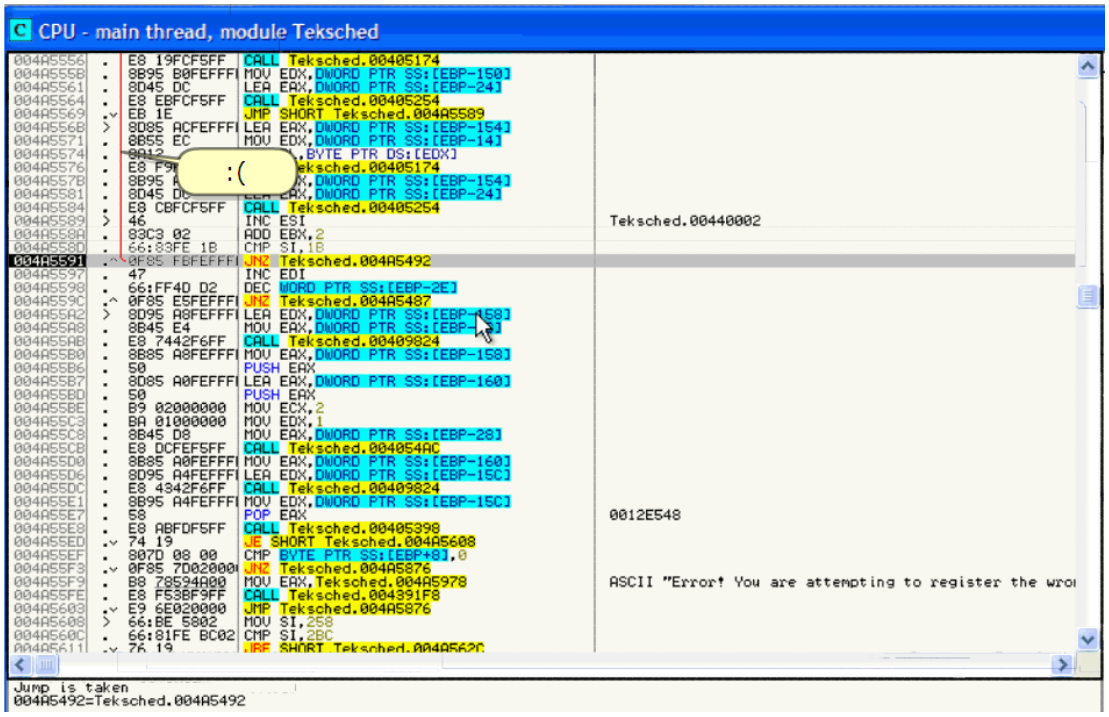


First, remove this BP, it's no longer needed(I hope :))

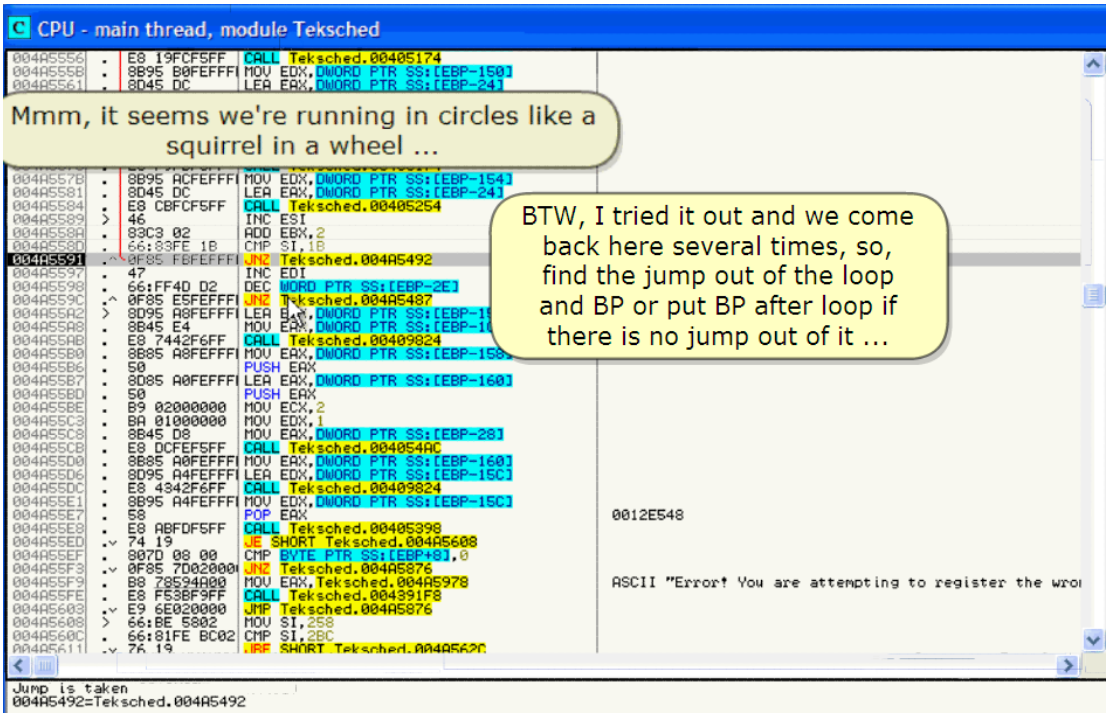
먼저, 이 BP 를 삭제 해, 더 이상 필요하지 않다.(나는 희망한다 :))

Continue F8

F8 눌러.



:(



Mmm, it seems we're running in circles like a squirrel in a wheel ...

음, 우리가 circle 에서 실행되는 것으로 보인다. 마치 다람쥐가 쳇바퀴 도는 것처럼.

BTW, I tried it out and we come back here several times,

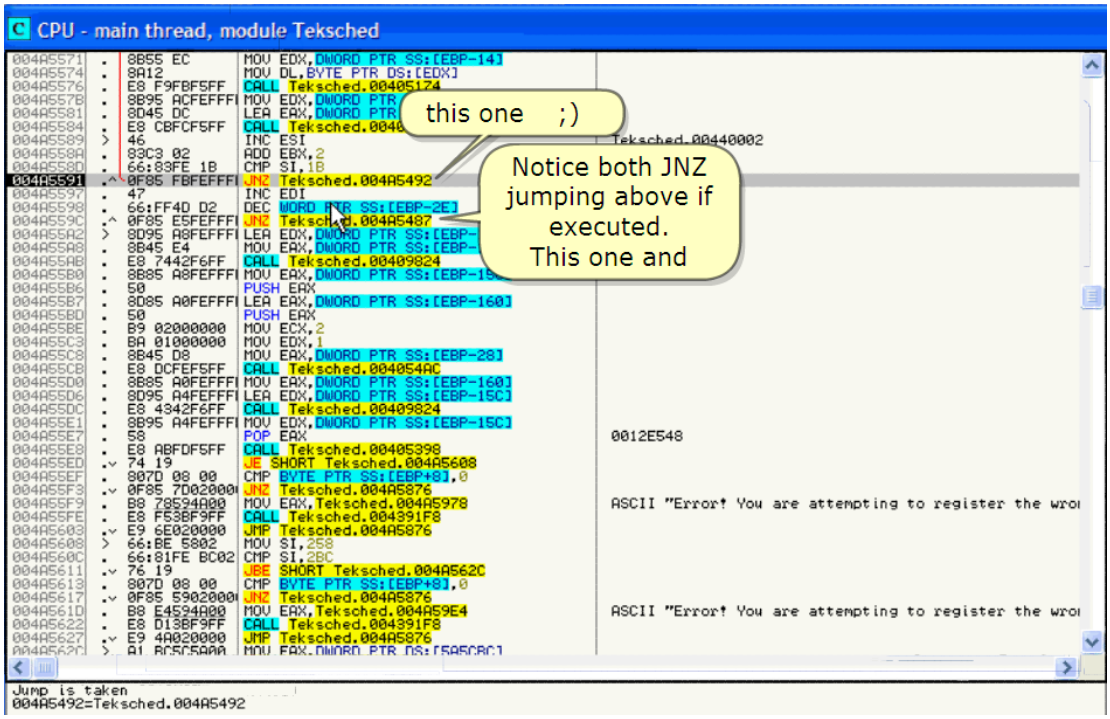
By the way, 우리는 이것에 도전 했다. 그리고 우리는 많은 시간 동안 여기로 돌아왔다.

so, find the jump out of the loop and BP or put BP after loop if there is no jump out of it ...

그래서, loop 를 탈출하는 jump 를 찾았다. 그리고 만약에 이것을 벗어나는 jump 가 없다면 loop 후에 BP 를 설치했다.

Ok, BP and run ...

Ok, BP 설치하고 실행 해 ...

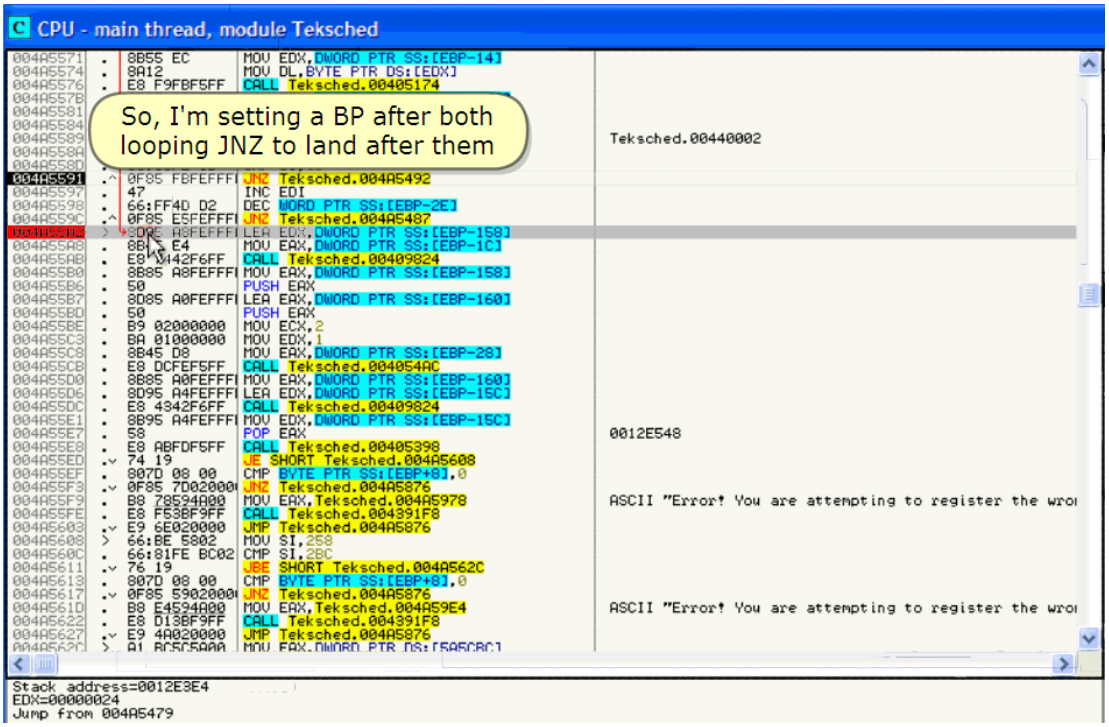


Notice both JNZ jumping above if executed.

This one and this one ;)

실행될 경우, JNZ 가 모두 위에 있습니다.

이것 하나와 이것 하나다 ;)



So, I'm setting a BP after both looping JNZ to land after them

그래서, 나는 2 가지 JNZ looping 이 끝나는 곳에 BP 를 설치한다.

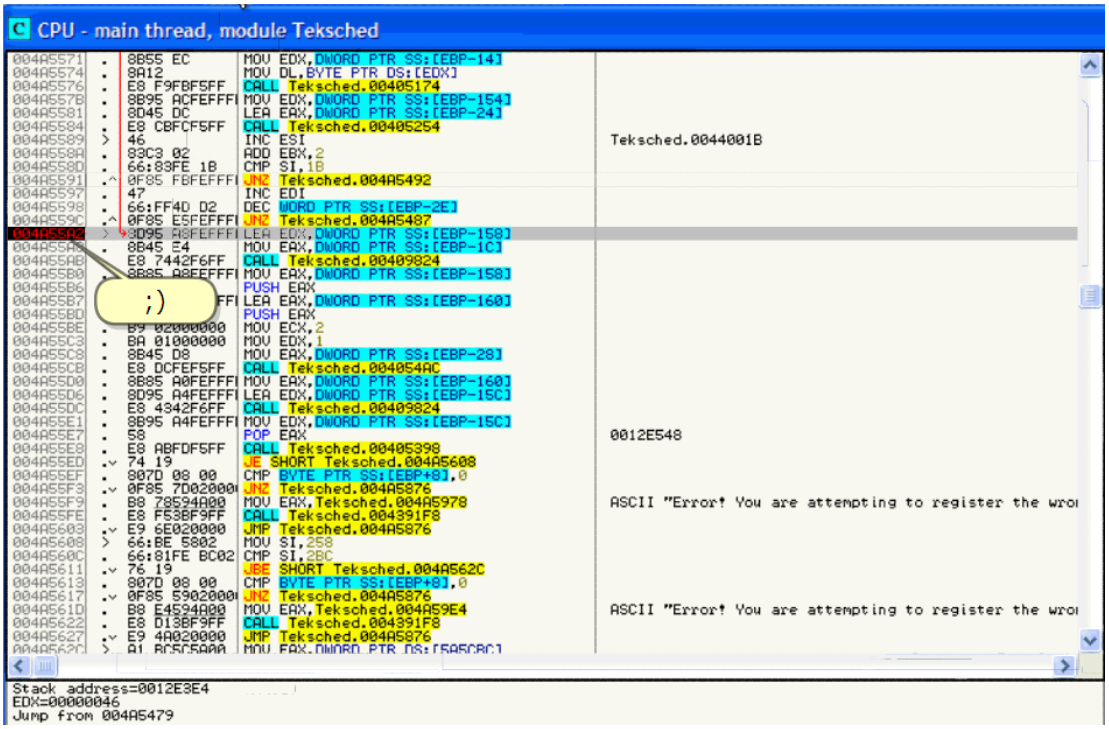
INFO :

It is very well possible that the right serial is calculated in a loop like this.

이것은 매우 잘 될 가능성이 있다. 옳은 serial 은 loop 에서 이것처럼 계산됐다.

I'm not interested in it today, I've not even verified it :)

오늘은 이것에 흥미가 없다, 나는 이것조차 검증하지 않았다 :)



;)

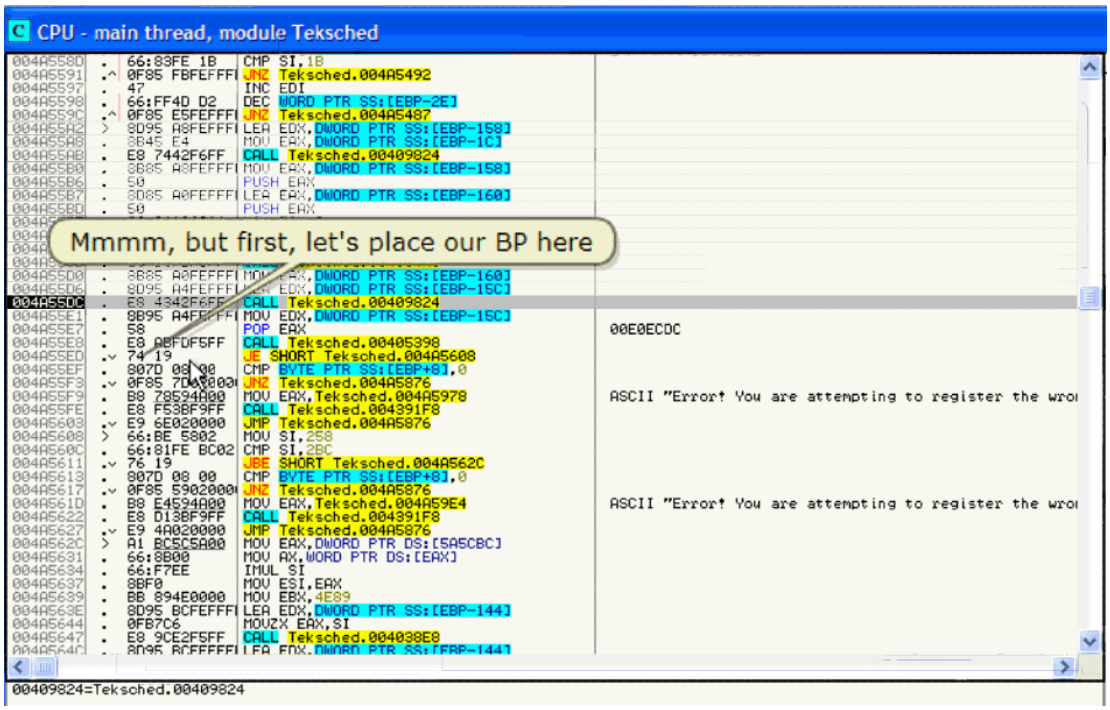
번역 주)2 가지 loop 가(역행하는) 끝난 후에 BP 를 설치하고 F9 를 눌러서 실행했더니 정상적으로 넘어왔다. 이렇게 해서 다람쥐 쳇바퀴 도는 것을 한 방에 끝낼 수 있다.

Now that the loops are history, continue stepping F8 again

이제 loops 는 지나갔다. 다시 F8 로 계속 진행하자.

Now it gets interesting

이제 이것에 흥미가 있다.



Mmmm, but first, let's place our BP here

음, 먼저 우리의 BP 를 이곳에 설치하자.

번역 주)Error message 가 있으니까 이것을 잘 회피하면 goodboy 에 갈 수 있으니까 흥미롭습니다!

Because so far, nothing interesting has happened after the NOPed call

지금까지, NOPed call 후에 흥미가 없었다.

And if a re-run should be necessary ...

재시작 하는 것이 필요하다면 ...

It's always interesting to remember that one can run till the next BP

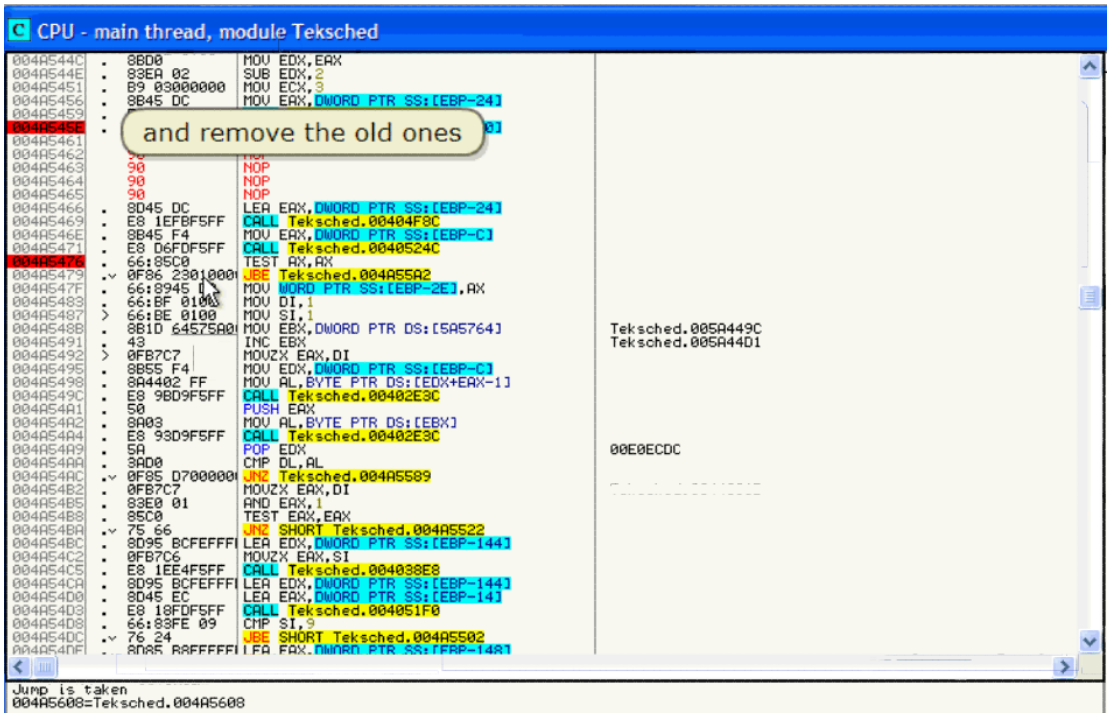
항상 다음 BP 까지 실행할 수 있다는 것을 기억하는 것이 흥미롭다.

Ok. We won't need the other BPs any longer.

Scroll up ...

Ok. 우리는 더 이상 다른 BPs 가 필요하지 않다.

Scroll 올려 ...

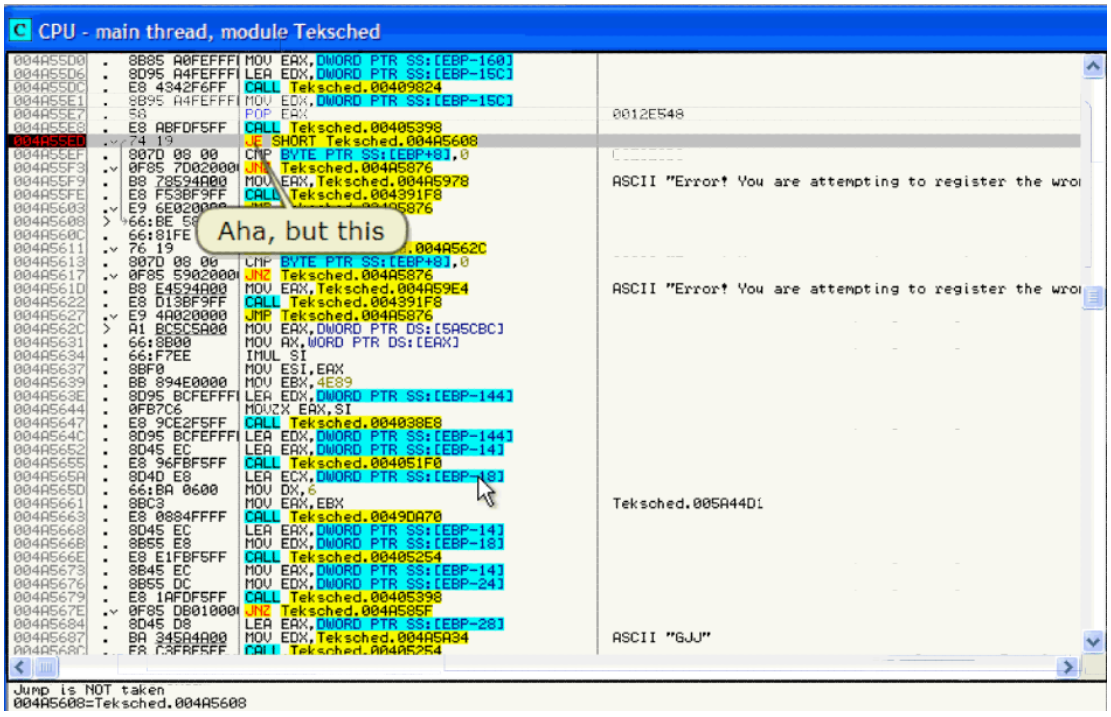


And remove the old ones

오래된 것들을 삭제 해.

Continue F8

F8 눌러.

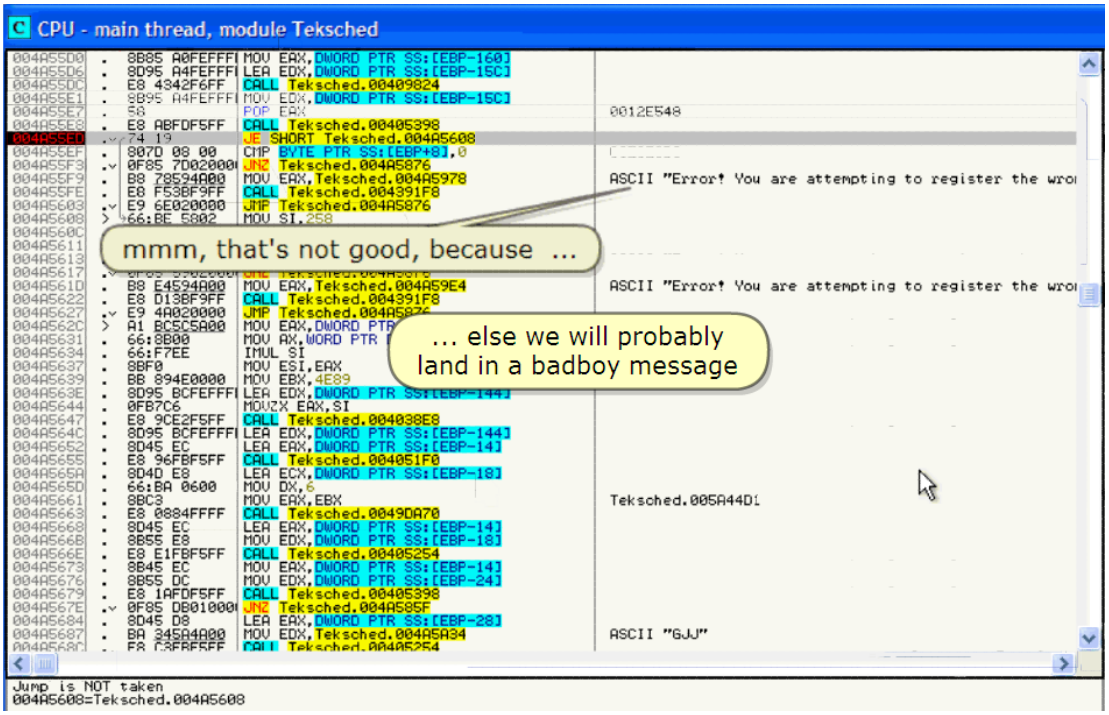


Aha, but this

아하, 그러나 이것은

Not being executed ....

실행되지 않는다.

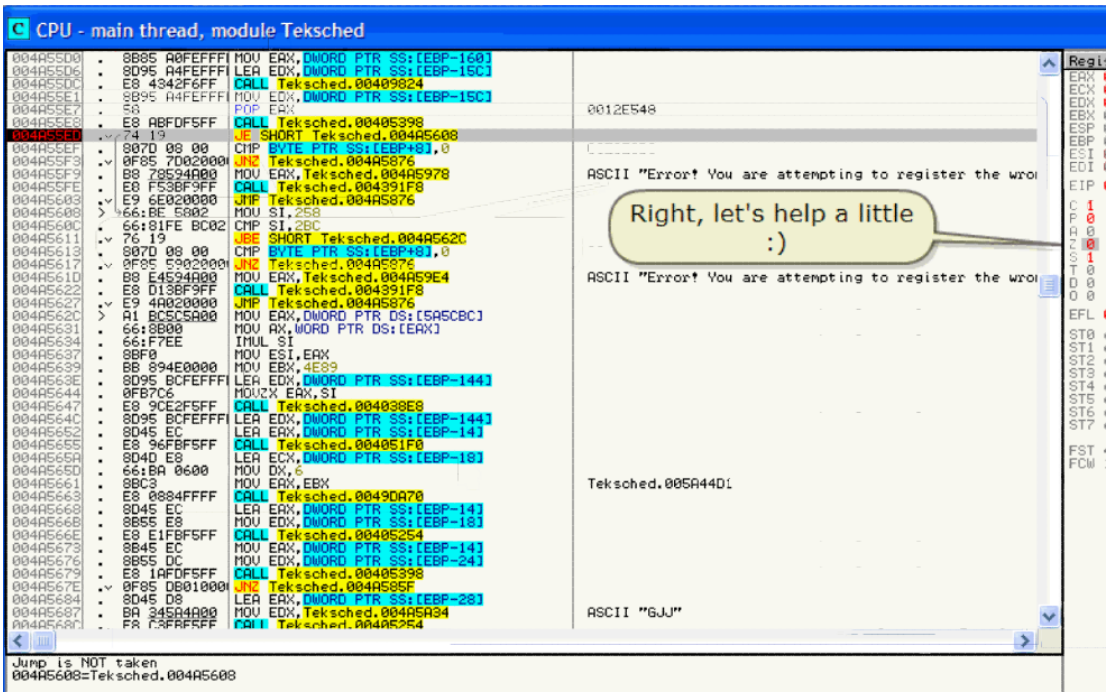


Mmm, that's not good, because ...

음, 이것은 좋지 않아. 왜냐하면...

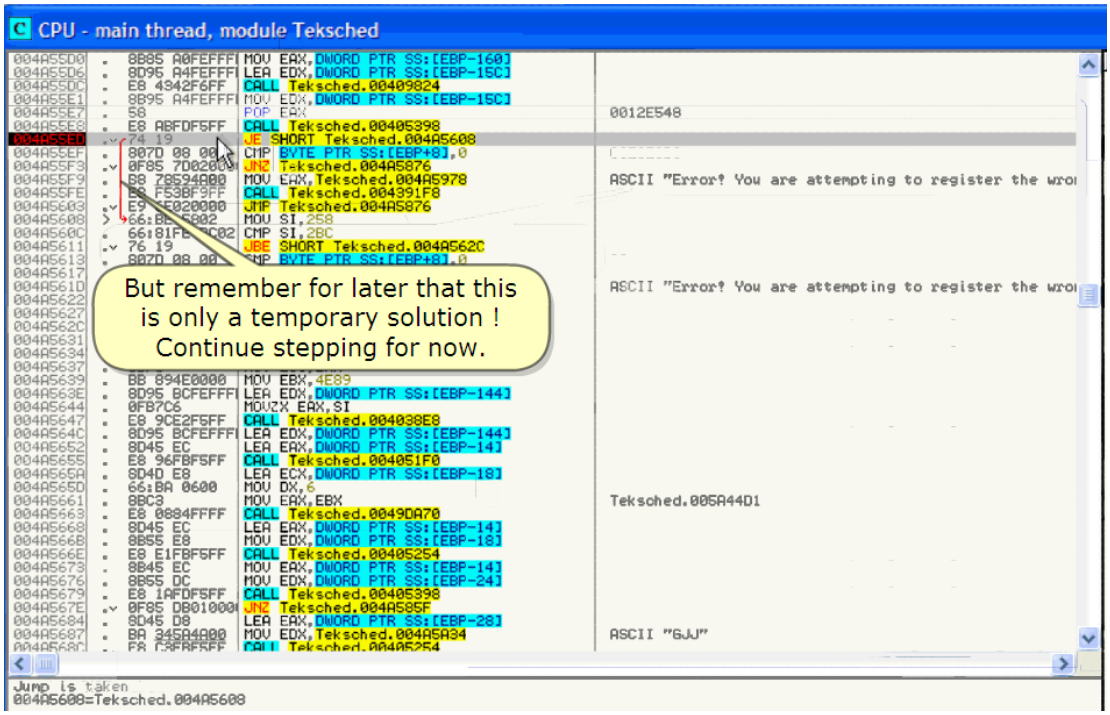
... else we will probably land in a badboy message

우리는 아마 badboy message 에 도착할 거야.



Right, let's help a little :)

좋아, 여기에서 도움을 얻자 :)

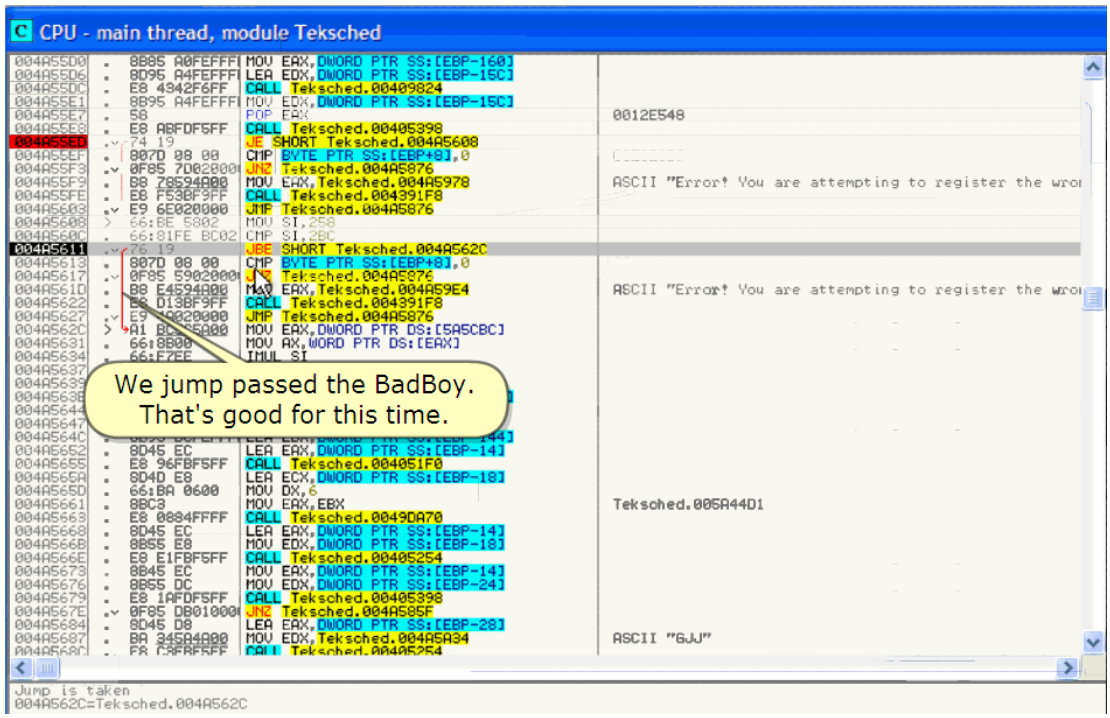


But remember for later that this is only a temporary solution!

그러나 나중을 위해 기억해. 이것은 오직 임시적인 해결책이야!

Continue stepping for now.

이제 계속하자.



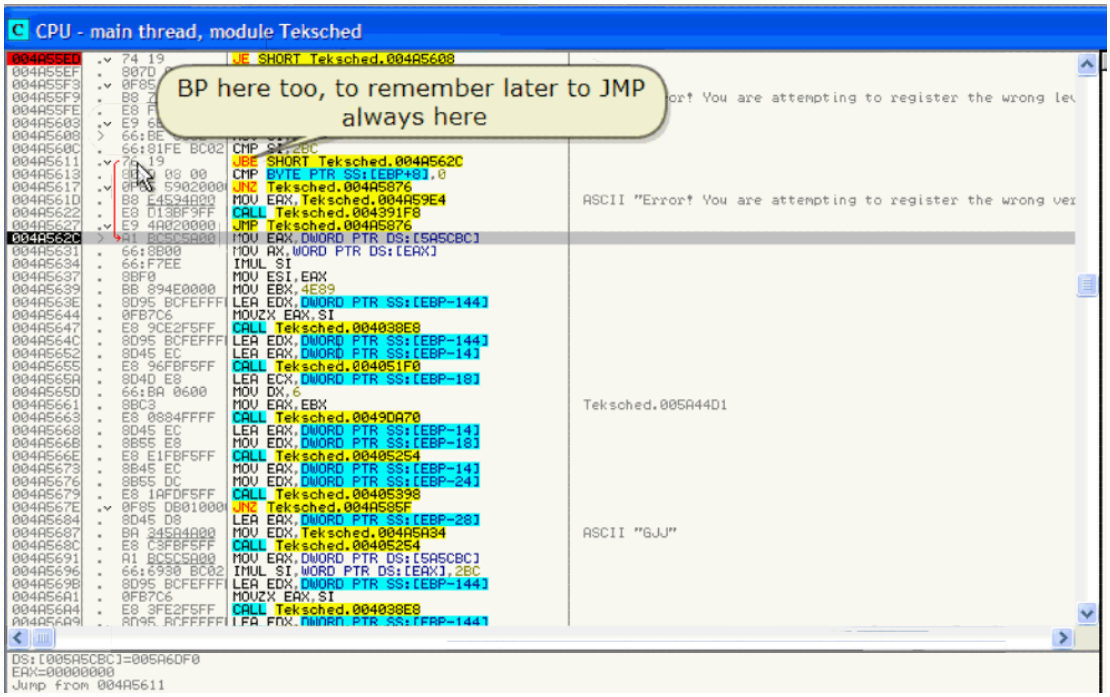
We jump passed the BadBoy.

That's good for this time.

우리는 지나가 버린 badboy 를 jump 한다.
좋아.

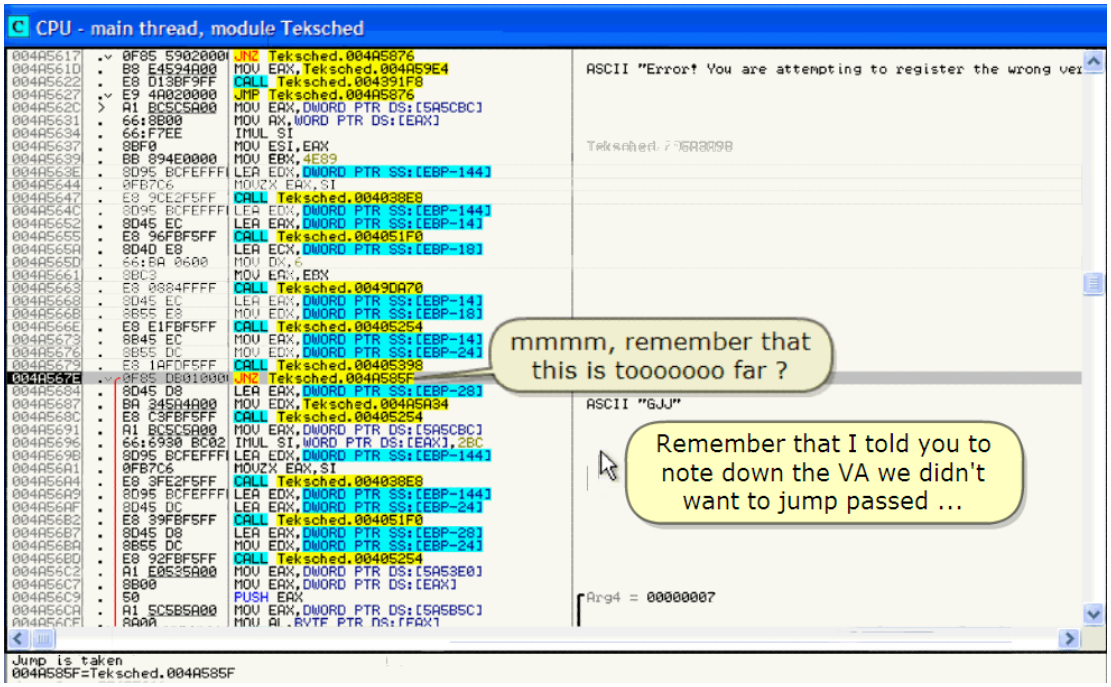
Now, think with me ...
If this JBE doesn't get executed for one or another reason if we are registered
이제, 나랑 같이 생각해보자.
우리가 등록하기를 원할 때 만약에 한 가지 이유나 다른 이유로 이 JBE 가 실행되지 않는다면

...i.e. after some amount of time or whatever .....
꽤 많은 시간이 지나거나 무엇이든지 ....
So, for extra security, let's ...
그래서, 추가적인 보호가 필요하다.



BP here too, to remember later to JMP always here
항상 JMP 하기 위해 BP 를 여기에 설치 해,

Continue stepping F8
F8 로 계속 해.



Aha, but here it gets serious ...

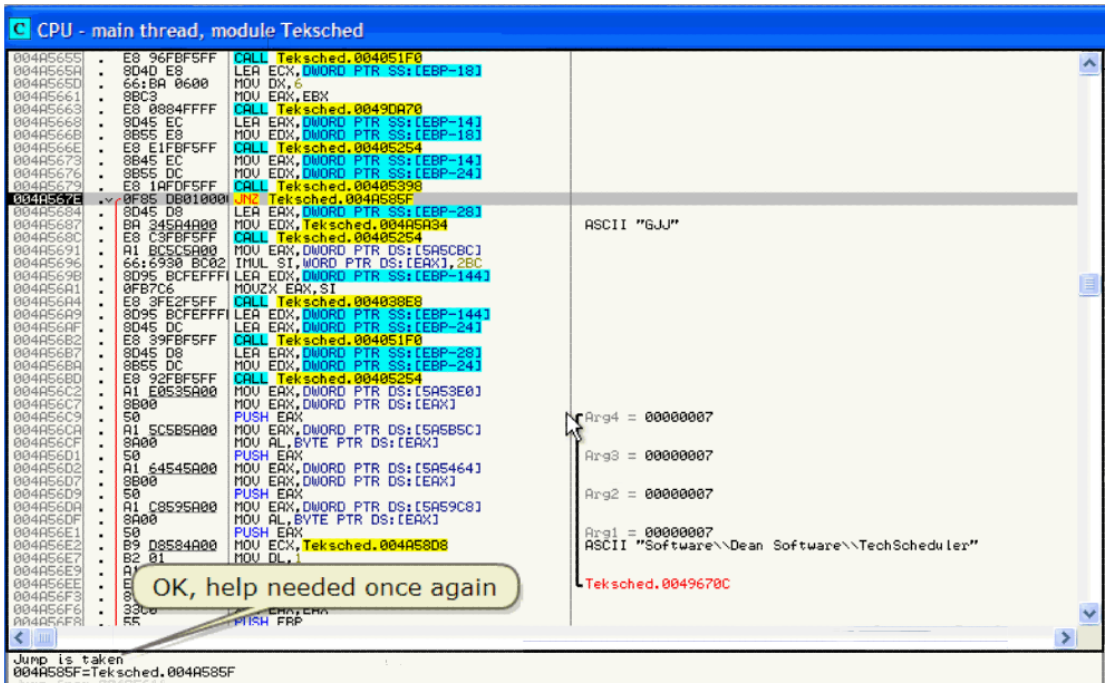
아하, 좀 더 진지하게 얻을 수 있다.

Mmmm, remember that this is tooooooo far ?

음, 이것이 너무 멀다는 것을 기억해?

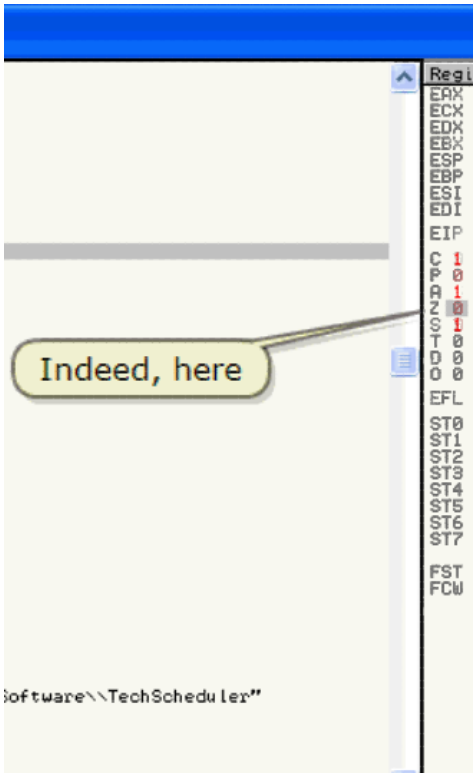
Remember that I told you to note down the VA didn't want to jump passed ...

VA 를 적으라고 했던 것을 기억해. VA 는 jump 를 원하지 않는다.



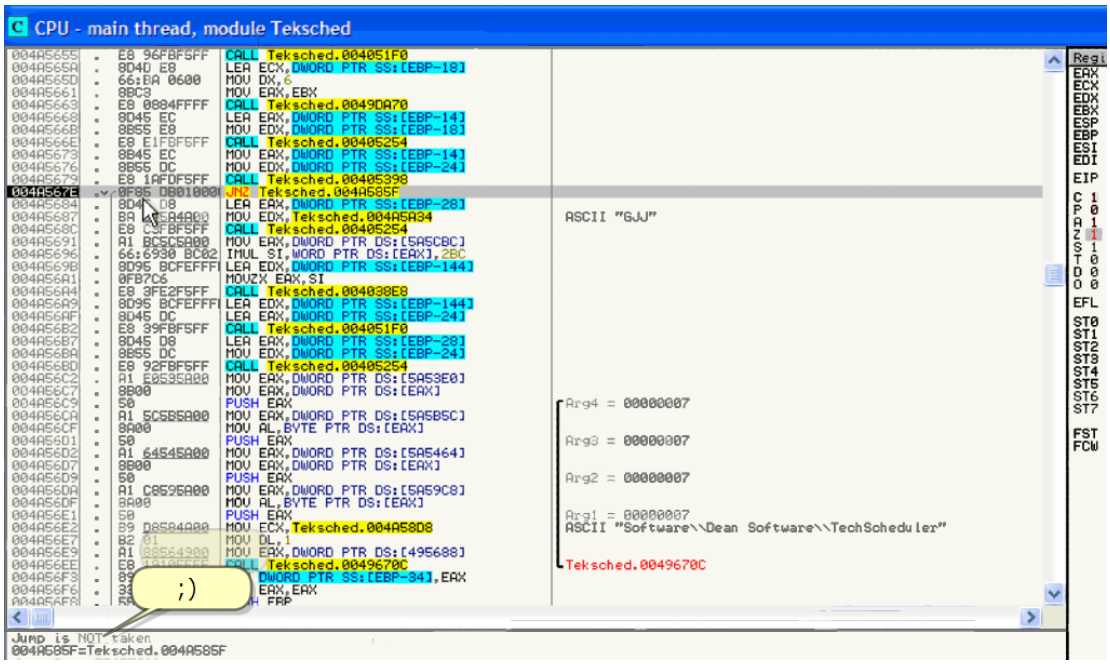
OK, help needed once again

Ok, 다시 한 번 도움을 얻었다.



Indeed, here

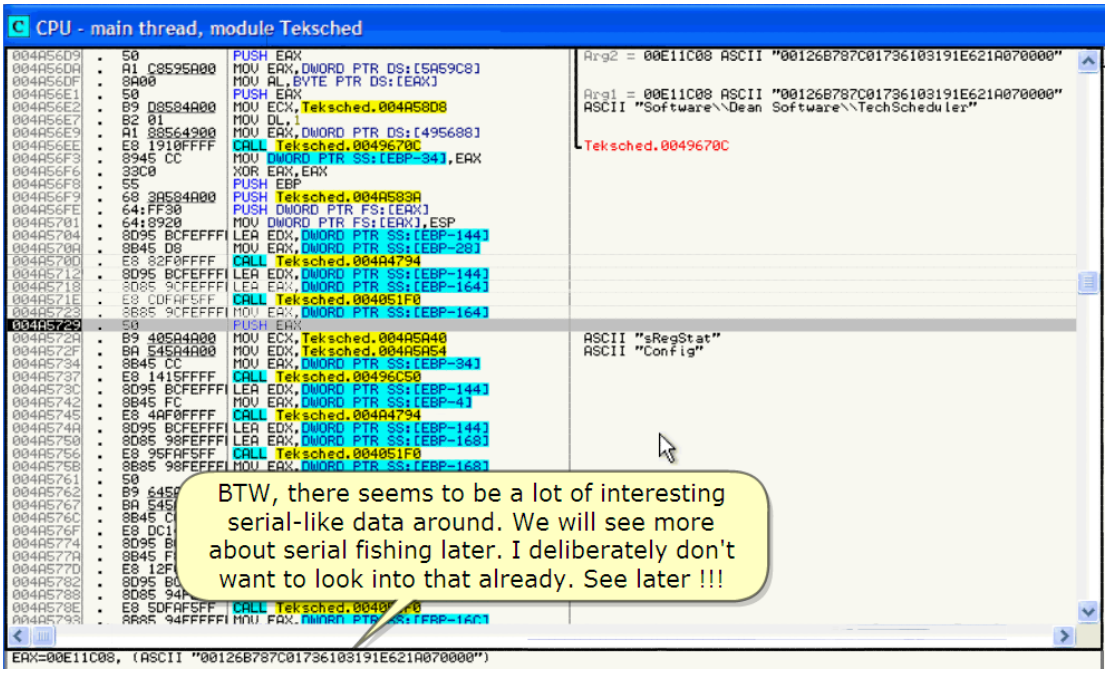
정말, 여기



;)

Place BP to remember

기억하기 위해 BP 를 설치 해.



BTW, there seems to be a lot of interesting serial-like data around.

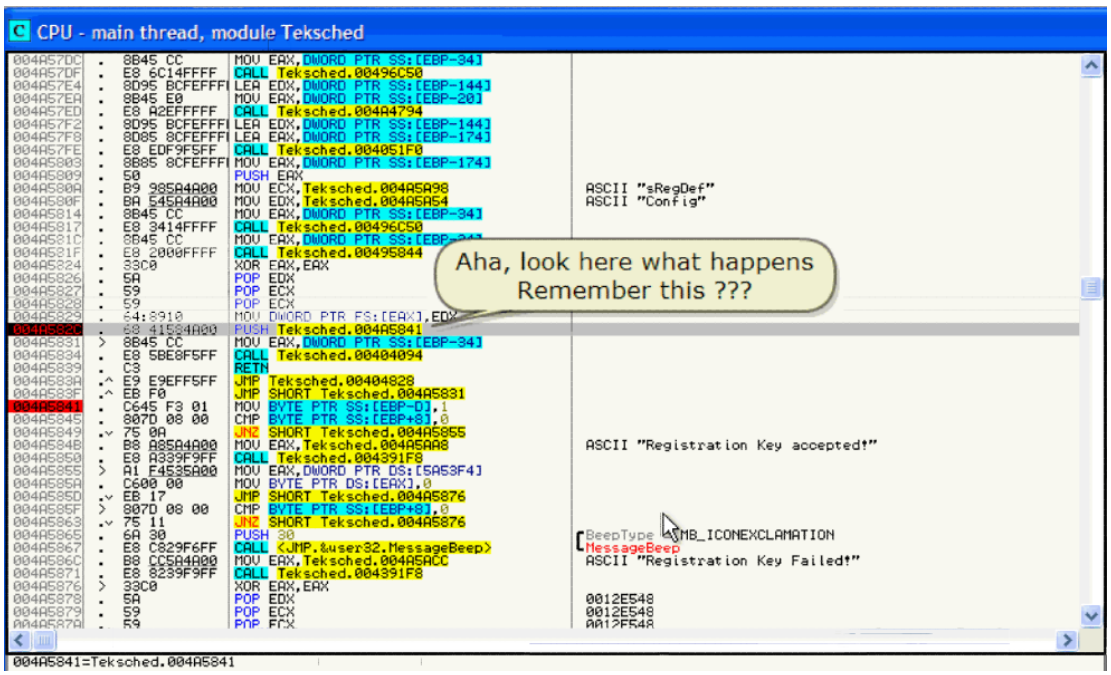
By the way, 그곳은 serial data 가 꽤 흥미롭게 있는 것이 보인다.

We will see more about serial fishing later.

우리는 serial 낚시에 대하여 나중에 좀 더 볼 수 있다.

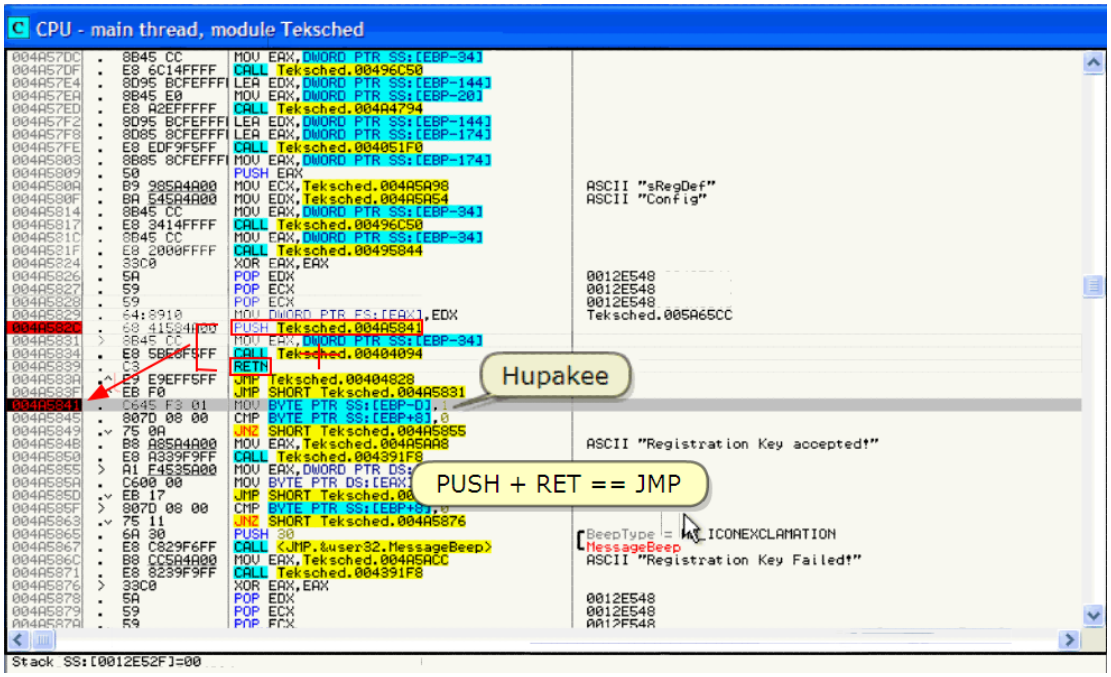
I deliberately don't want to look into that already. See later !!!

나는 고의적으로 그것을 미리 보기 원하지 않는다. 나중에 보자 !!!



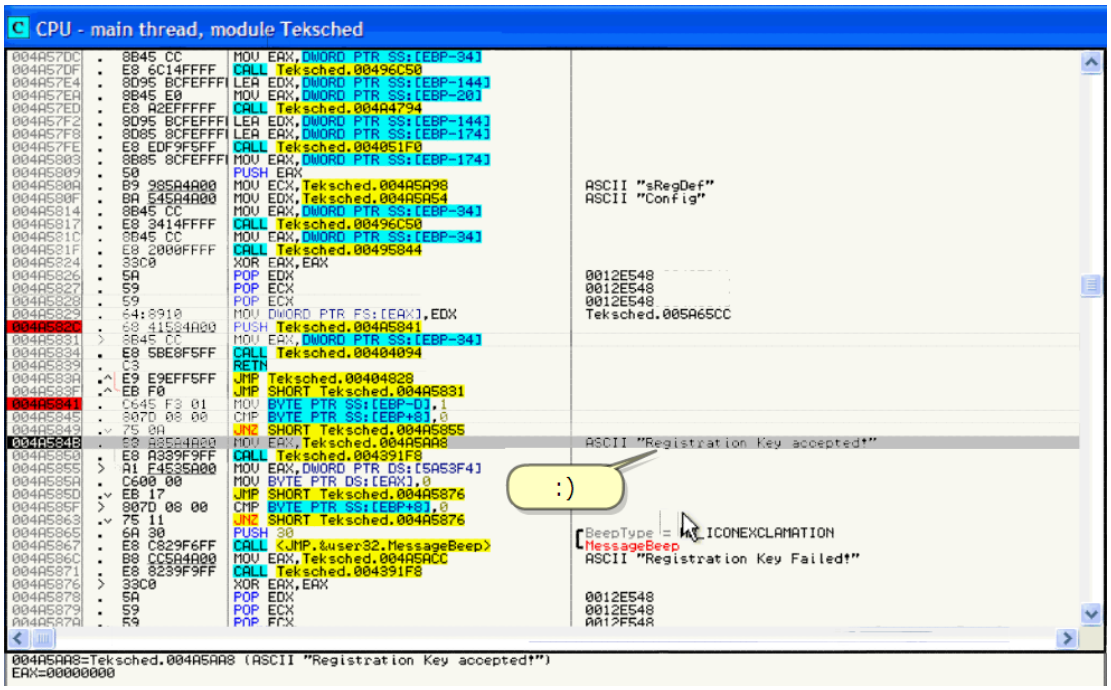
Aha, look here what happens Remember this ???

아하, 여기를 봐. 이것에 무슨 일이 발생했는지 기억해 ???

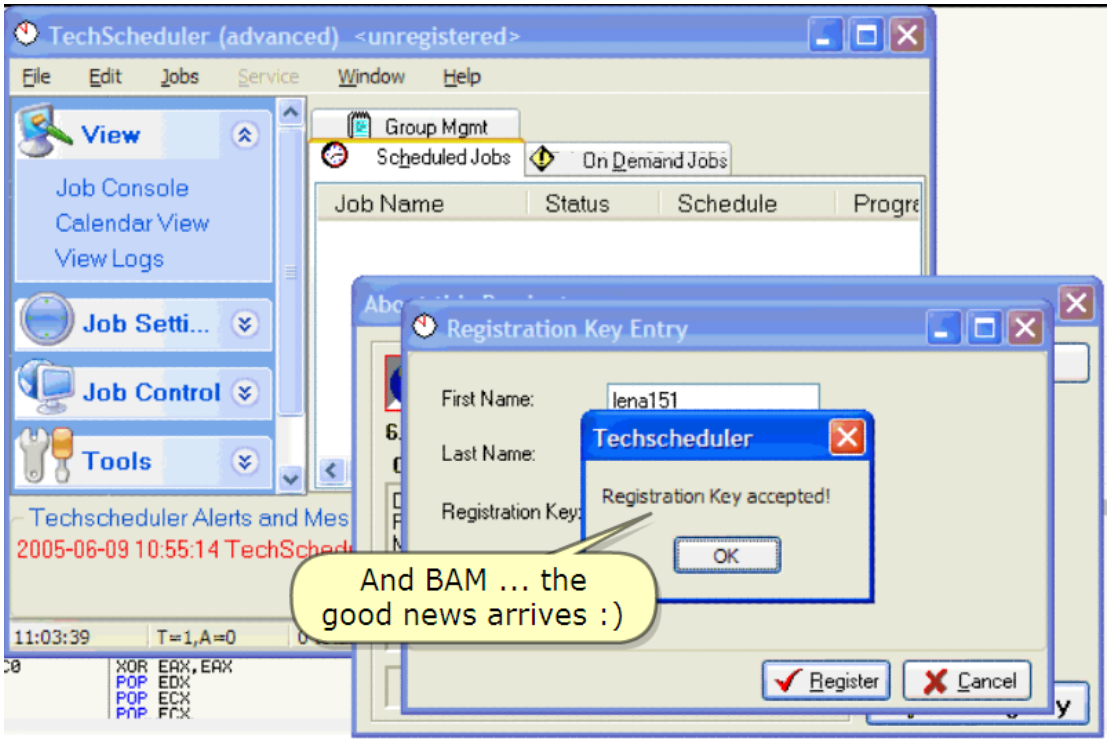


Hupakee

PUSH + RET == JMP

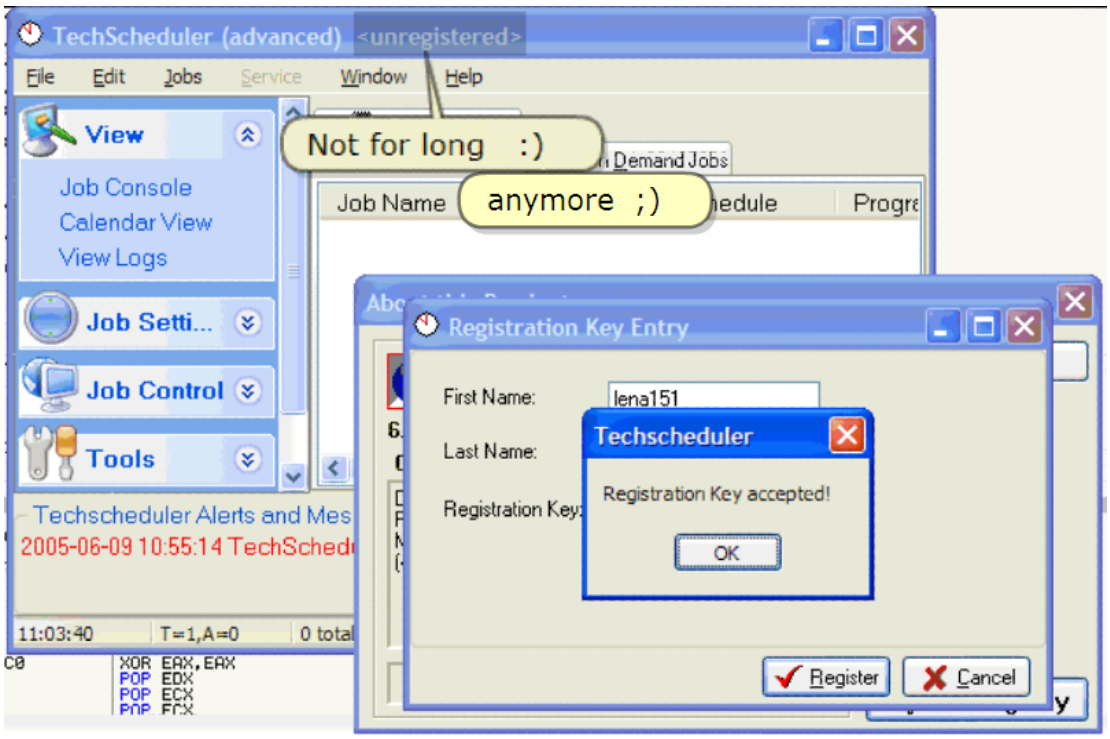


:)



And BAM ... the good news arrives :)

꽝... 좋은 소식이 도착했다 :)



Not for long :)

Anymore ;)

영원히 없다. :)

더 이상 ;)

5. Patching and testing

Scroll up

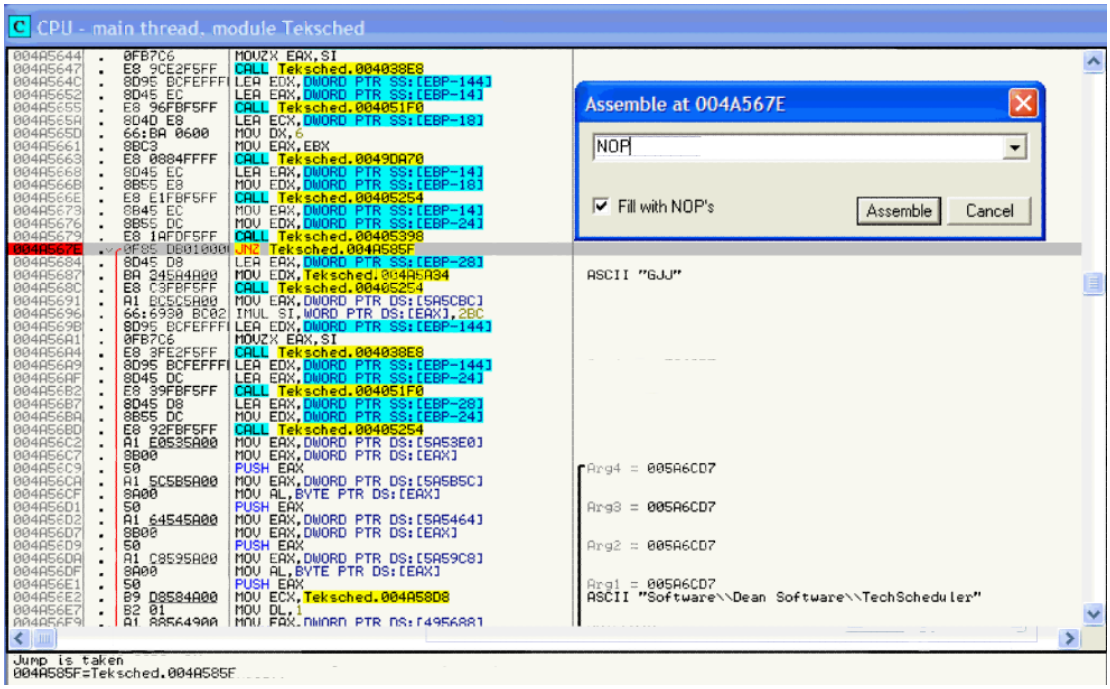
Scroll 올려.

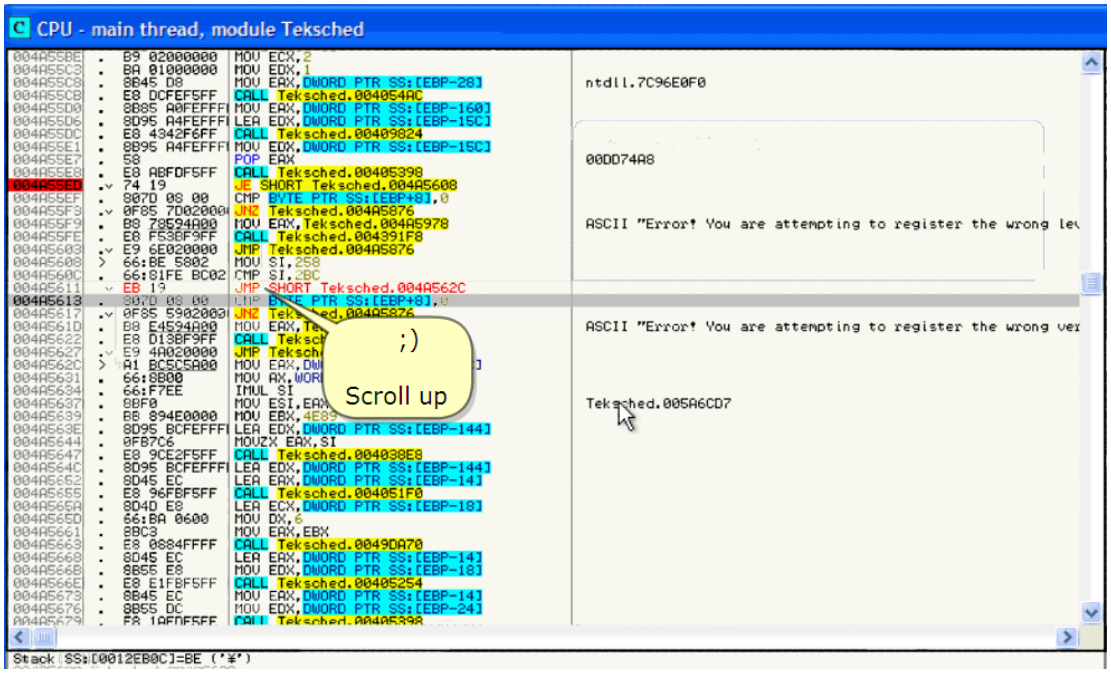
And make the necessary changes permanent

영원히 변화가 필요한 것을 만들어.

And change as follows ...

따라와. 변경하자.

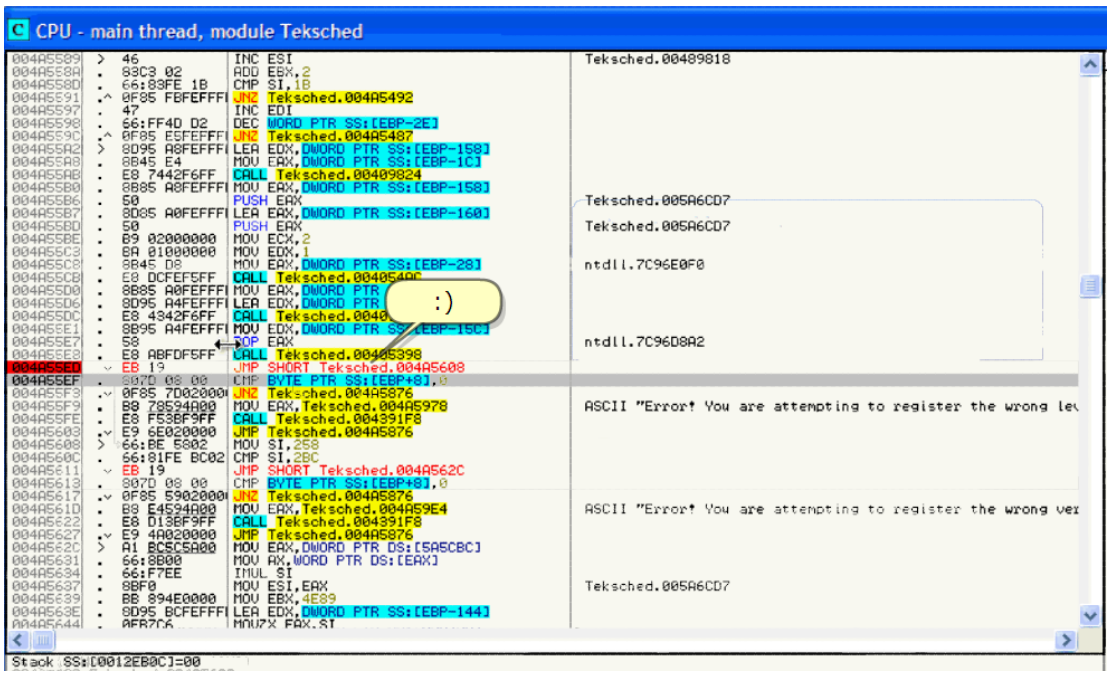




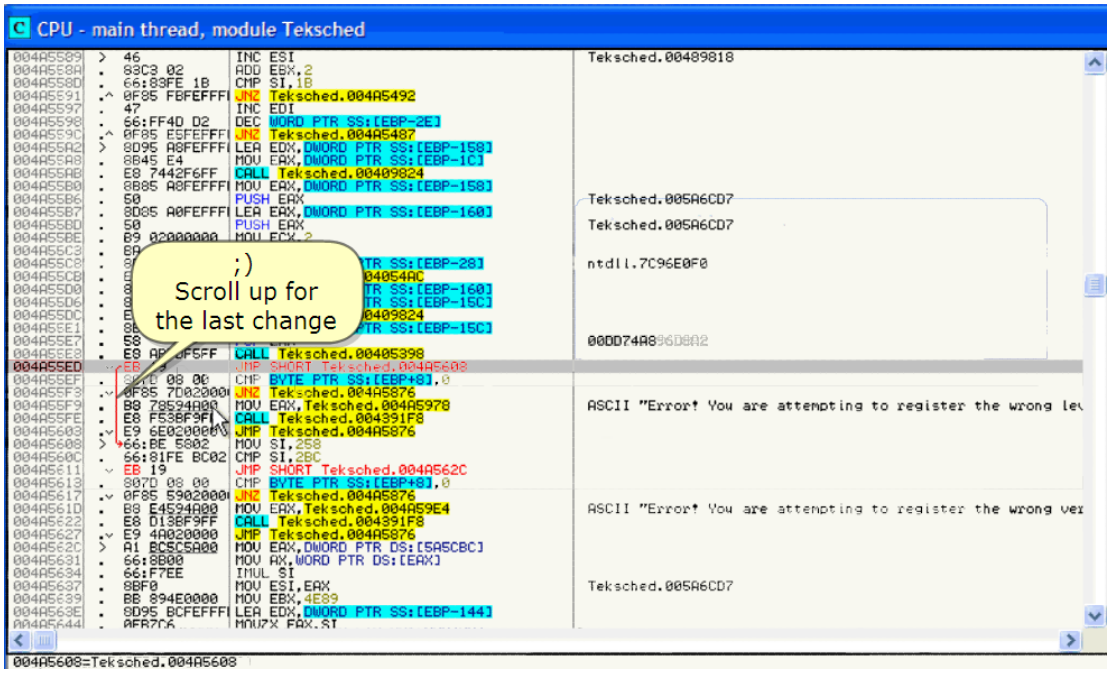
;)

Scroll up

Scroll 올려



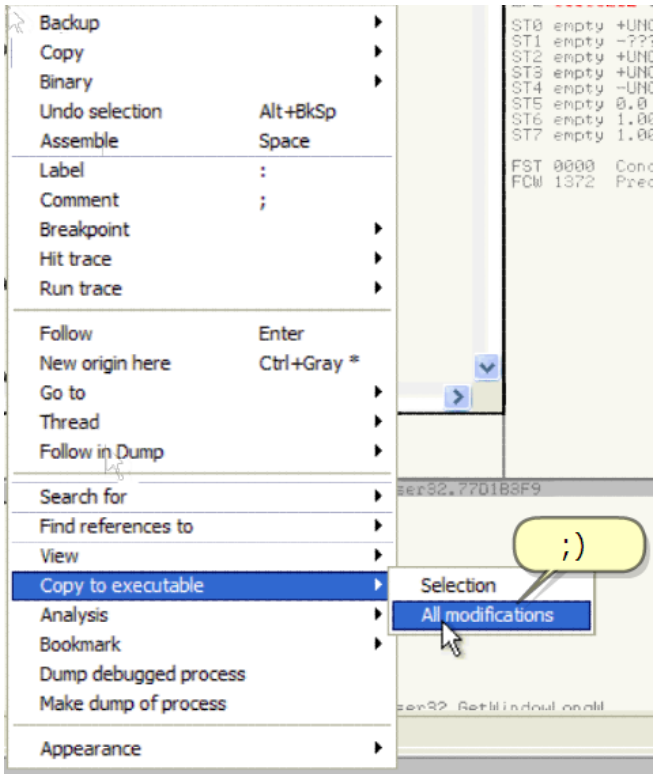
:)



;)

Scroll up for the last change

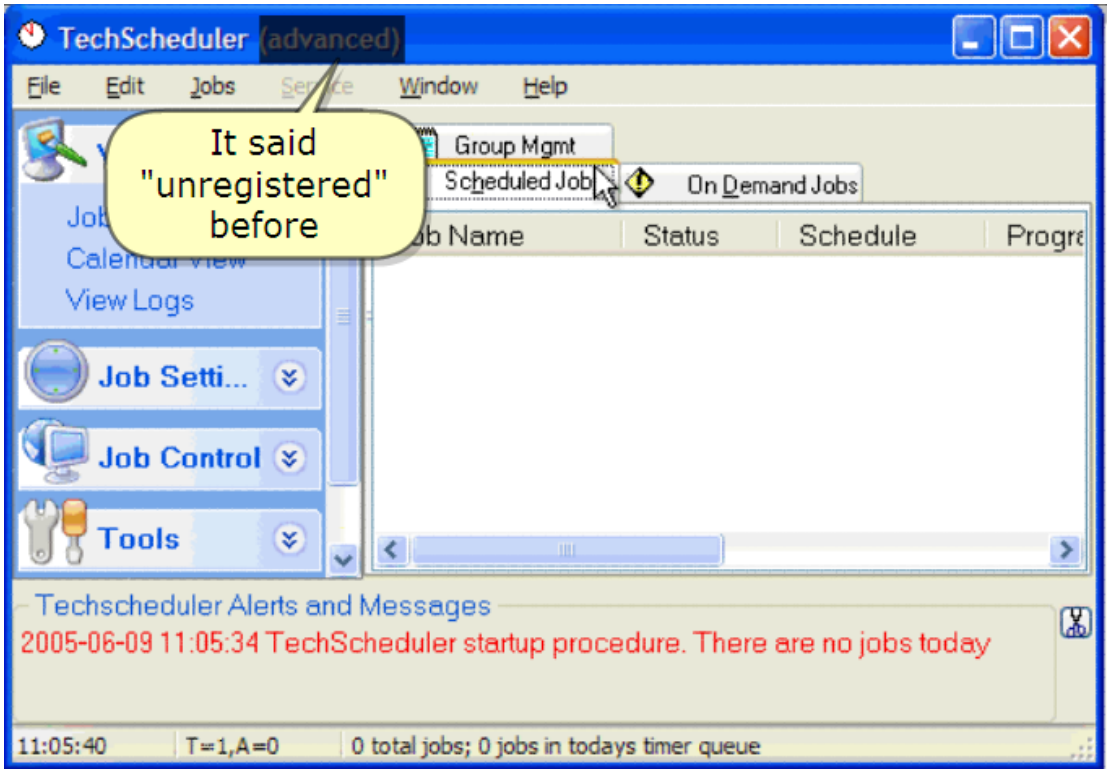
마지막 변경을 위해 Scroll 올려.



;)

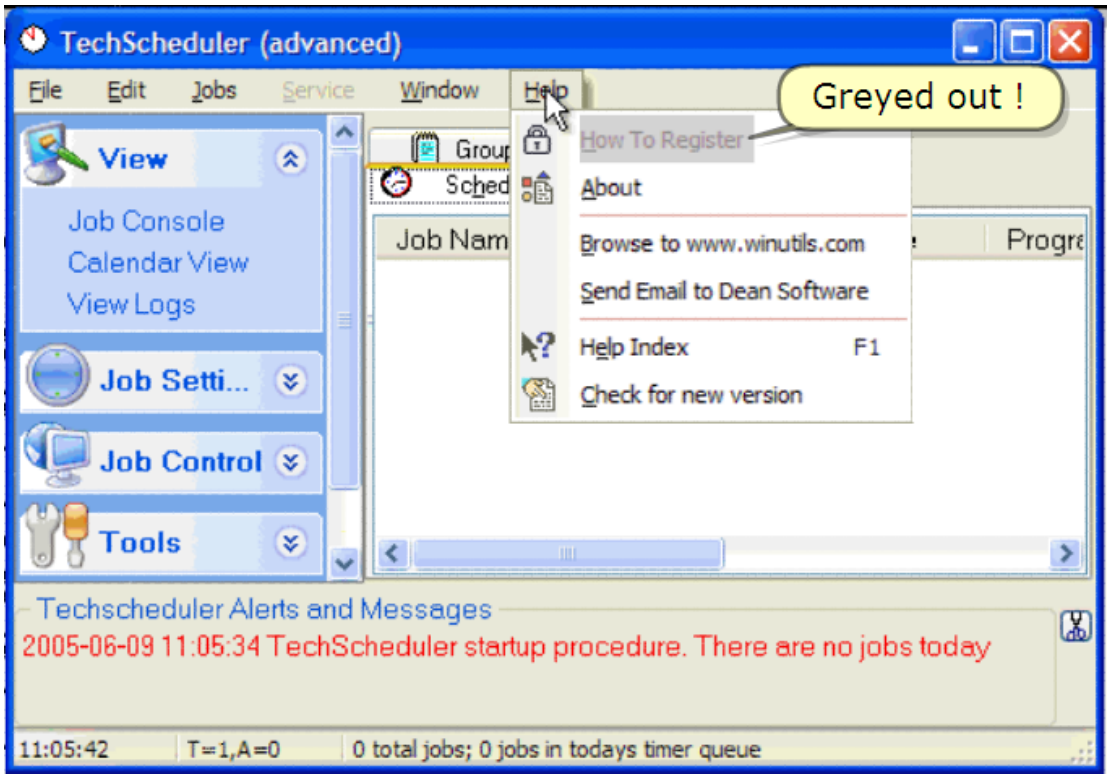
:)

;)



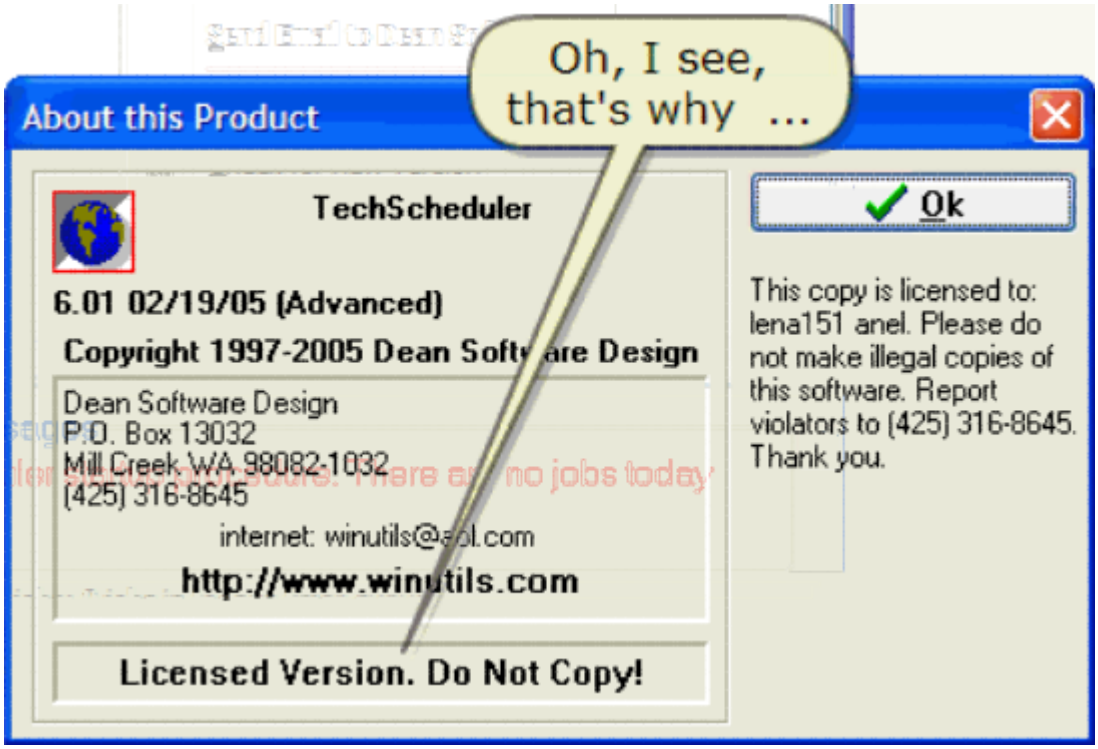
It said "unregistered" before

전에는 "unregistered" 라고 말했다.



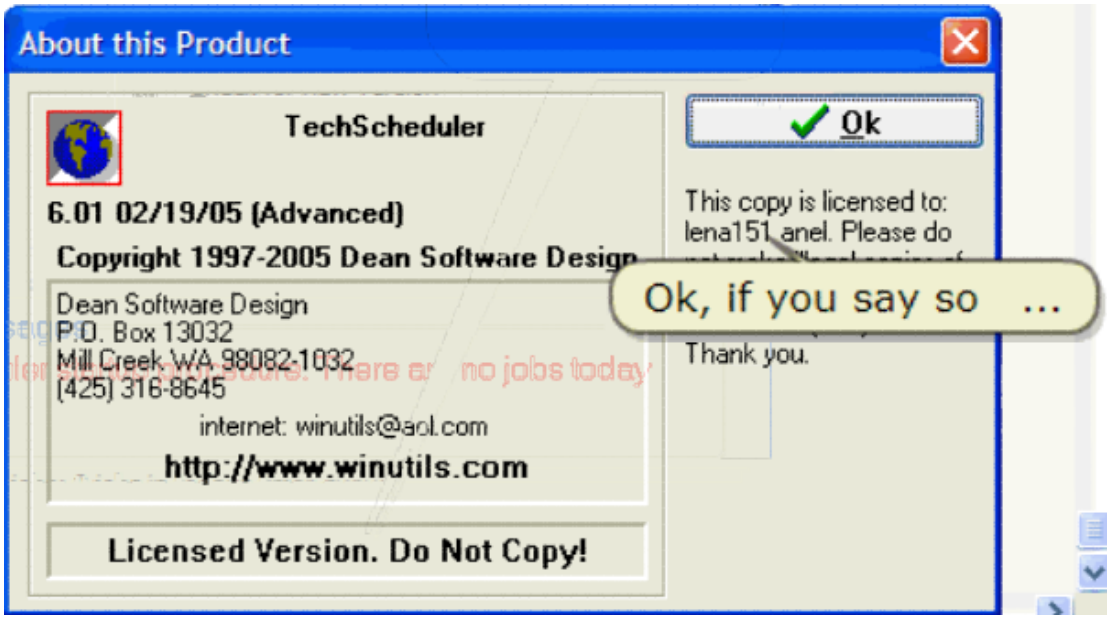
Greyed out !

회색으로 변했어 !



Oh, I see, that's why ...

오, 왜 하는지 봐.



Ok, if you say so ...

Ok, 네 id 기억하고 있어...

Ok. All seems fine.

Ok. 모두 괜찮아 보인다.

INFO :

Only guiding the program by patching the reg scheme to the goodboys doesn't often also effectively register the program.

오직 program 을 reg scheme patching 에 의해 Goodboy 로 guiding 한다. 효과적인 program 등록은 자주 필요하지 않다.
It did in this case however. Remember though that there are often also doublechecks afterwards or at a restart.
이번 경우는 해냈다. 그러나 그곳에 자주 doublecheck 가 후에 있는지 또는 restart 할 때가 있는지 기억해.
There may even be a separate registration scheme that is run at startup.
나눠진 registration scheme 가 있다. Startup 할 때 있다.
We will discuss these cases in later Parts in this series. Stay tuned.
우리는 이 series 의 나중 Parts 에서 이 경우를 의논할 수 있다. 계속 지켜봐.

6. Conclusion

In this part 12, the primary goal was to show multiple patching to guide a program into "registered".
이번 Part 12 에서, 주요한 목표는 등록하기 위해 program 을 다양한 patch 로 guide 하는 것을 보여주는 것이었다.

I hope you understood everything fine and I also hope someone somewhere learned something from this. See ma back in part 13 ;)
네가 모든 것을 좋게 이해했기를 희망한다. 또한 누구든지 어디서든지 이것에서 무언가를 배웠으면 좋겠다. Part 13 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.