06."The plain stupid patching method", searching for textstrings

2012 년 1 월 28 일 토요일
오후 7:59

Hello everybody.
모두들 안녕.
Welcome to this Part 6 in my series about reversing for newbies/beginners.
나의 초보자 reversing series Part 6 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 6, we will reverse a "real" application to learn something about basic patching. Basic patching is often referred to as "The plain stupid method".

이번 Part 6 에서, 우리는 무언가 기본적인 patch 를 배우기 위해 "real" application 을 reverse 할 것이다.
기본적인 patching 은 "분명히 멍청한 방법이다"로 자주 참고된다.
We study a real application because indeed, the best practice is found in real applications.
우리는 real application 에서 공부한다. 왜냐하면, 가장 좋은 연습을 real application 에서 찾았다.
In following Parts in this series, we will then also study "Intermediate" and "Advanced" patching.
이 series 의 이번 part 를 따라와. 우리는 "중급"과 "진화"된 patch 를 배울 것이다.
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 좋은 이해력을 위해, 나는 너에게 이 part 를 보기 전에 첫번째 part 부터 보라고 조언해 줄께.
The goal of this tutorial is to teach you something about a program's behaviour.
이 tutorial 의 목표는 너에게 program's 의 behaviour 를 가르치는 것이다.
In my search not to harm anybody, I managed to find PCsurgeon V4.20. This version is no longer available and has meanwhile been updated many times.
나는 누구에게도 해가 되지 않는 것을 찾았다. 나는 Pcsurgeon V4.20 을 찾아 관리했다. 이 version 은 더 이상 사용할 수 없고 많은 시간 동안 update 가 없었다.
However, because it could be misused, I only included the main executable (not the install exe !) for your research in the shown techniques.
그러나, 그것은 잘못된 정보일 수 있다. 나는 오직 기술만 보여주는 범위 내에서 main executable 만 너의 연구를 위해 포함했다.(install exe 는 포함하지 않았다)
This makes the program useless for all other purposes than studying material.
Program 은 공부보다 다른 목적으로는 쓸모 없게 만들었다.
Taking a look in the specialized media, I also found this application to be "cracked" numerous times before.
전문화된 media 에 관심을 가져라. 또한 나는 이 application 의 "cracked" 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 선발됐다. 왜냐하면 이것은 reversing tutorial 에 이상적이고 오직 교육적인 목적인 target 이다.
I hope you will exploit your newly acquired knowledge in a positive way.
네가 새로 얻은 지식을 긍정적인 방향으로 이용하기를 바란다.
In this matter, I also want to refer to Part 1.
이 문제는, part 1 을 참고하기를 바란다.

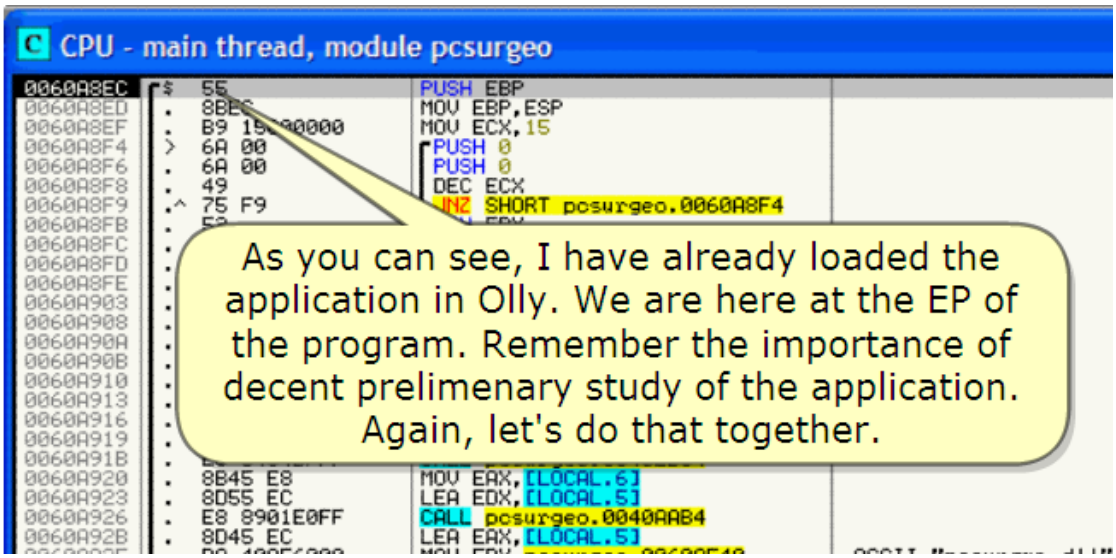
2. Tools and target

이것도 똑같음

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

Again, the brain is your responsibility ;)
다시 말하지만, 너의 뇌는 너의 책임감이다.
Todays target is the program called PCsurgeon v4.20. I have included the main executable in this package for your research.
오늘 target program 은 Pcsurgeon v4.20 이다. 나는 이 package 에 너의 연구를 위해 main executable 을 포함했다.

3. Behaviour of the program



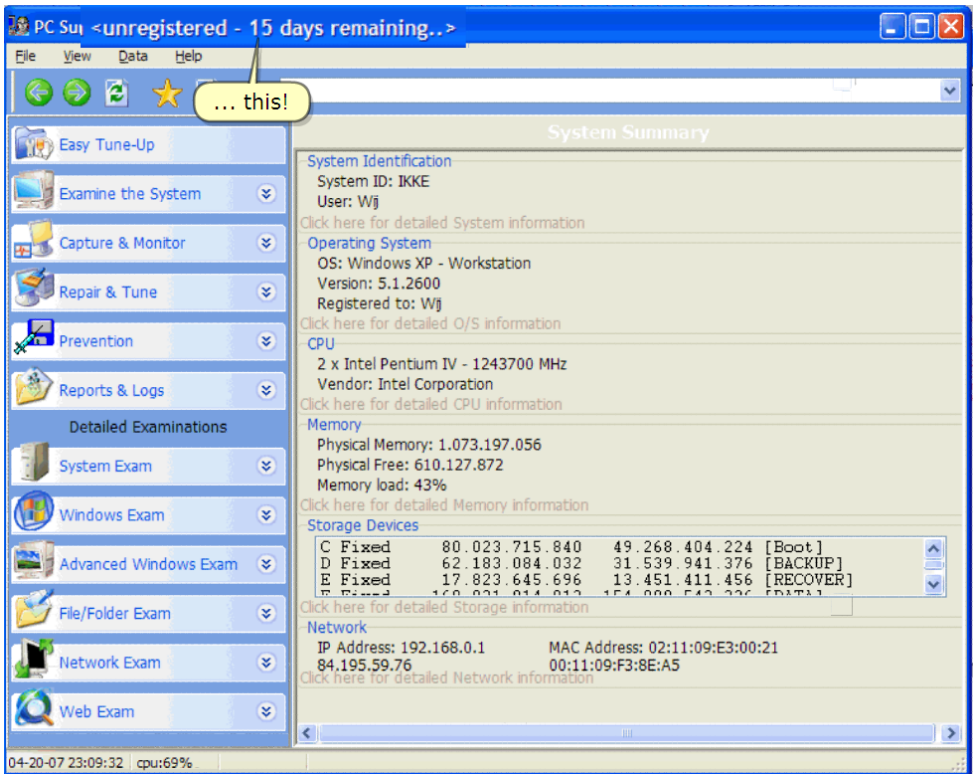
As you can see, I have already loaded the application in Olly. We are here at the EP of the program. Remember the importance of decent preliminary study of the application.
네가 볼 때, 나는 이미 Olly 에 application 을 load 했다. 우리는 program 의 EP 에 있다. 기억해. 중요한 괜찮은 application 에 대한 예비 공부다.
Again, let's do that together.
Run the application
다시, 같이하자.
Application 실행해.

Aha! A splash window! It is a nag window!
TIP: probably not showing up if registered!

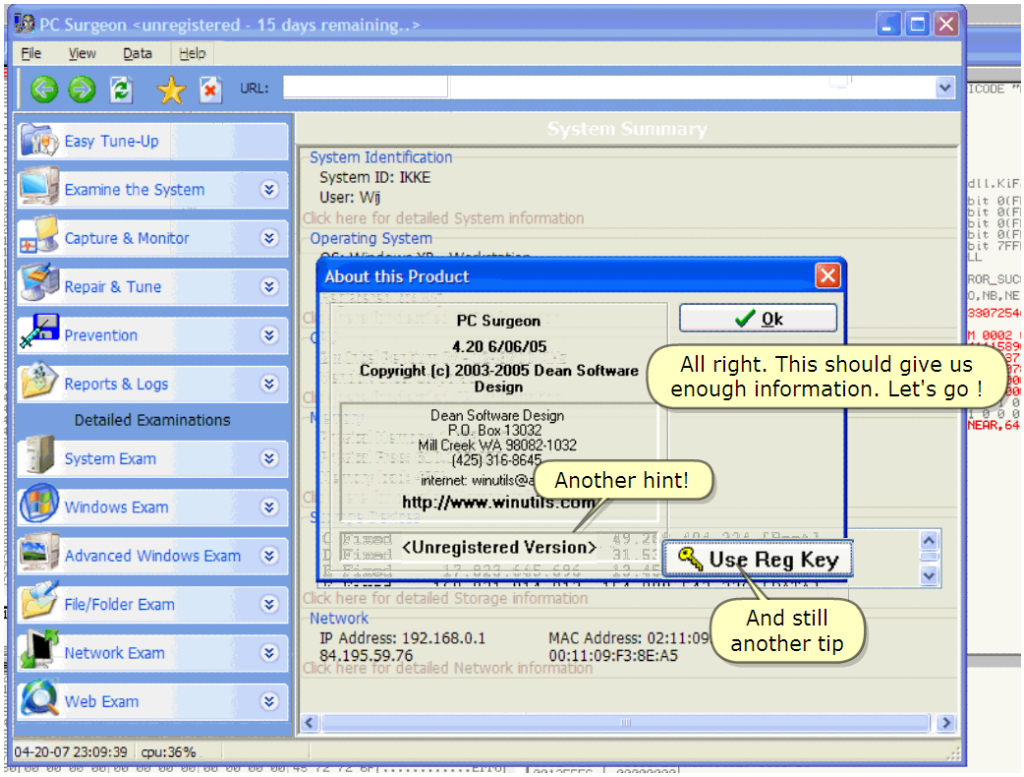
--> this forms a possible attack point ;)
아하! 장식된 window! 이것은 nag window 다.
TIP: 우리가 등록하면 보이지 않을 것이다.
--> 이 form 은 우리가 공격할 가능성이 있다.

It means that we could try to find out where is decided if this nag window is to be shown or not --> unregistered or registered !!!
이것은 뜻한다. 우리가 nag window 가 보일지 안 보일지 결정되는 곳을 찾을 수 있다. --> 미등록이거나 등록 !!!
(Or at least where is decided about setting Unregistered or Registered on the window)
최신의 장소는 window 에서 미등록이거나 등록된 setting 에 대한 것이 결정된다.

And then the nag disappears to show us ...
Nag 이 우리에게 보여주고 사라진다.



... this !
이곳
Let's find some more info
좀 더 많은 정보를 찾아보자.



Another hint

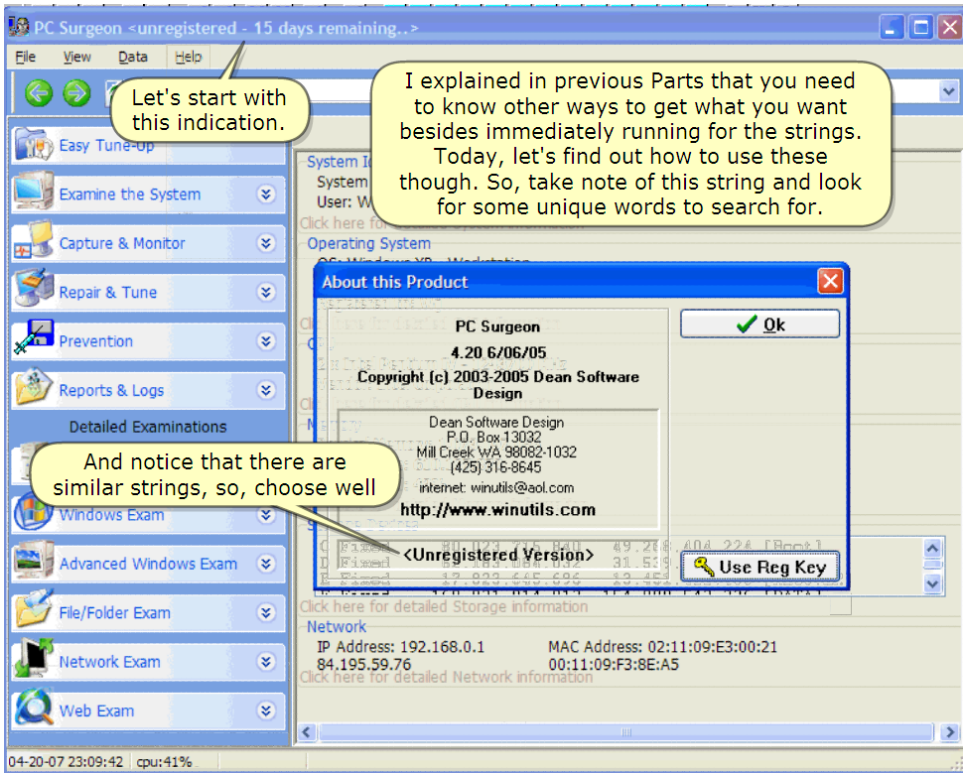
다른 hint

And still another tip

아직 다른 tip 이 있다.

All right. This should give us enough information. Let's go !

좋아. 이것은 우리에게 충분한 정보를 주었다. 시작하자!

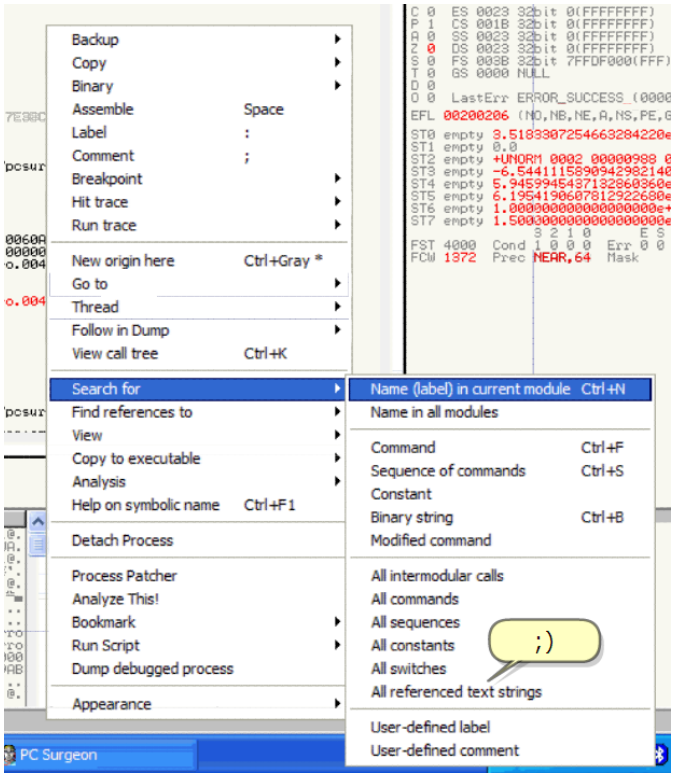


Let's start with this indication.
시작은 이 지시자와 함께 하자.

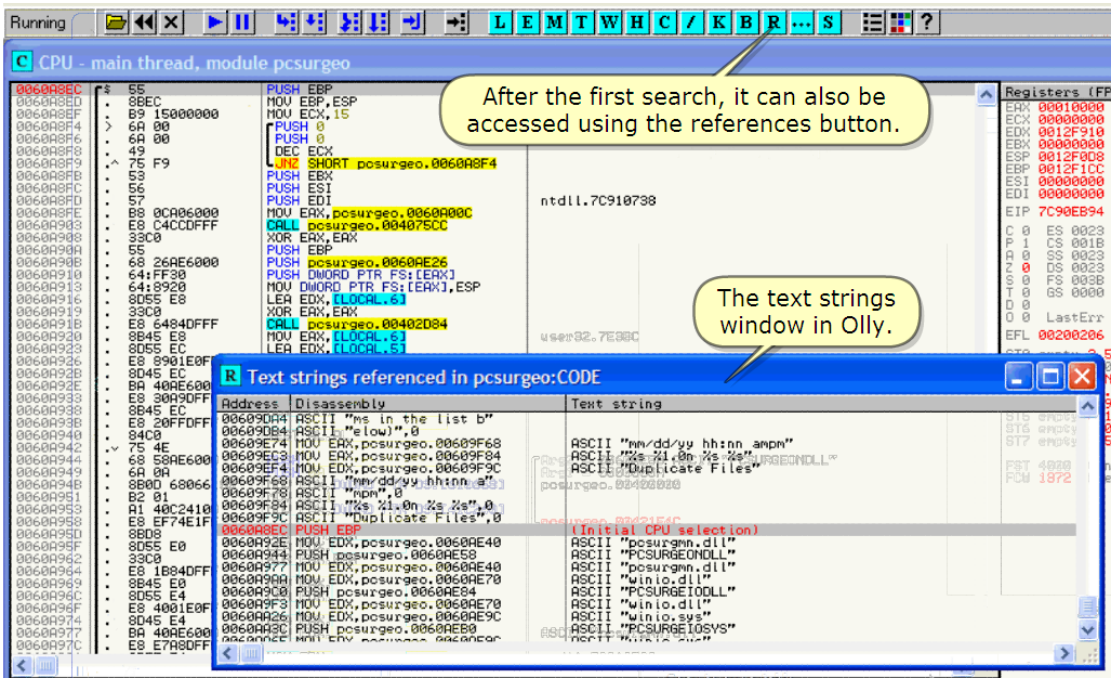
I explained in previous Parts that you need to know other ways to get what you want besides immediately running for the strings.
나는 이전의 part 에서 설명했다. 다른 방법을 알고 string 을 위해 네가 원할 때 즉시 실행하는 것이다.
Today, let's find out how to use these though. So, take note of this string and look for some unique words to search for.
오늘, 그것들이 어떻게 사용되는지 찾자. 그래서 string 을 적고 유일한 단어를 찾는데 활용하자.
And notice that there are similar strings, so, choose well
Close the "About" box and return to Olly
Return to Olly
그리고 비슷한 string 들을 적자. 그래서 잘 선택해야 된다.
"about" box 를 닫고 Olly 로 돌아가자
Olly 로 돌아가자.

Rightclick in Cpu window to bring up the pop-up menu
CPU window 에서 Rightclick 해. POP-up menu 가 뜰 거야.

4. Finding the patches



;)



The text strings window in Olly.

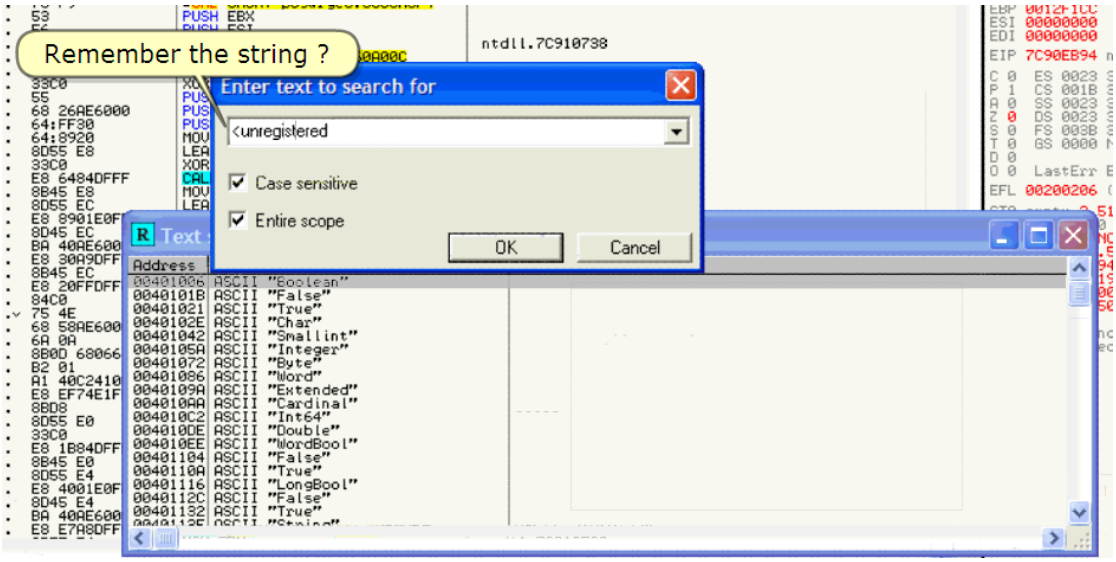
Olly 에서 Text strings window 다.

After the first search, it can also be accessed using the references button.

먼저 찾자,이것 또한 참조 button 을 사용하여 access 할 수 있다.

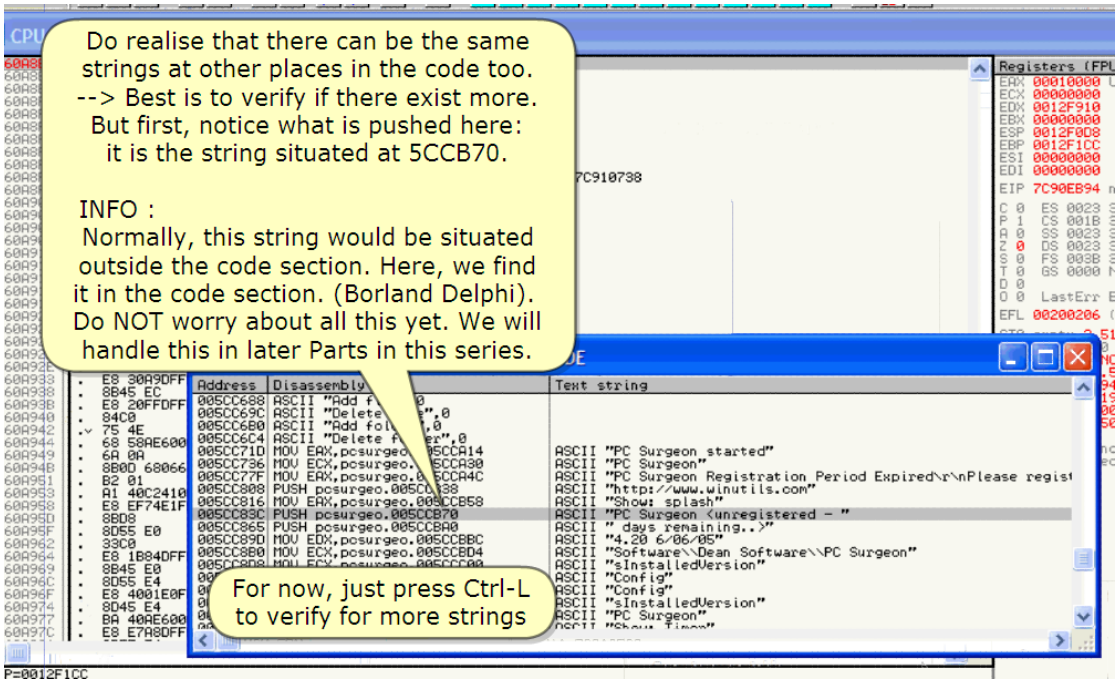
Scroll up first when searching for textstrings, (Olly searches top to bottom and we don't want to miss anything).
Strings 을 검색할 때는 먼저 Scroll 올려. (Olly 는 top 에서 bottom 으로 검색하고 우리는 어느것이든지 miss 되는 것을 원하지 않는다.

Now rightclick ...
이제 rightclick ...



Remember the string?
String 을 기억해?
Best is to search case insensitive in the entire scope so we don't miss anything.
전체 범위에서 대소문자 구별없이 찾는게 가장 좋다. 우리는 어떤 것도 실수하지 않는다.
Here is a very resembling string, so, I'm performing the search with the "Case sensitive" box checked
이곳은 자주 resembling string 된다. 그래서 나는 search 를 "대소문자 구별있이"의 box 에 check 하고 실행한다.

Found!
찾았다!



Do realize that there can be the same strings at other place in the code too.

깨달아라. Code 에서 같은 string 은 다른 곳에 위치해 있을 수도 있다.

--> Best is to verify if there exist more.

--> 가장 좋은 방법은 그곳이 존재하는지 검증하는 것이다.

But first, notice what is pushed here:

It is the strings situated at 5CCB70.

먼저, 우리는 이곳에 넣겠다.

이 string 은 5CCB70 에 위치해 있다.

INFO :

Normally, this string would be situated outside the code section. Here, we find it in the code section. (Borland Delphi).

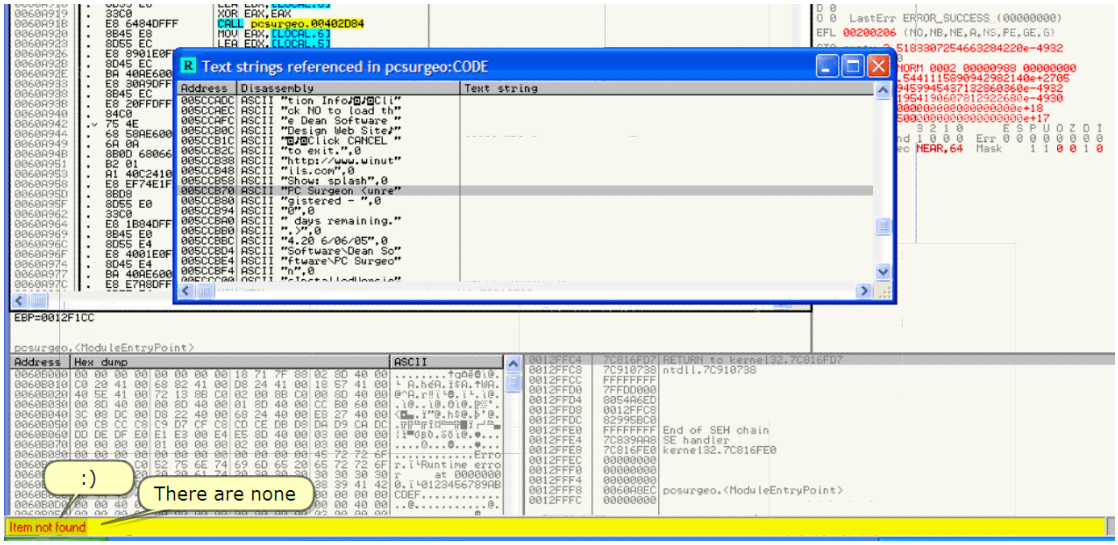
보통, string 은 code section 바깥에 위치해 있다. 여기, 우리는 code section 에서 찾았다. (Borland Delphi)

Do NOT worry about all this yet. We will handle this in later Parts in this series.

아직 걱정하지마. 우리는 이 series 의 나중 Part 에서 조종할 수 있다.

For now, just press Ctrl-L to verify for more strings

이제, Ctrl-L 을 눌러 좀 더 많은 string 을 검증하자.



Bam, we find another one?

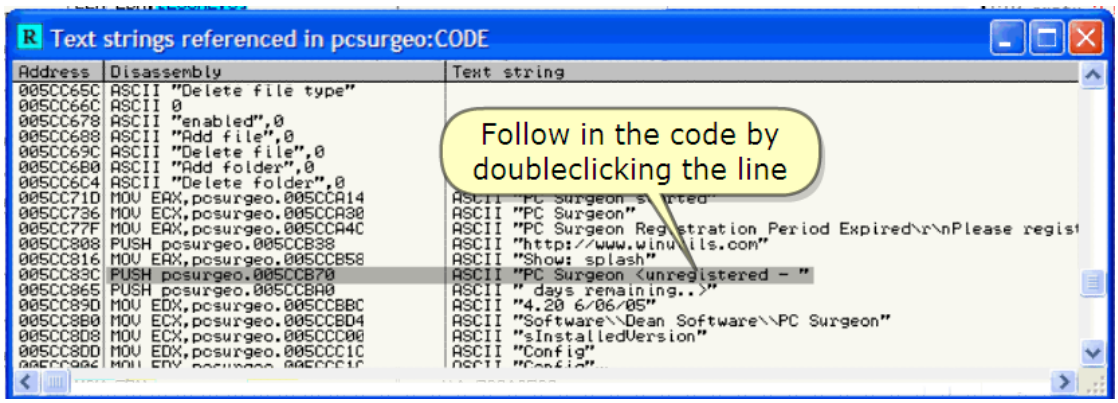
우리는 다른 곳에서 찾았다.

:) There are none

이곳은 없다.

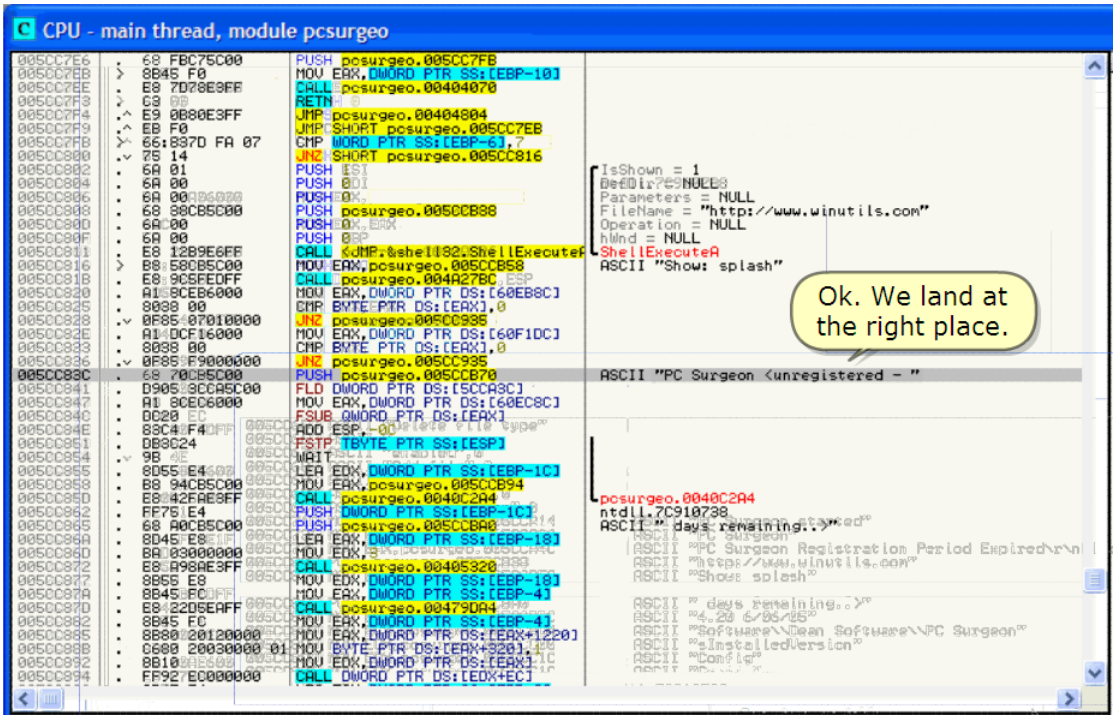
There is only one instance of the searched text string. Let's go after it and scroll up again

그곳은 오직 찾게 된 text string 사례이다. 그걸 하고 다시 scroll up 해.



Follow in the code by doubleclicking the line

Code 에서 이 line 을 doubleclick 하여 따라가자.



Ok. We land at the right place.

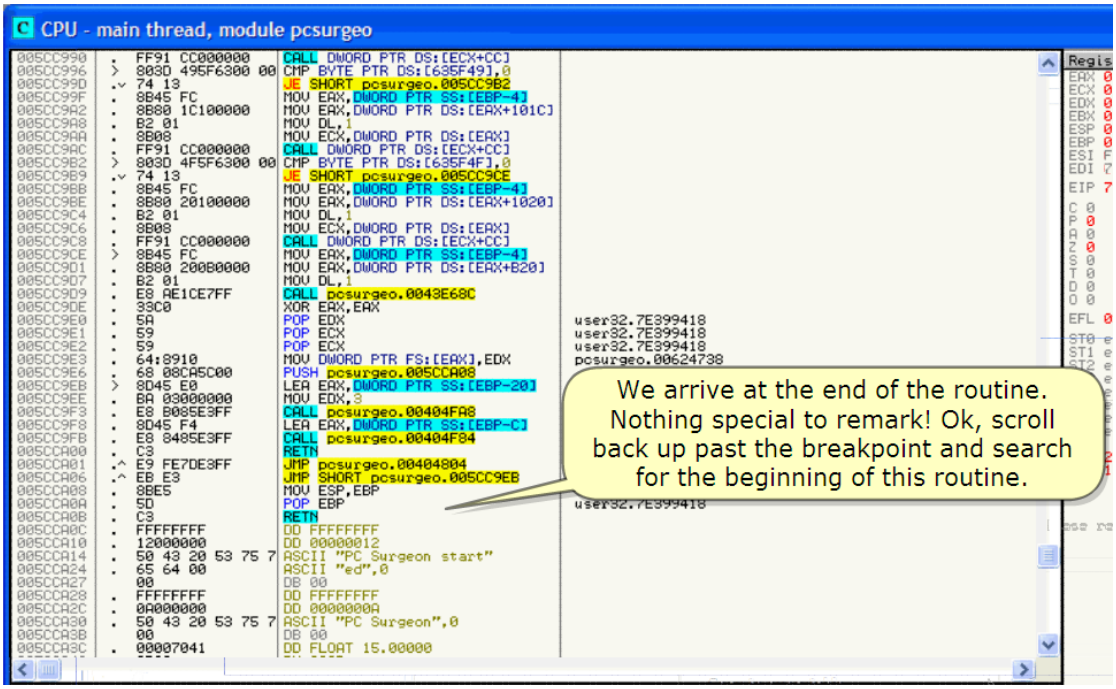
Ok. 우리는 정확한 장소에 도착했다.

Studying this better, we immediately notice the conditional jumps

공부하기 좋게, 우리는 즉시 조건 jump 를 있다.

Let's take an overview first. Scroll down this routine and keep the eyes open (which means to search the API names for important things and looking at recognizable strings)

먼저 관점을 갖자. 이 routine 에서 Scroll 내려봐. 그리고 눈을 뜨고 잘 봐.(이 뜻은 API name 을 중요한 정보와 잘 구성된 string 으로 찾는다.)

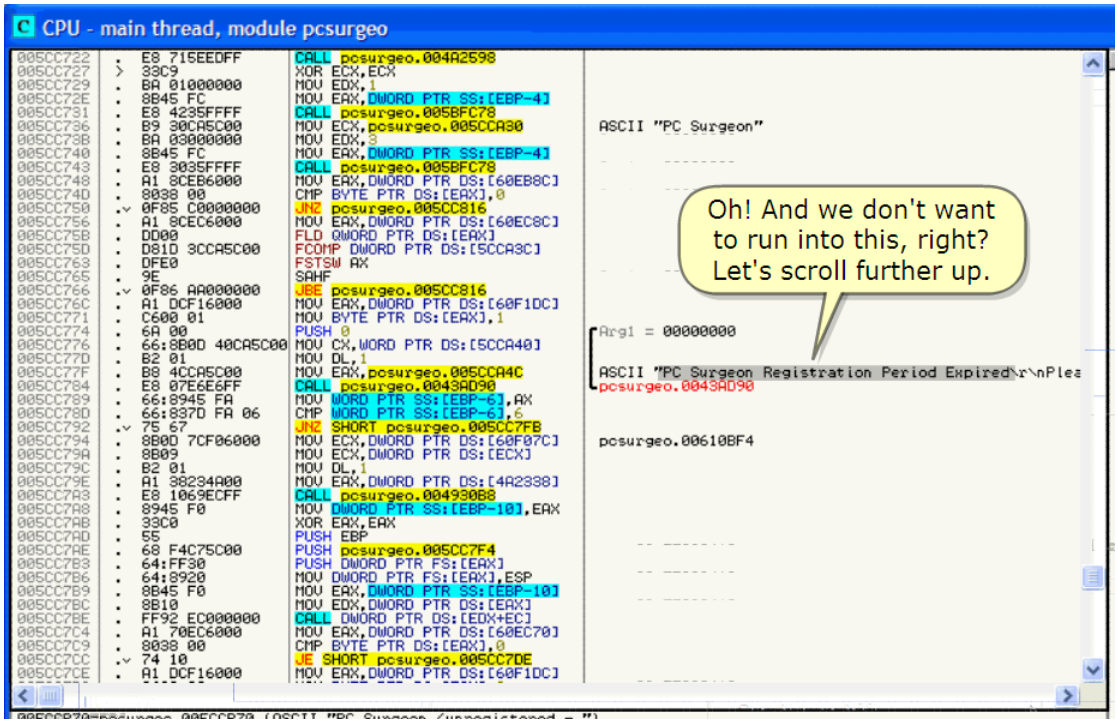


We arrive at the end of the routine.

우리는 routine 의 끝에 도착했다.

Nothing special to remark! Ok, scroll back up past the breakpoint and search for the beginning of this routine.

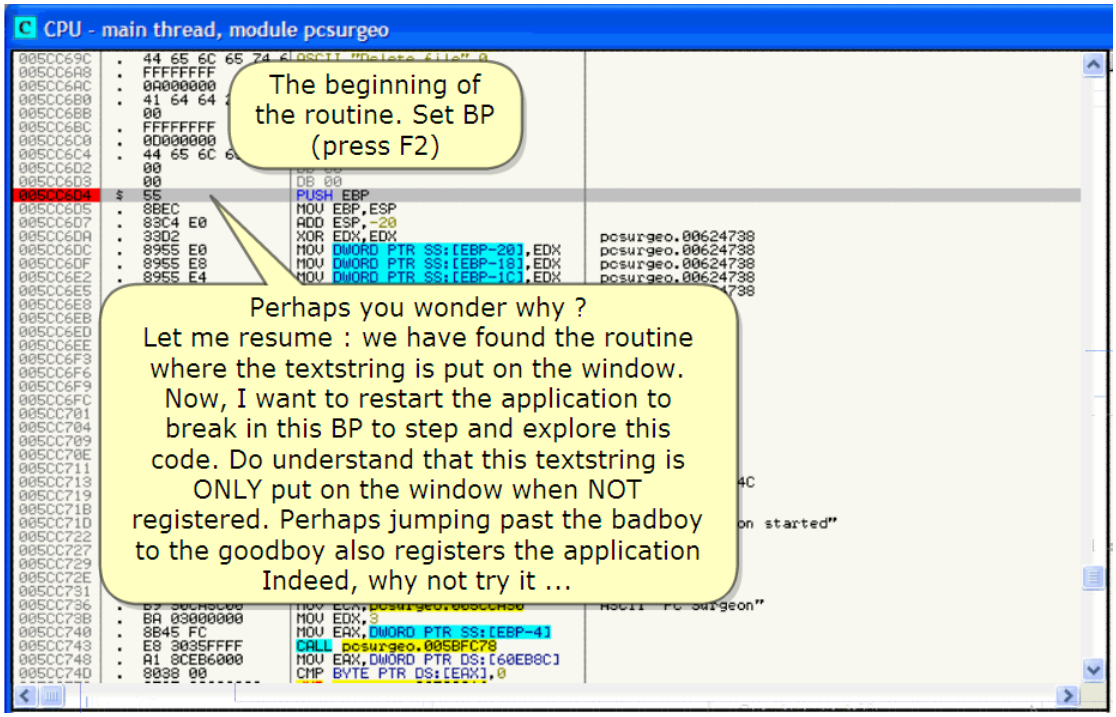
특별히 주목할 것은 없다. Scroll 다시 breakpoint 를 지나 원래대로 올려. 그리고 routine 의 시작을 찾아.



Oh, and we don't want to run into this, right?

Let's scroll further up.

오, 그리고 우리는 이곳으로 실행하기를 원하지 않는다. 그렇지?
Scroll 좀 더 올려.



The beginning of the routine. Set BP(press F2)
Routine 의 시작이야. BP 설치해(F2 눌러)
Perhaps you wonder why?
아마 너는 왜 하느냐고 생각하겠지?
Let me resume : we have found the routine where the textstring is put on the window.
요약하자면 : 우리는 textstring 이 window 에 들어가 있는 routine 을 찾았다.
Now, I want to restart the application to break in this BP to step and explore this code.
이제, 우리는 application 을 restart 하고 BP 에서 멈출 것이다. 그리고 code 를 탐험하자.
Do understand that this textstring is ONLY put on the window when NOT registered.
등록되지 않았을 때 Textstring 이 오직 window 에 들어갔다는 것을 이해 했을 거야.
Perhaps jumping past the badboy go to the goodboy also registers the application Indeed, why not try it...
Application 이 등록되었다면, 아마 jumping 은 과거의 badboy 를 지나쳐서 goodboy 로 갈 거야. 왜 우리는 도전하지 않지?

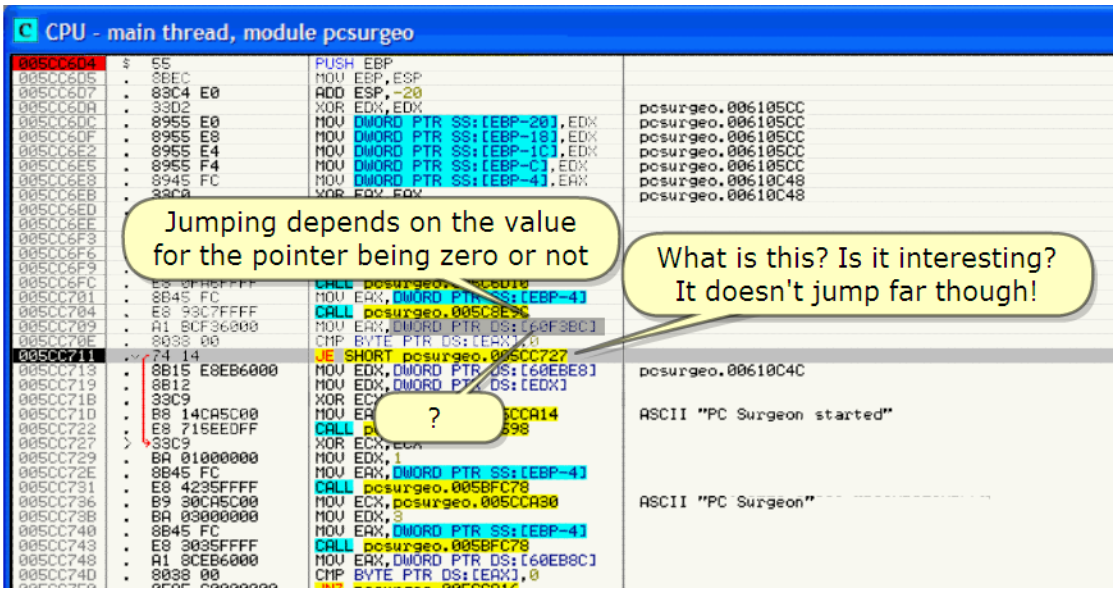
Restart
재시작
Run
실행

Fine. Olly breaks in the BP.

좋아. Olly 가 BP 안에서 멈췄다.

Start exploring and trace F8

탐험을 하자. F8 로 추적해.



What is this? Is it interesting?

뭐야? 재미있어?

It doesn't jump far though!

이것은 멀리 jump 하지 않는다.

?

Jumping depends on the value for the pointer being zero or not

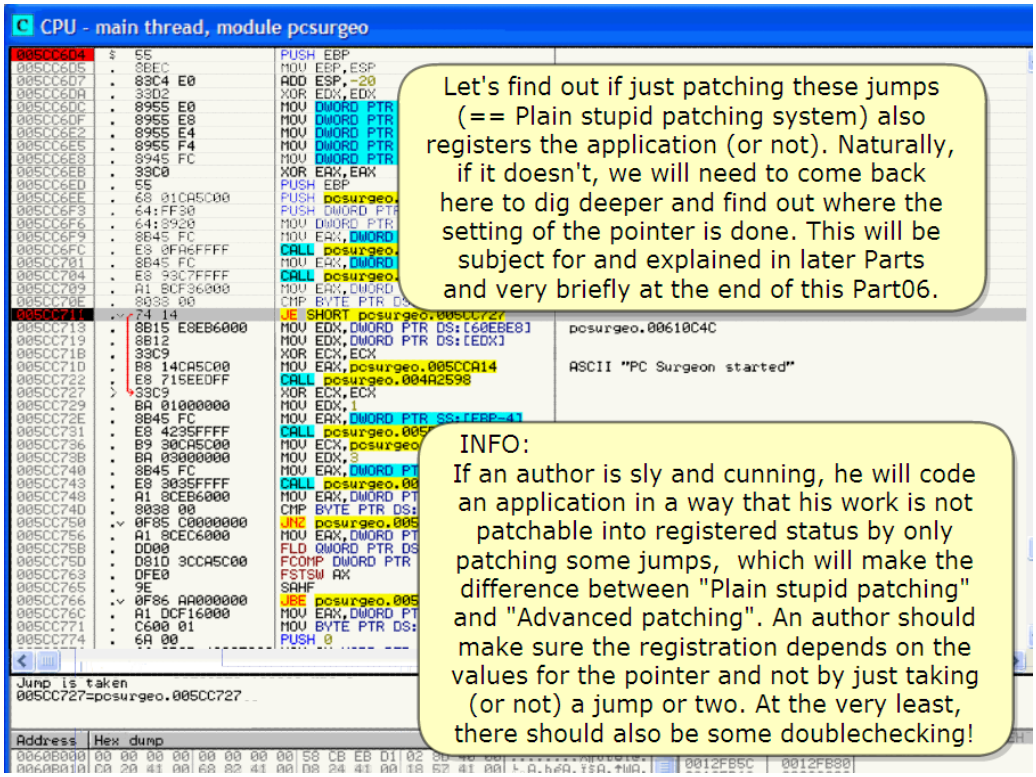
Jumping 은 pointer 의 값이 zero 인지 아닌지 의존한다.

For now, let's just continue to see what more is interesting or happens

이제, 무엇이 점점 더 흥미롭거나 어떤 일이 일어나는지 보기 위해 계속하자.

But set a BP here to remember for later that [60F3BC] may be important and require deeper investigation

그러나 나중에 기억하기 위해 BP 를 이곳에 set 할 수 있다. [60F3BC]는 매우 중요하다. 그리고 깊이 있는 조사가 요구된다.



INFO:

If an author is sly and cunning, he will code an application in a way that his work is not patchable into registered status by only patching some jumps, which will make the difference between "Plain stupid patching" and "Advanced patching".

만약에 제작자가 교활하다면, 그는 application 을 암호화 했을 것이다. 약간의 jump 에 의해 그것을 등록된 상태로 patch 할 수 없게 했을 것이다. 이것은 Plain stupid patching"과 "Advanced patching"은 다르다.

An author should make sure the registration depends on the values for the pointer and not by just taking (or not) a jump or two.

제작자는 물론 Pointer 를 위해 등록에 의존할 수 있다. 그리고 jump 에 의해 가질 수 없다.

At the very least, there should also be some doublechecking!

매우 적게, 그곳은 약간의 doublecheck 를 하는 게 있다.

Let's find out if just patching these jumps (== Plain stupid patching system) also registers the application (or not).

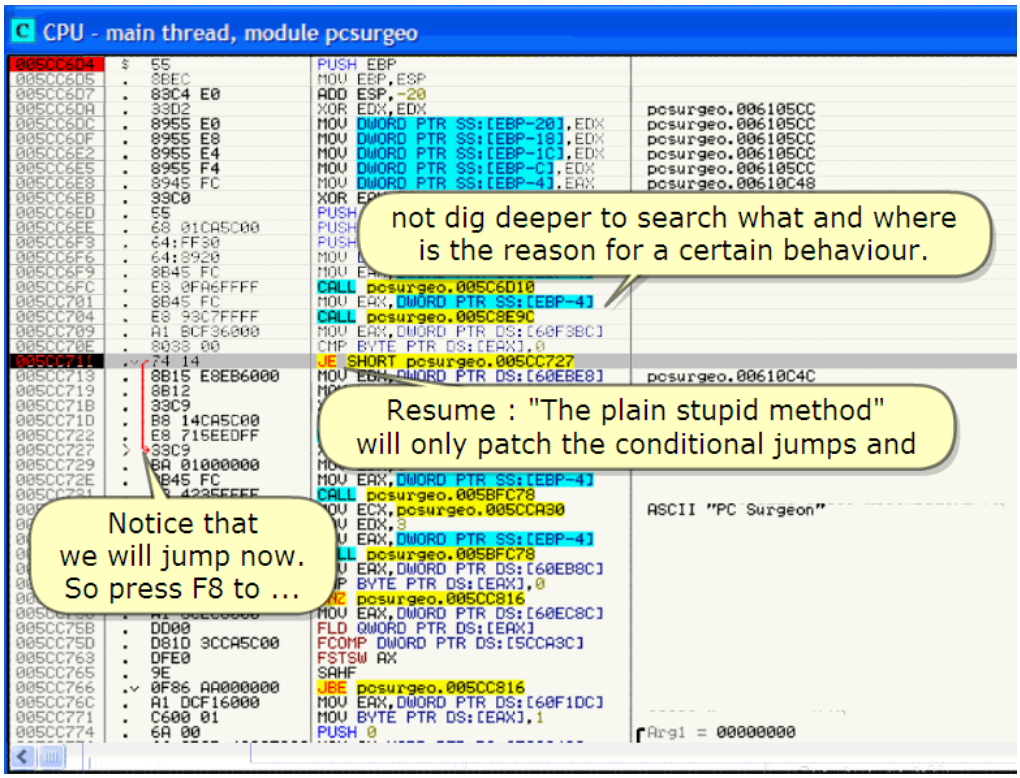
Patching 된 jump 를 찾아라. (== 명백히 멍청한 patching system) 또한 application 을 등록하자(하지 않거나)

Naturally, if it doesn't, we will need to come back here to dig deeper and find out where the setting of the pointer is done.

자연적으로, 그것을 하지 않는다면, 우리는 좀 더 깊게 파기 위해 여기로 돌아오는 게 필요하다. 그리고 pointer setting 이 된 곳을 찾아라.

This will be subject for and explained in later Parts and very briefly at the end of this Part06.

종속될 수 있다. 그리고 나중 Part 에서 설명된다. 그리고 매우 간단히 Part 6 에서 언급될 것이다.



Resume : "The plain stupid method" will only patch the conditional jumps and not dig deeper to search what and where is the reason for a certain behaviour.

요약 : "The plain stupid method" 는 오직 조건 jump 를 patch 한다. 그리고 정확한 행동을 위해 무엇과 어디에 있는지 깊게 들어가서 찾을 수 없다.

Notice that we will jump now.

우리는 이제 jump 할 것이다.

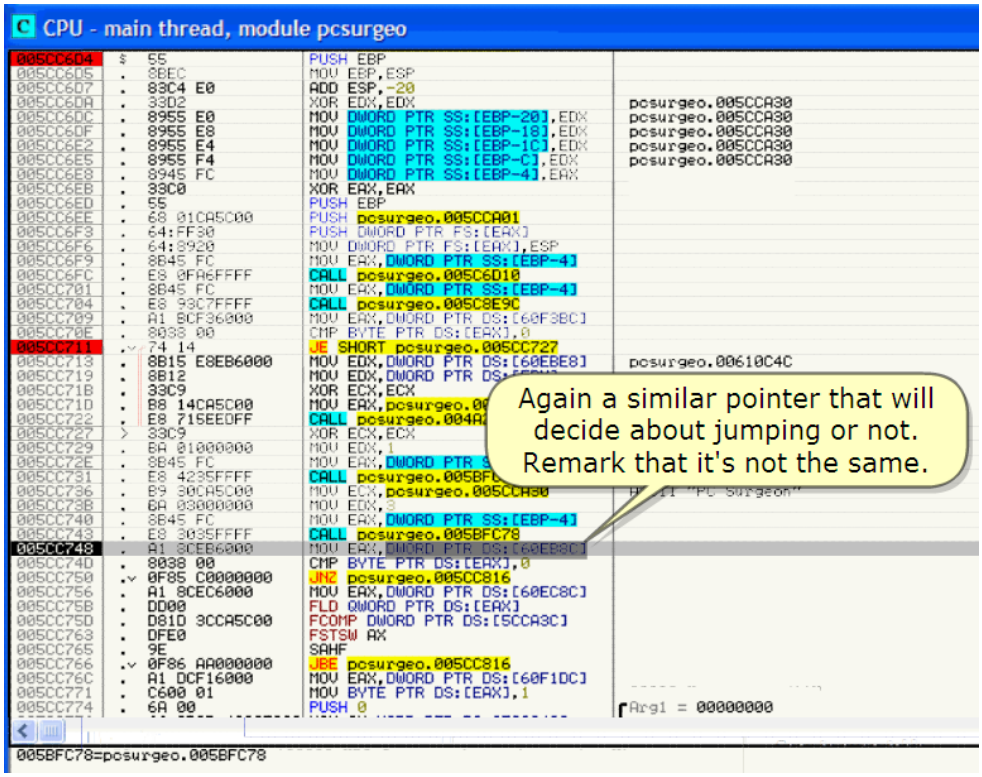
So press F8 to ...

... land here.

그래서 F8 을 눌러서 이곳에 도착해

Continue exploring

탐험을 계속하자.

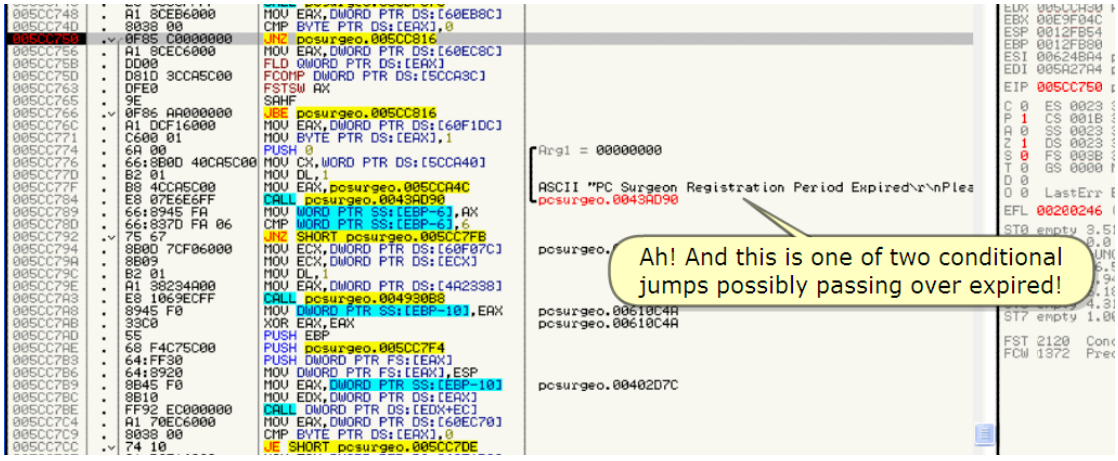


Again a similar pointer that will decide about jumping or not.

Remark that it's not the same.

다시 비슷한 pointer 다. 이것은 jump 할지 안 할지 결정할 것이다.

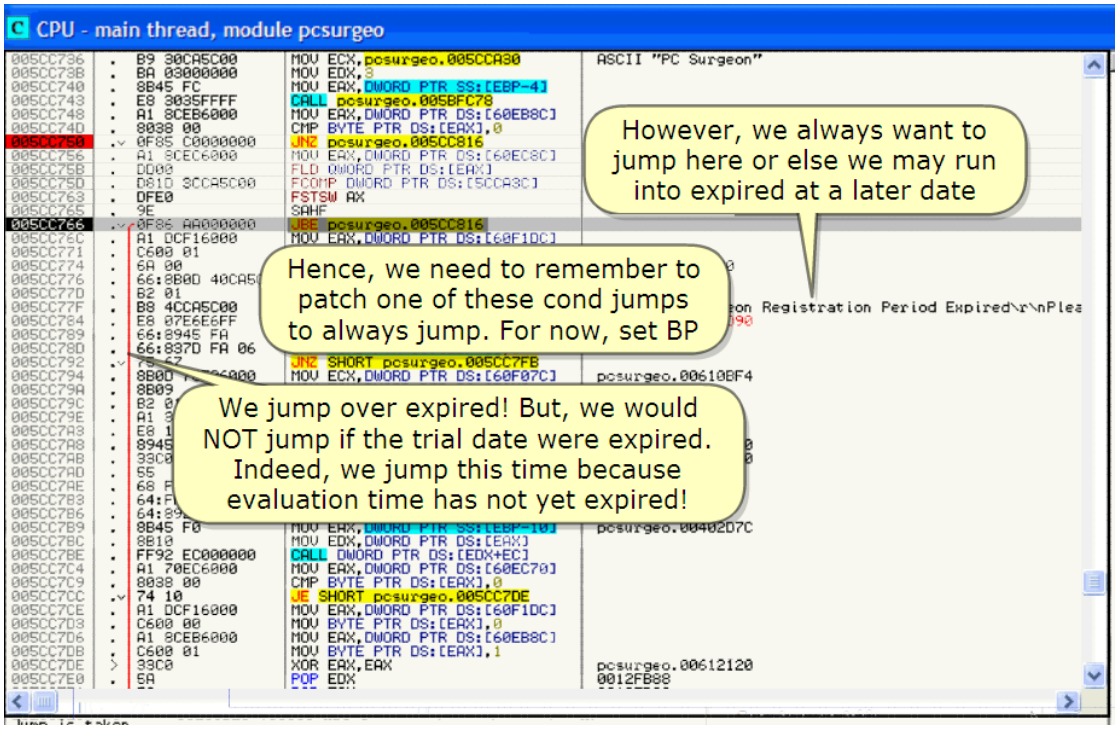
주목. 이것은 같지 않다.



Ah! And this is one of two conditional jumps possibly passing over expired!

Step F8

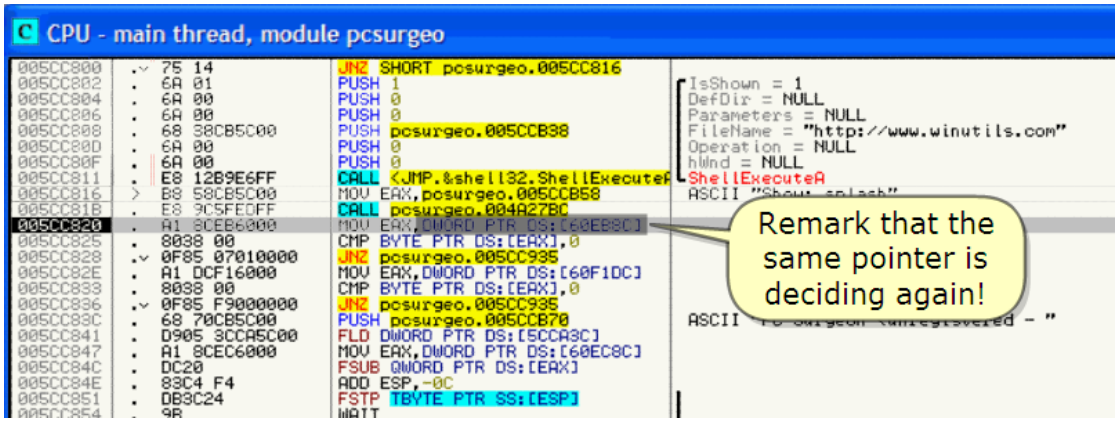
아! 2 가지 조건 jump 가 expire 되는 곳을 jump 할 수 있다.



We jump over expired! But, we would NOT jump if the trial date were expired.
Indeed, we jump this time because evaluation time has not yet expired!
우리는 expired 를 jump 한다. 그러나, 우리는 trial date 가 만료되었을 때는 jump 할 수 없다.
정말, 우리는 이 시간에 jump 해야 한다. 왜냐하면 평가하는 시간이 아직 종료되지 않았기 때문이다.

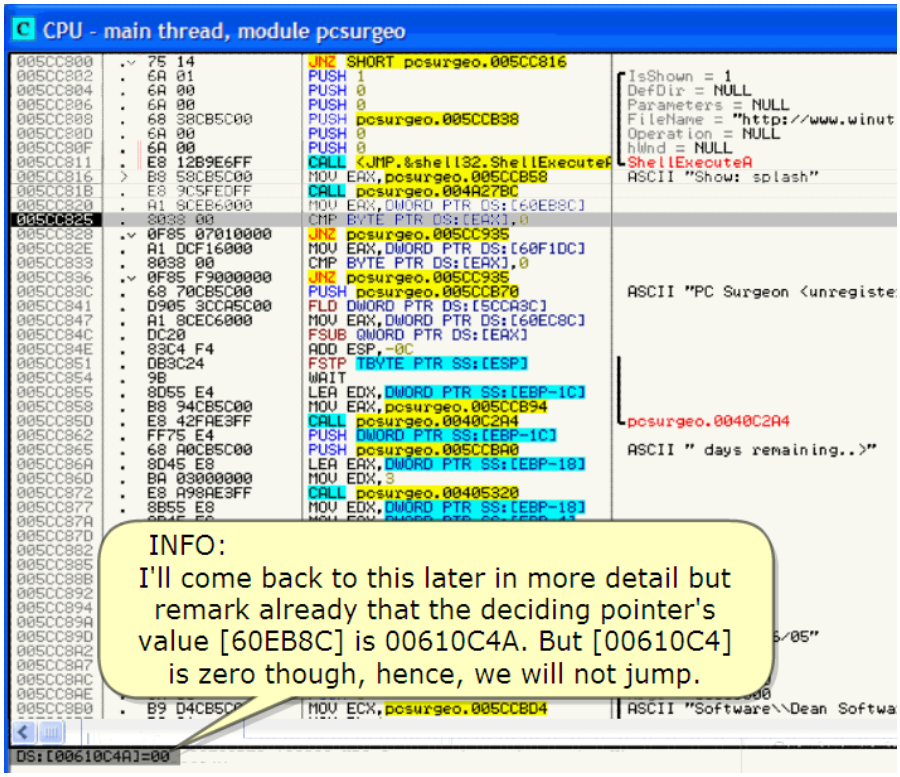
However, we always want to jump here or else we may run into expired at a later date
그러나, 우리는 항상 여기에서 jump 하기를 원한다. 또는 우리는 사용기간이 종료 됐다는 것을 실행하게 될 것이다.
Hence, we need to remember to patch one of these cond jumps to always jump. For now, set BP
그리하여, 우리는 그것들의 조건 jump 를 patch 하고 기억하기 위해 항상 jump 해야 한다. 이제, BP 를 설정하자.
So, let's jump and continue stepping F8
이제 jump 하고 F8 을 눌러 계속하자.
We land here.
우리는 이곳에 도착했다.

Remember from before, this is the place where is decided about setting "unregistered" on the window or not ...
전으로부터 기억해, 이곳은 "미등록" 이거나 등록이거나 설정이 결정된다.
...as well as how many days are still remaining in the evaluation period
좋아, 얼마나 많은 날이 남아있는지 기간을 계산한다.
Continue stepping
계속하자.



Remark that the same pointer is deciding again!

주목. 같은 pointer 가 다시 결정한다.



INFO :

I'll come back to this later in more detail but remark already that the deciding pointer's value [60EB8C] is 00610C4A.

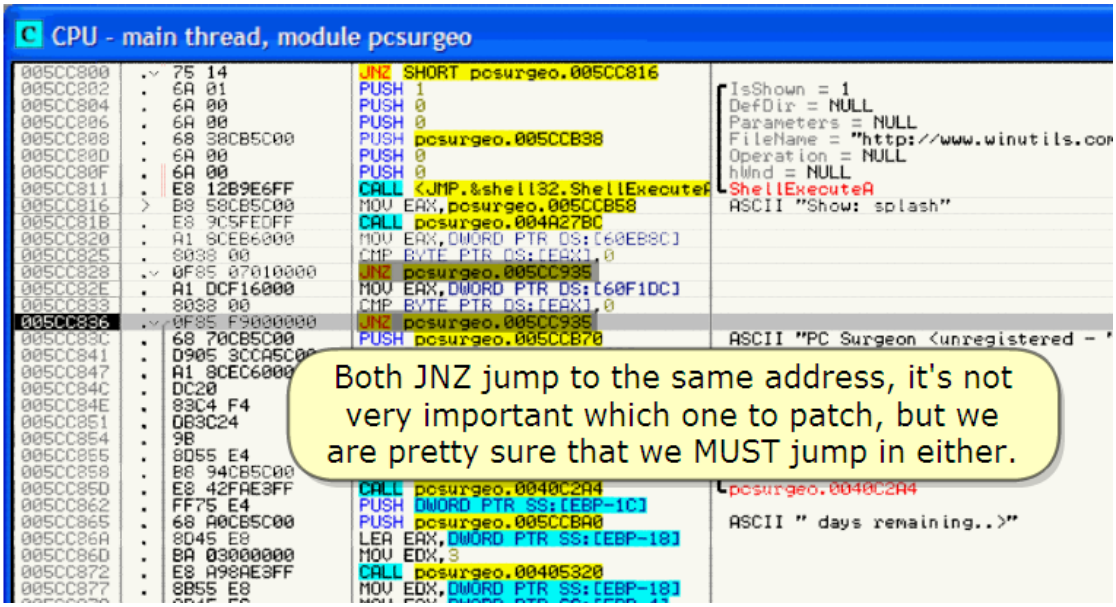
나는 좀 더 자세하게 이곳으로 돌아올 것이다. 이미 [60EB8C]주소의 값이 00610C4A 이다.

But [00610C4] is zero though, hence, we will not jump.

그러나 [00610C4]는 0 다. 그리하여 우리는 jump 하지 않는다.

Step F8

F8 눌러



Both JNZ jump to the same address, it's not very important which one to patch, but we are pretty sure that we MUST jump in either.

2 개의 JNZ jump 는 같은 주소로 향한다. 이것은 패치하기 위해 중요하지 않다. 우리는 확실히 어느 쪽에서든지 jump 를 해야 한다.

Let's jump temporarily and for further investigation but remember to patch one of both cond jumps and save to file later

임시적으로 jump 하자. 그리고 멀리 투자하기 위해. 기억해. 2 개의 조건 jump 중에 하나를 patch 하고 file 로 나중에 저장하자.

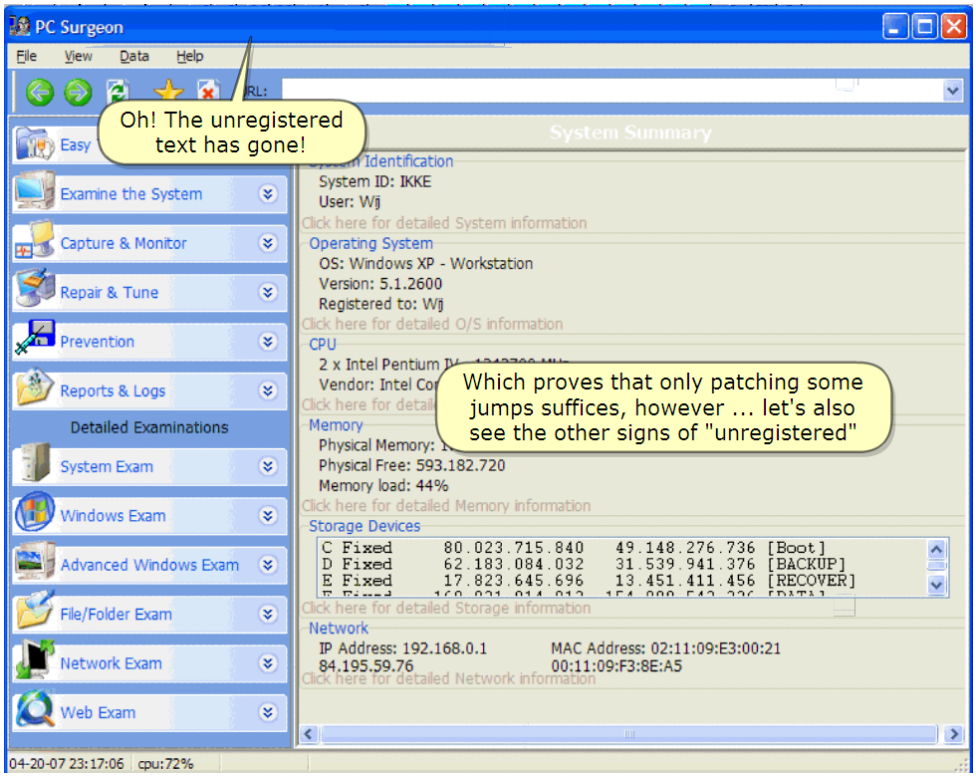
;)

Right. We will jump past setting the unregistered text on the window. So, let's run and verify.

맞아. 우리는 과거의 setting 인 미등록 된 text 를 jump 할 것이다. 실행하고 검증하자.

BAM! No nag!?!

오 오! Nag 없네!?!



Oh! The unregistered text has gone!

오! 미등록 된 text 도 사라졌어

Which proves that only patching some jumps suffices, however ... let's also see the other signs of "unregistered"

약간의 patch 한 jump 로 충분히 증명했다. 그러나 다른 "unregistered" 표시를 본다.

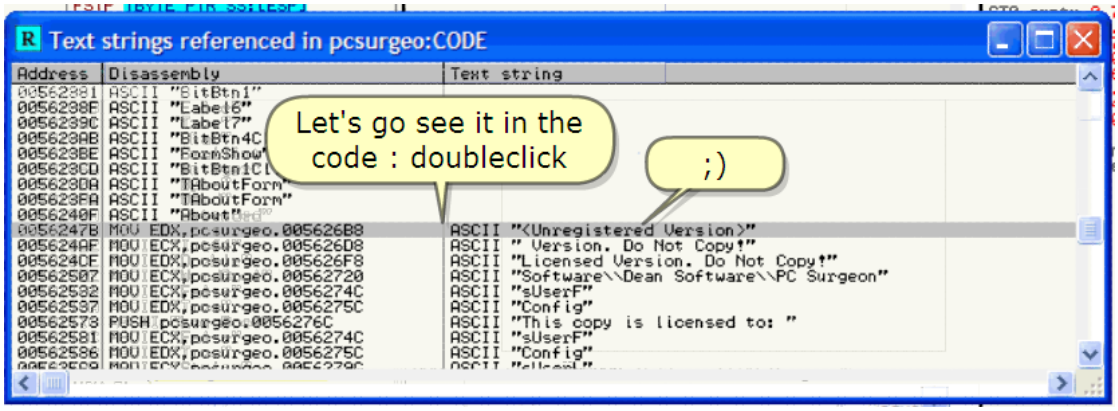
Mmmm, that could be better So, still work to be done ...

좋아질 거야, 일 하는 것을 끝내자.

And let's use the textstrings again

Textstrings 을 다시 사용하자.

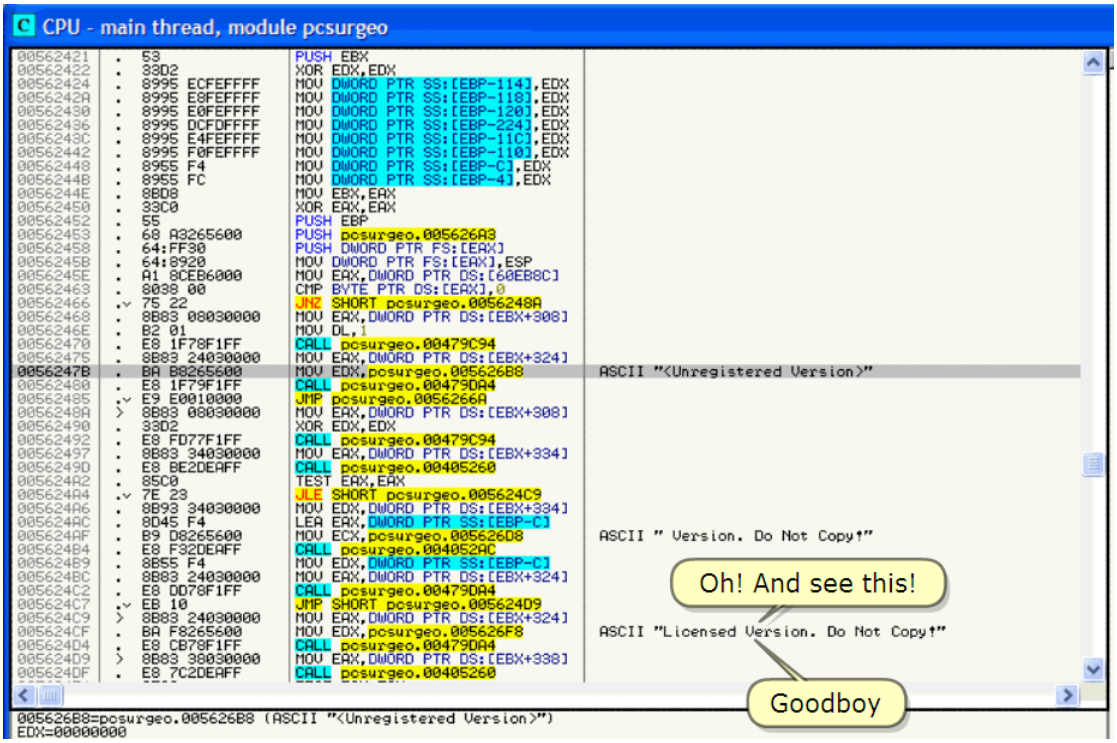
;)



Let's go see it in the code : doubleclick

Code 에서 보자 : doubleclick

;)



Oh! And see this!

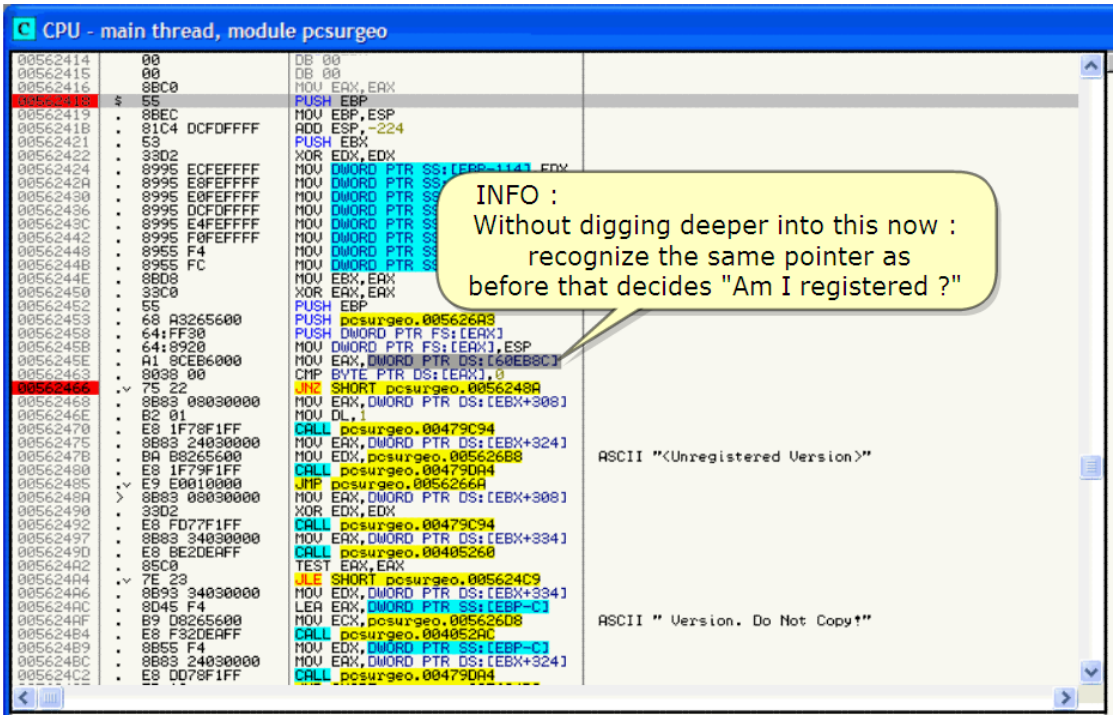
Goodboy

오! 봐!

goodboy

Scroll up to BP the beginning of the routine

Routine 의 시작하는 BP 로 Scroll 올려 봐.



INFO :

Without digging deeper into this now : recognize the same pointer as before that decides "Am I registered ?"

더 깊게 들어가지 않고 : 결정한다. "나 등록됐어?" 할 때 같은 pointer 를 알아보자.

Ok. We found the About Box routine with the setting from these strings on it. Let's restart and make all necessary patches

Ok. 우리는 Box routine 에 대하여 그것들의 string 으로부터 setting 을 찾았다. 재시작하고 모든 필요한 patch 를 하자.

Yep. We break in the beginning of the nag routine. But we can as well immediately run till the first important conditional jump. No need to leave the other BP's set.

예, 우리는 nag routine 의 처음에서 멈췄다. 그러나 우리는 즉시 첫번째로 중요한 조건 jump 까지 실행할 수 있다. 더이상 BP set 이 필요하지 않다.

Remember that these are both cond jumps deciding about jumping expired or not.

기억해. 그것은 2 개의 조건 jump 는 만료되는지 아닌지 결정한다.

Press F2 to remove the BP and scroll down

F2 를 눌러 BP 를 삭제 후 scroll 내려.

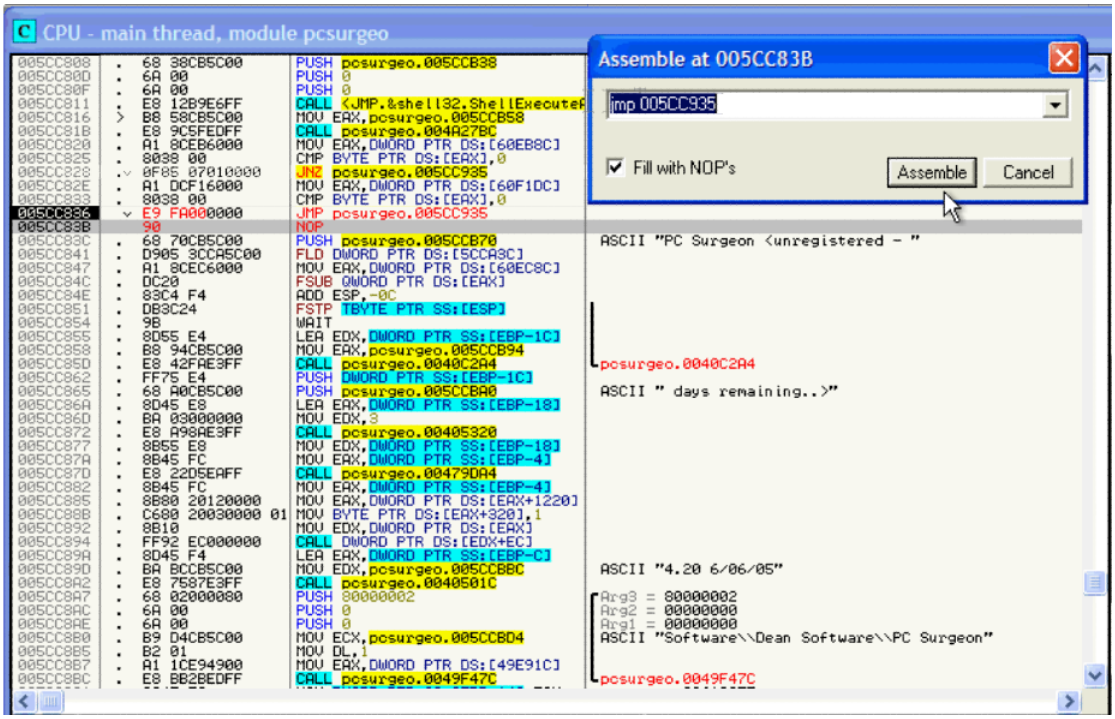
We decided before that we need only adjusting the conditional jump into always jumping to avoid the time trial to end (14 days). So, let's jump and press F8

우리는 그러고나서 조정하는 조건 jump 가 trial time 을 항상 회피해서 jump 한다.(14 일) 결정했다.
Jump 해. F8 을 눌러.

We land here just ...
우리는 여기에 도착한다 ...
... before the conditional jumps that decide on putting the "<unregistered " on the window(or not). Step F8 till the cond jumps
그러고나서 조건 jump 는 "unregistered "를 window 에 넣는 것을 결정한다. 조건 jump 까지 F8.

Indeed, we will need to help the application find the right direction :)
Remove BP and assemble the patch to always jump.
정말, 우리는 application 이 좋은 방향을 찾기 위해 필요하다.
BP 를 삭제 후 항상 jump 하게끔 patch 한다.

INFO :
If you leave the software BP (INT3) on a re-assembled line, Olly gets confused when the application is restarted because he needs to put a software breakpoint before "corrupted" code.
만약에 re-assemblend 된 line 의 software BP 를 남겨놓았다면, Application 을 재시작 한 후에 Olly 는 혼란을 겪는다. 왜냐하면 Olly 는 software breakpoint 가 필요하다. 그러고나서 "충돌"
You can leave it of course, Olly will warn you that he will set the BP inactive, you then just need to activate the BP again here.
너는 물론 남길 수 있다. Olly 는 너에게 경고한다. BP 를 설정하여 대화할 수 있고, 활동적인 BP 가 다시 필요하다.
This is the Breakpoints window. You can easily study this yourself.
이것은 Breakpoints window 다. 너는 쉽게 배울 수 있다.



:)

Ok. These first patches were needed to correct the time trial and also to correct the text being put as window caption.

Ok. 이 첫번째 patch 들은 정확한 time trial 과 정확한 window caption text 가 필요했다.

Remember that we still need to correct the Unregistered string in the About box.

기억해. 우리는 About box 에서 훨씬 정확한 Unregistered string 이 필요하다.

We are on the JMP instruction now.

우리는 JMP 로 바꿨다.

Let's press run to be able to break in the creation of the "About" box.

만들어진 "About" box 에서 멈출 수 있게끔 실행하자.

Indeed, the <unregistered is no longer in the window caption

정말, window caption 에서 <unregistered 는 더 이상 없다.

Let's patch the About Box too

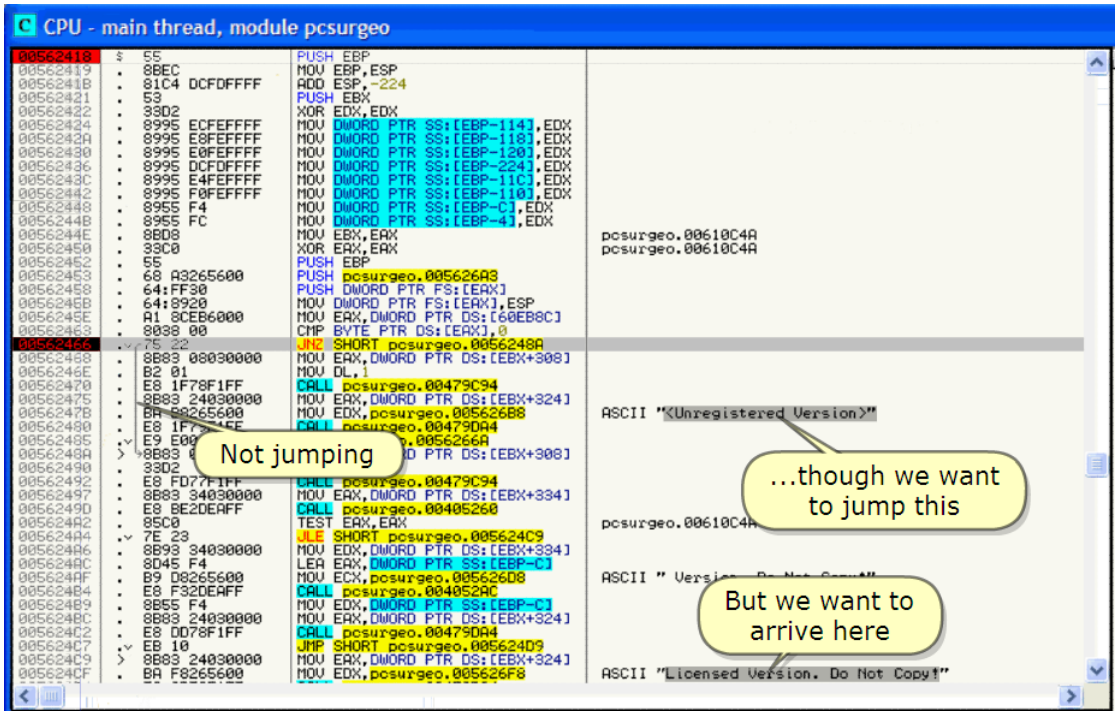
About box 를 patch 하자.

We land in our BP.

Run till the JNZ who is awaiting our patching

우리는 BP 에 도착했다.

JNZ 까지 실행하자. 그곳에서 우리의 patch 를 기다리고 있다.



Not jumping

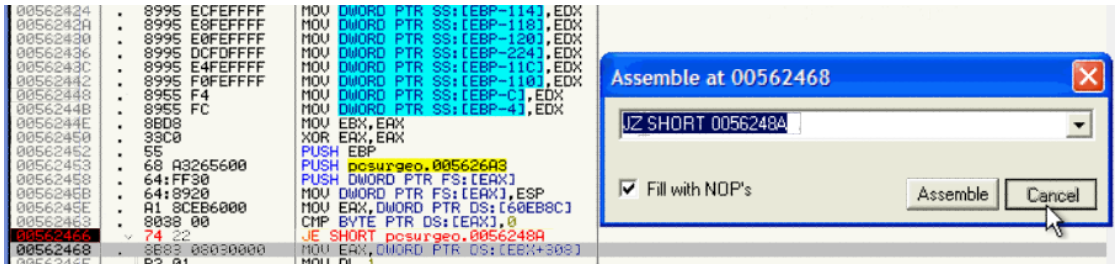
But we want to arrive here

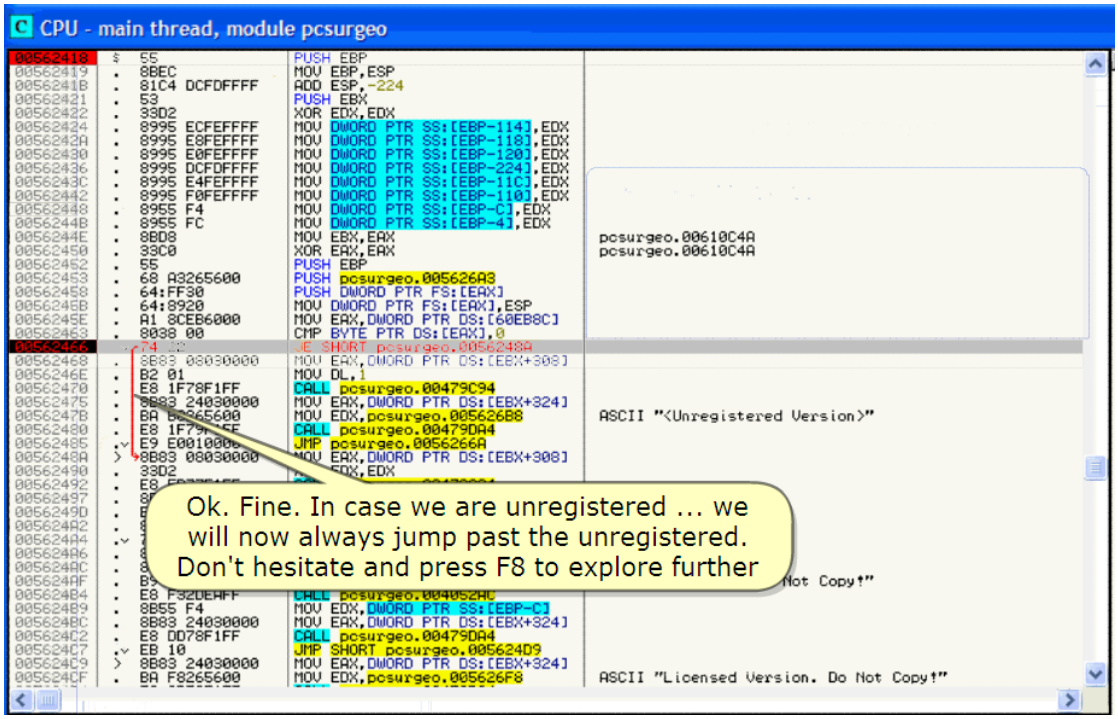
...though we want to jump this

Jump 하지 않는다.

그러나 우리는 우리는 이곳에 도착해야 한다.

우리는 jump 를 통하여



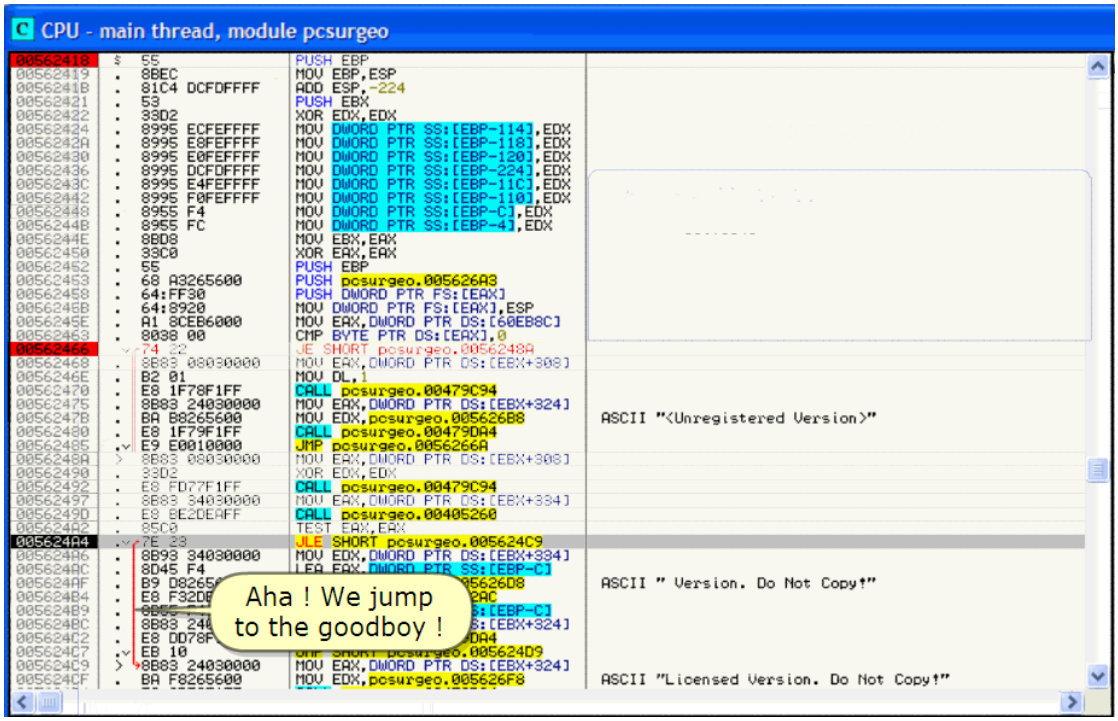


번역 주)JZ(jump if zero) = JE(jump if equal)와 같기 때문에

Ok. Fine. In case we are unregistered ... we will now always jump past the unregistered. Don't hesitate and press F8 to explore further

좋아. 이 case 는 우리가 unregistered 했다. 우리는 항상 unregistered 를 jump 해야 한다.

망설이지 마. F8 로 멀리 탐험하자.



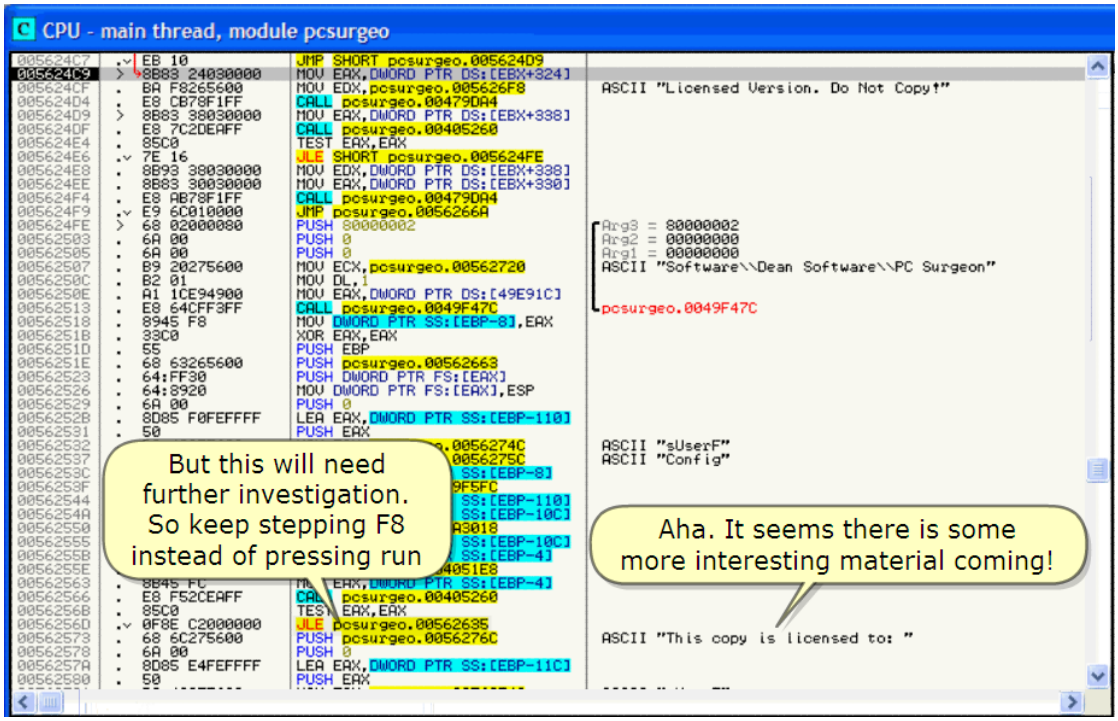
Aha! We jump to the goodboy !

아하! 우리는 goodboy 로 jump 한다.

!

Scroll down to see the way ahead ;)

Scroll 미리 앞 부분을 보기 내려봐.



Aha. It seems there is some more interesting material coming!

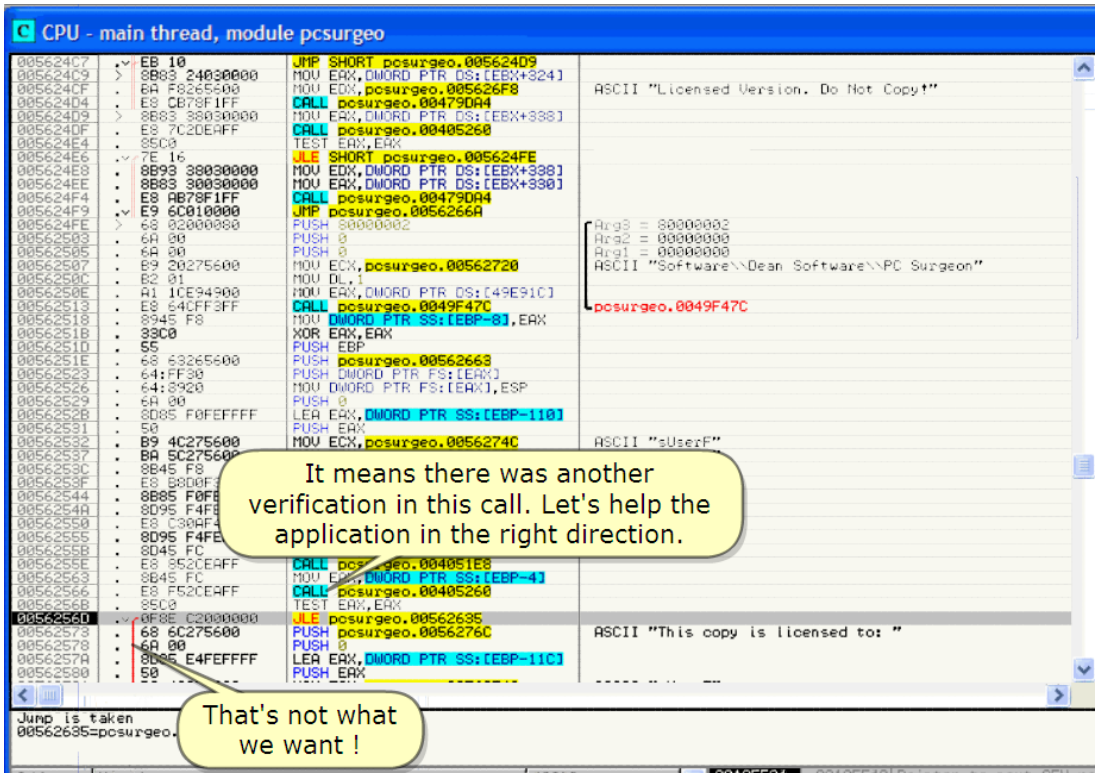
아하. 좀 더 재미있는 문구가 보인다.

But this will need further investigation.

So keep stepping F8 instead of pressing run

그러나 우리는 멀리 조사가 필요하다.

F8 을 눌러 code 를 진행하자.



That's not what we want!

It means there was another verification in this call.

Let's help the application in the right direction.

우리가 원한 게 아냐!

이것은 call 안에서 다른 검증이 있다는 것이다.

Application 이 올바른 방향으로 갈 수 있도록 도와줘.

Remember : NOP = do nothing

--> the code below will always execute

번역 주)(NOP = no operation)

기억 : NOP = 아무 동작 안함

--> 항상 우리가 실행 시킨 밑에 code 가 있다.

INFO :

Of course there are other possibilities again and we can always choose to assemble other instructions that achieve the same. (See previous parts).

물론 다른 가능성이 있다. 우리는 똑같은 결과를 얻기 위해 assemble 할 때 다른 명령어를 선택할 수 있다. (예전 part 를 봐)

For best comprehension, I will use the simplest options like NOP'ing though.

가장 좋은 이해력은, 나는 NOP 과 같이 간단한 option 을 사용할 것이다.

And now, let's finally try our patches

이제, 마지막으로 우리의 patch 를 보자.

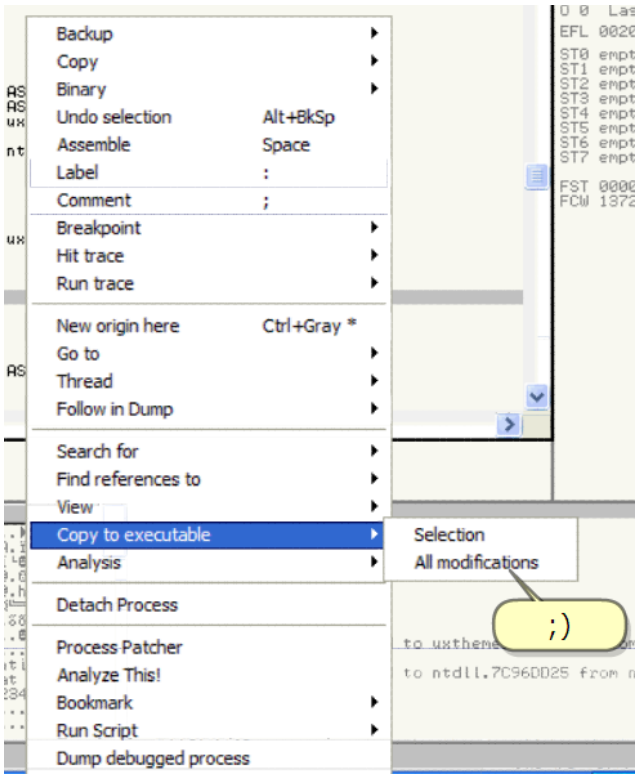
Aha! That looks quite better than before!
And here too!
아하! 전보다 꽤 좋은데
여기도!

We still need to save the patches to file.
Return to Olly
우리는 patch 된 것을 file 로 저장하는 것이 필요하다.
Olly 로 돌아가자.

5. Testing the patched application

We've made/found several patches so far.
우리는 지금까지 몇가지 patch 를 만들고/찾았다.
So, we will have to save "All changes" to file.
그래서, 우리는 file 로 저장하기 위한 "All changes"가 있다.
Right-click to do so.
rightclick 하고 해봐.

;)



;)

;)

Saving under a different name

다른 이름으로 저장해

Let's test the patches

Patch 된 것을 test 해보자.

;)

;)

No more splash screen ...

더 이상 필요없는 screen 이 없다.

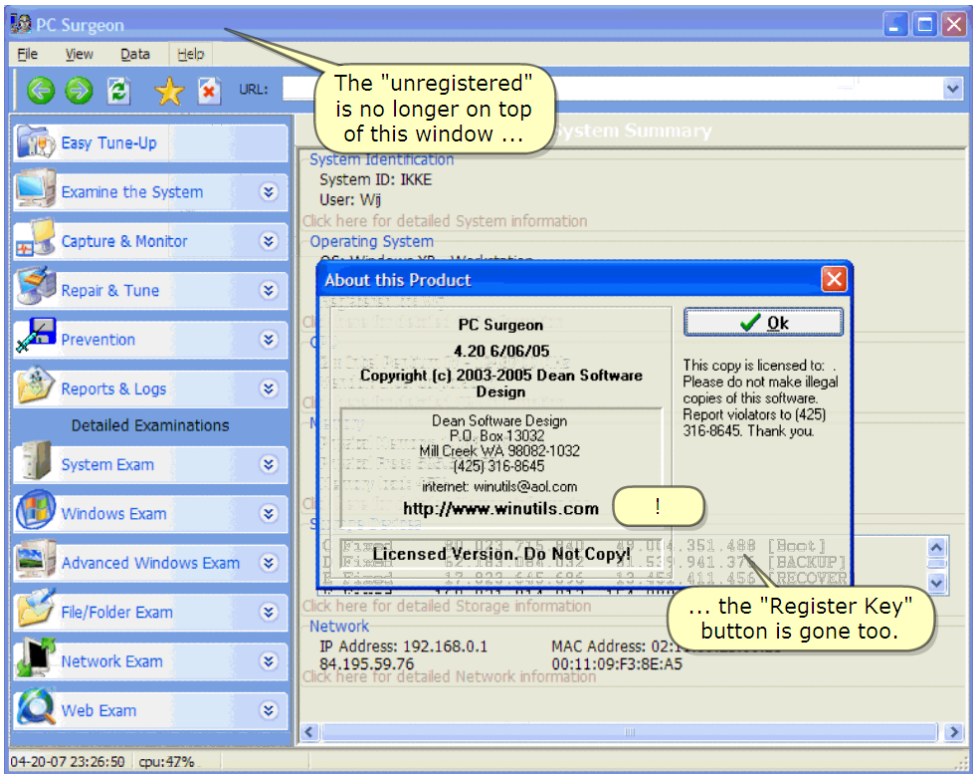
No more unregistered text ...

더 이상 unregistered text 가 없다.

Let's see the about box

About box 를 보자.

!



... the "Register Key" button is gone too.

The "unregistered" is no longer on top of this window ...

"Register Key" button 이 사라졌다.

Windows 의 top 에 "unregistered"도 더 이상 없다.

So, we did a nice job! In fact, our goal is achieved and the software runs fine, nor will it stop working (time limit)
그래서, 우리는 좋은 일을 했다! 사실, 우리의 목표는 달성했다. 그리고 software 도 좋게 실행된다. Time limit 가 돼서 멈출 때까지

CONSIDERATION :
고려사항
Have you understood from this Part06 that in fact, the main protection in this program is the time limit?
Part06 의 사실을 이해했어? Program 에서 main protection 이 time limit 다.
You may want to try it out on your own: once the time limit is patched, the program will keep running fine although displaying the ugly nag and badboys.
너는 도전하기를 원할 것이다 : time limit 가 patch 된 것을, program 은 못생긴 nag 과 badboy 가 보여도 잘 실행되게 유지할 것이다.
Both other patches are only aesthetical patches (top of window kills also the nag and About Box string patch does nothing more but no longer displaying the unregistered string)
2 가지 patch 는 오직 미적인 patch 다. (window top 에 있는 nag 을 삭제한 것과 About box 에서 더 이상 unregistered string 이 display 되지 않게 patch 한 것도.)

We succeeded in patching this application into registered only by patching some conditional jumps (== Plain Stupid Patching).
우리는 약간의 조건 jump 를 patch 하여 registered 하게 application patch 에 성공했다.(== 명백히 멍청한 patch 다)
The job is finished here. However, and only in the light of a more complete understanding and preparation for the next soft ...
일은 여기에서 끝났다. 하지만, 좀 더 완벽하고 다음을 준비하기 위해서.
I already want to show some more thinking to make it possible to get right to some more difficult material for next part.
나는 다음 part 를 위해 많은 생각들을 가능한 약간 어려운 방법으로 보여주기를 원한다.
Hence, let's return back to the original executable and load and run it in Olly to break in the very first BP
그리하여, original executable 로 돌아가. 그리고 load 하고 첫번째 BP 까지 실행해.

6. A better way

;)
Right! We break in the very first BP that we set. (Setting the text on the main window)
맞아! 우리는 우리가 set 했던 첫번째 BP 에 멈췄다.(text 를 setting 해)

Instead of concentrating on patching the jumps only, let's concentrate a short while on the underlying deeper reasons for jumping! Indeed: why is the software effectively deciding on jumping or not?

오직 jump patch 보다 집중해보자. Jump 하기 위해 근본적으로 깊은 이유를 집중해보자. : 효과적인 jump 는 어떻게 결정돼지?

Right, it is because of the setting (true or false) of our famous pointer [60EB8C].

좋아, 이것은 유명한 pointer[60EB8C]에서 setting 해야 한다.

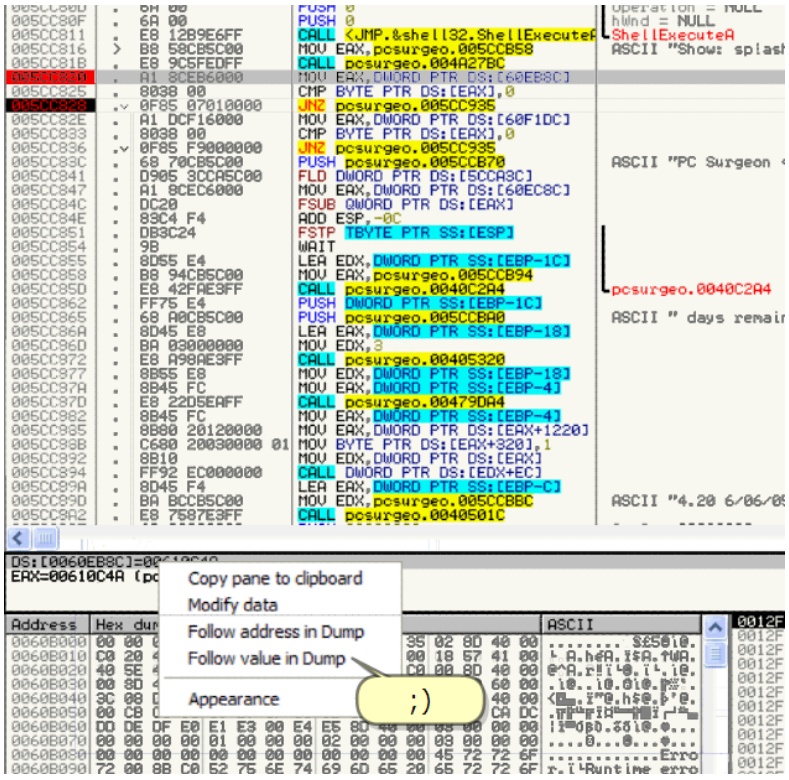
Now, think with me for a moment: if the pointer can be true (1) or false (0), that also means there must be a mechanism somewhere that decides on setting the pointer to true for registered! And finding this mechanism is mostly very easy.

이제 나와 같이 잠시 생각해 보자 : 만약에 pointer 가 1 이거나 0 일 때, 그 뜻은 mechanism 이 있다. 등록하기 위해 true pointer 는 결정한다. 그리고 mechanism 을 매우 쉽게 찾는다.

Just follow along.

날 따라와.

;)



This the byte that will be compared ...

... in the next line (with zero to decide on jumping or not).

Byte 는 다음 line 에서 Pointer 는 00610C4A 를 가리킨다.(zero 는 jump 할지 말지 결정한다)

INFO:

As always, we have different ways to find the mechanism that sets the pointer.

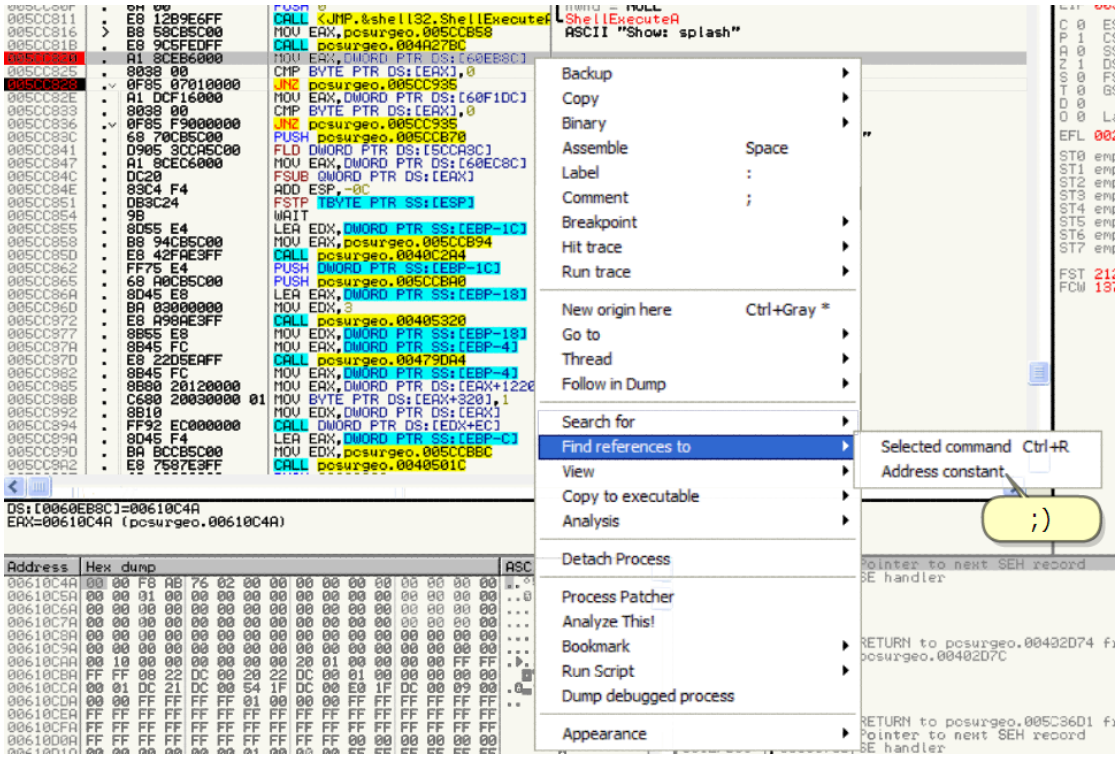
항상, 우리는 pointer 를 set 할 수 있는 다른 방법을 찾아보자.

One way is using a HW or MEM BP on write (or access) on this byte, but let's leave this for later (it's a little more complicated) because we have another way that is very easy:

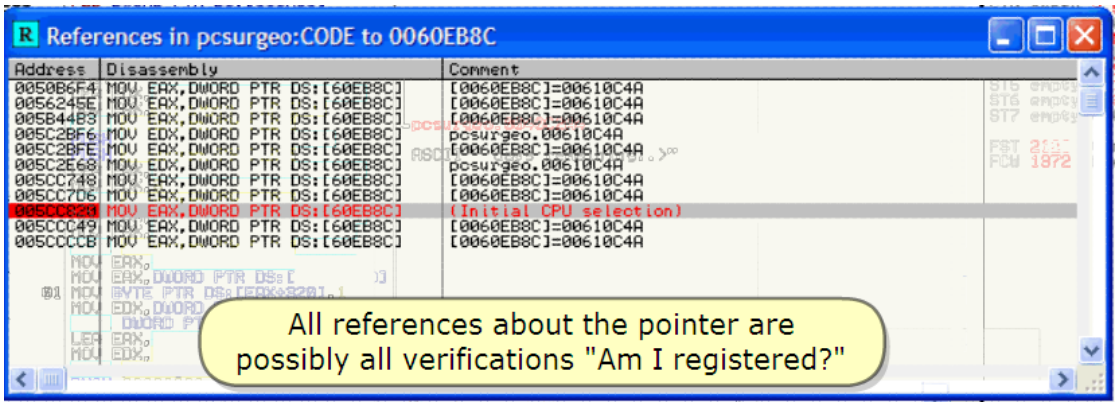
한 가지 방법은 HW 나 Memory BP 를 byte 에 사용하는 것이다. 그러나 나중을 위해 이것을 남긴다.(꽤 복잡하다) 왜냐하면 우리는 다른 방법을 가지고 있는데 그것은 매우 쉽다. :

finding the references to the pointer. Just follow along!

Pointer 에 대한 reference 를 찾는다. 따라와!



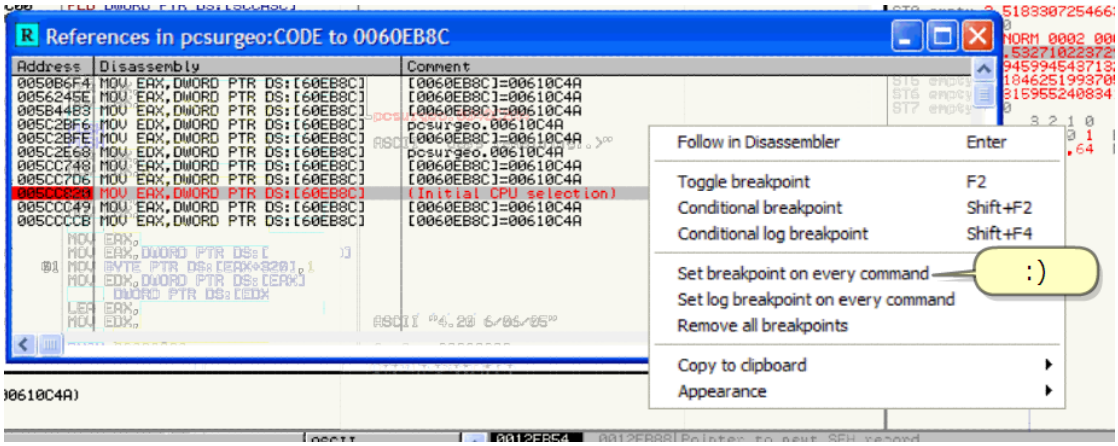
;)



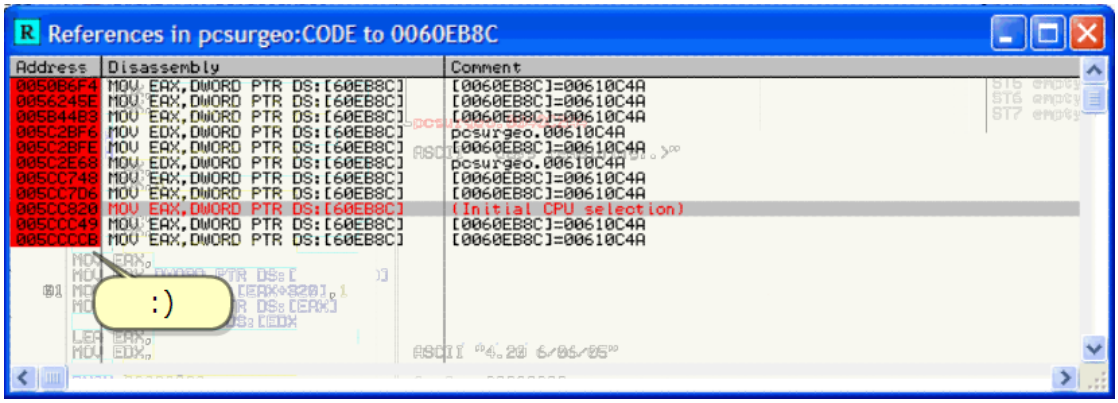
All references about the pointer are possibly all verifications "Am I registered?"

모든 reference pointer 는 가능한 검증이 있다."등록됐어?"

INFO:
The idea is now to set BP's on every reference to the pointer so that we can break when restarting.
Idea 는 모든 참조에서 BP 를 pointer 에 설치하고 restart 할 때 우리는 멈출 수 있다.
Sooner or later (probably even the first or second break) will be the setting for the pointers value.
Thus, which is the setting for registered!
가깝거나 나중에(아마 첫번째나 2 번째 break 조차) pointer value 를 위해 setting 될 것이다.
Registered 를 위해 setting 한다.



:)

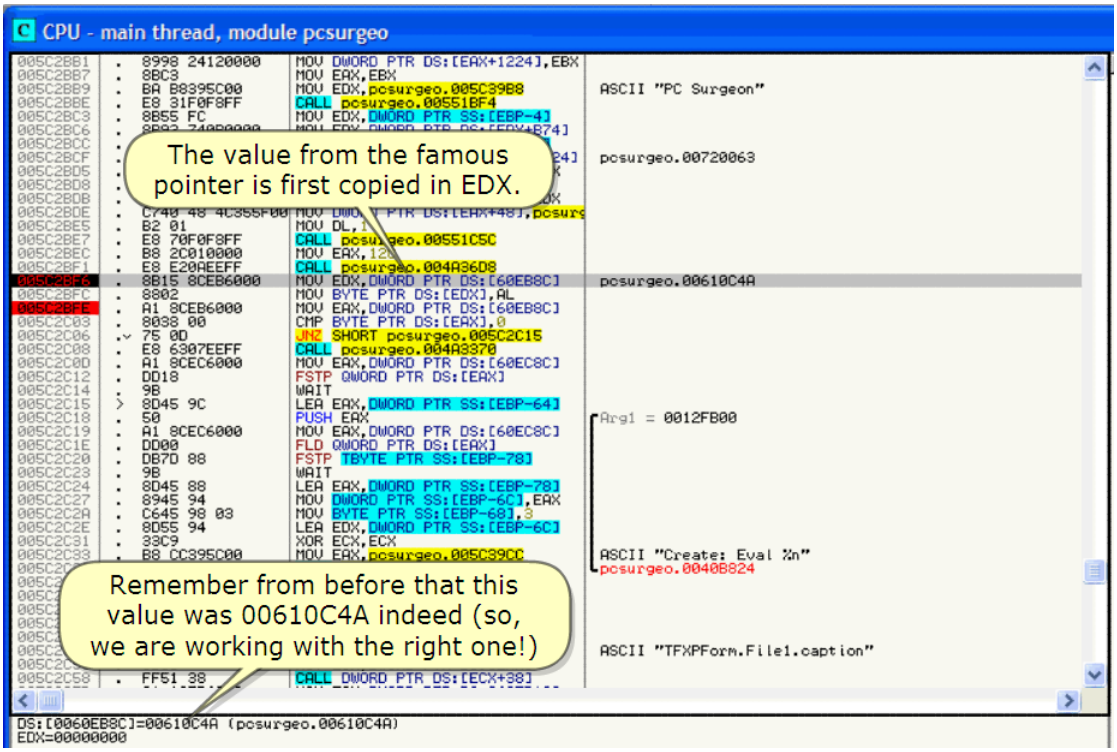


:)

;)

;)

BAM! Fine, we break in one of the BP's.
Is it the setting for the pointer?
Let's study whilst stepping the code.
좋아, 우리는 BP's 의 하나에 멈췄다.
Pointer 를 위해 setting 됐어?
Code 를 넘기면서 공부하자

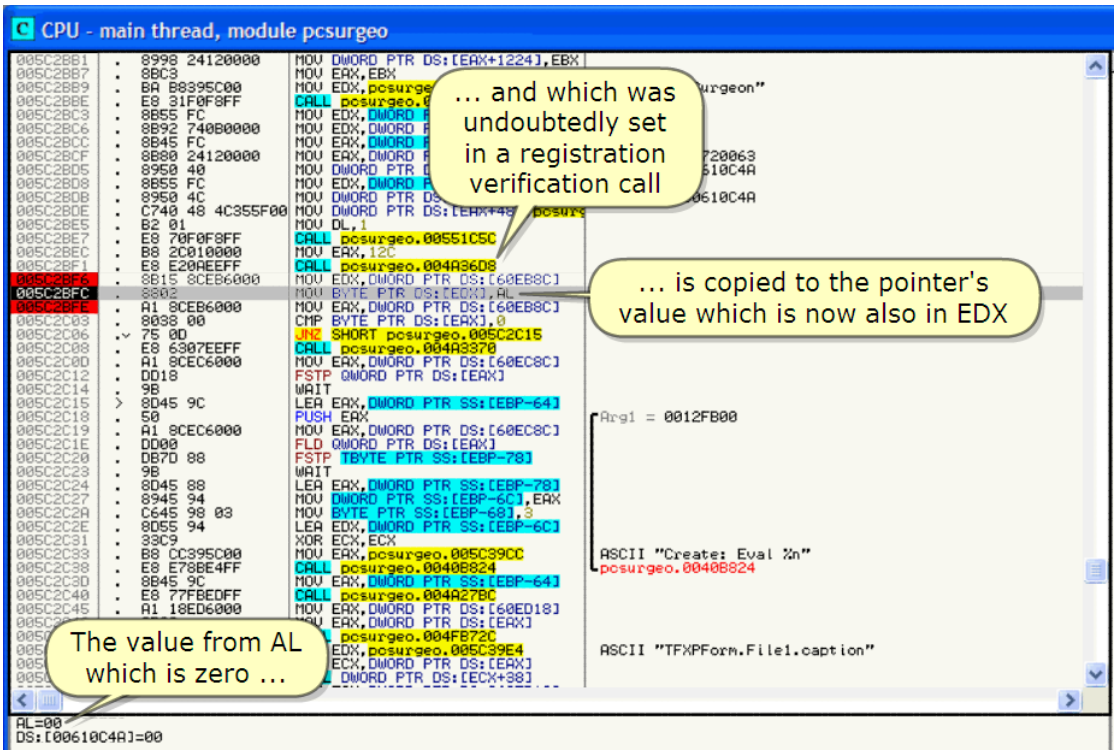


The value from the famous pointer is first copied in EDX.

Famous pointer 의 Value 는 첫번째로 EDX 에 copy 된다.

Remember from before that this value was 00610C4A indeed (so, we are working with the right one!)

기억해. 이 value 는 00610C4A 였어.(그래서, 우리는 right one 과 일 할 수 있다)



The value from AL which is zero ...

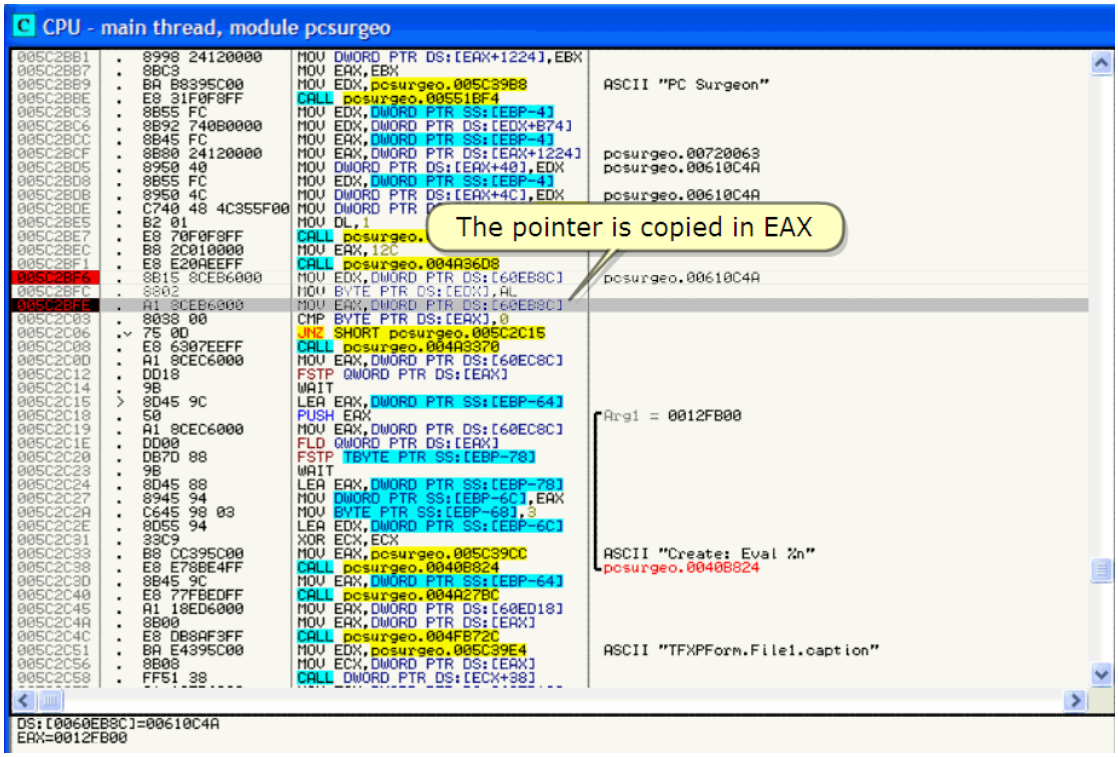
AI 값은 zero 다.

... and which was undoubtedly set in a registration verification call

... is copied to the pointer's value which is now also in EDX

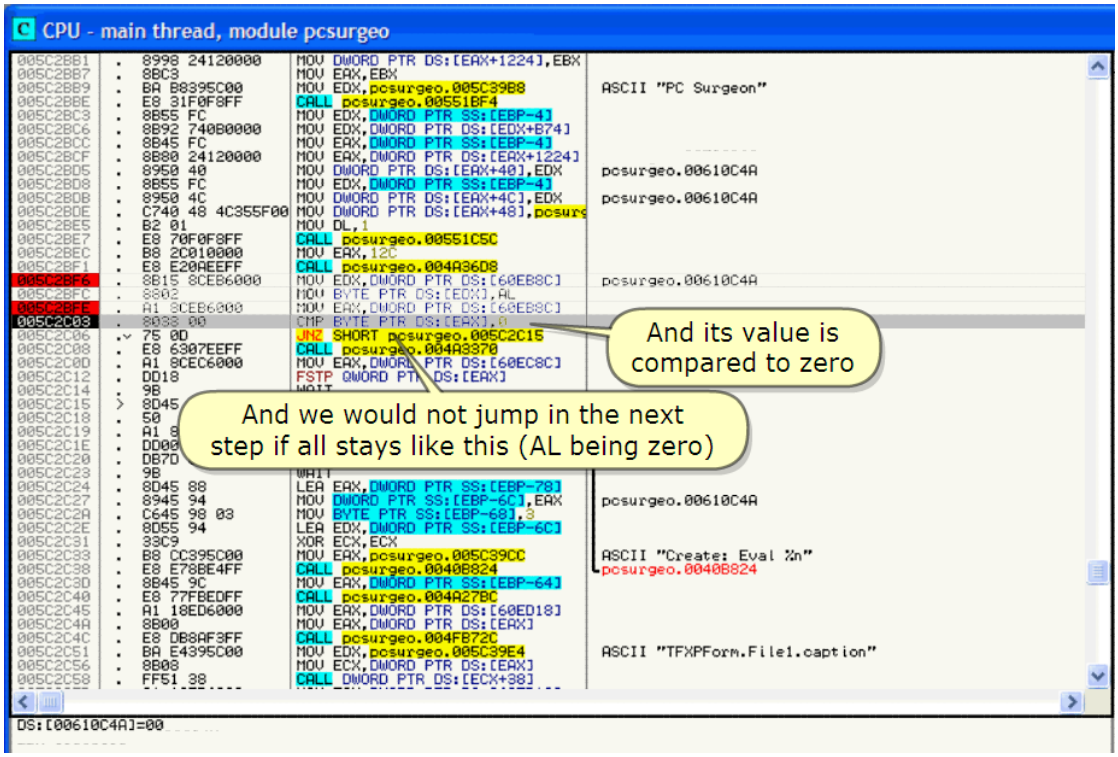
Registration 검증 call 에서 확실히 set 됐다.

pointer value 값을 EDX 로 복사했다.



The pointer is copied in EAX

Pointer 값이 EAX 로 복사됐다.

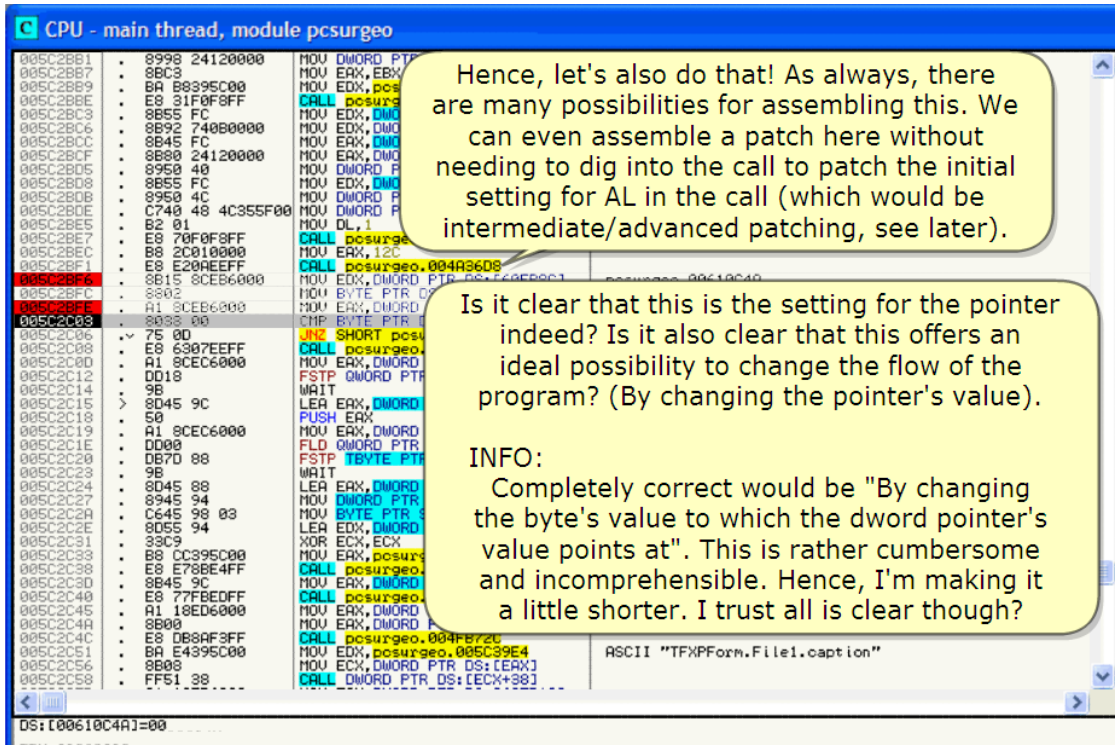


And its value is compared to zero

이 값은 zero 와 비교됐다.

And we would not jump in the next step if all stays like this (AL being zero)

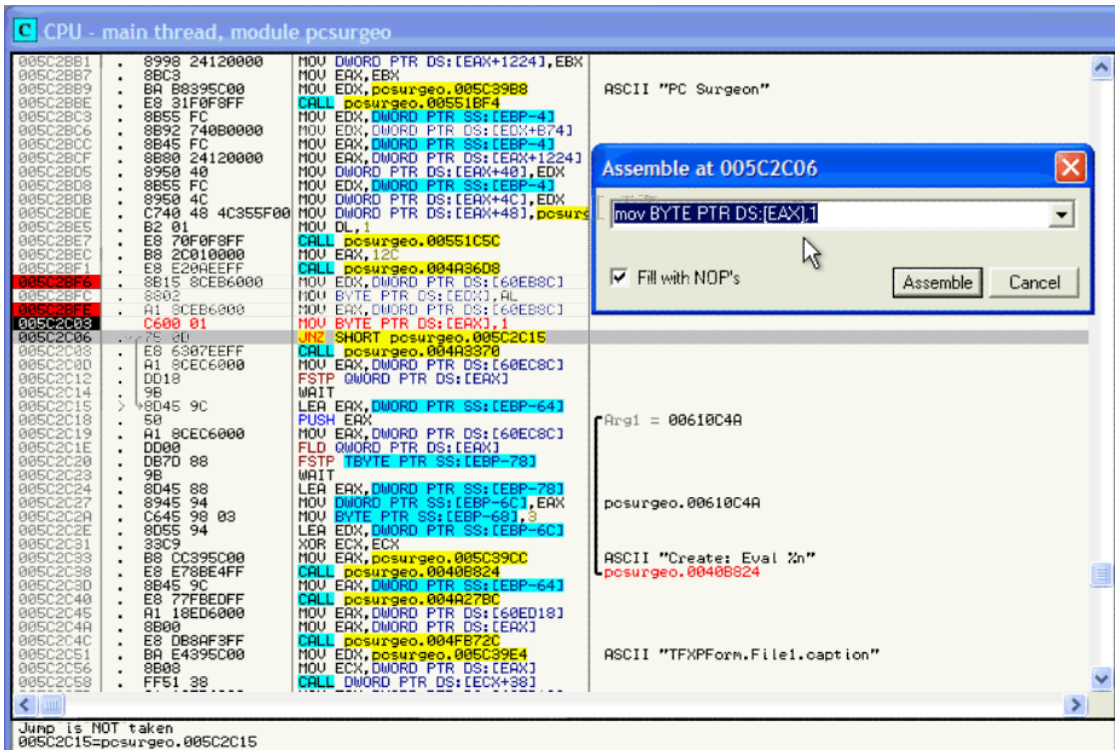
그리고 모든 것이 이렇게 있다면(AL 이 zero) 우리는 다음 step 으로 jump 할 수 없다.



It is clear that this is the setting for the pointer indeed? Is it also clear that this offers an ideal possibility to change the flow of the program? (By changing the pointer's value).

이것은 명확하다. Pointer 를 위해 setting 된다. 이것은 명확하다. 이것은 program 의 flow 를 바꾸는데 이상적인 가능성을 제공한다.(pointer's value 를 바꾼다)

INFO:
Completely correct would be "By changing the byte's value to which the dword pointer's value points at".
완벽히 정확하다. "dword pointer value 의 값으로 byte 의 값을 변경"
This is rather cumbersome and incomprehensible.
이것은 꽤 무겁고 이해할 수 없다.
Hence, I'm making it a little shorter.
그리하여, 나는 적게 바꿀 것이다.
I trust all is clear though?
모든 게 명확해졌다고 믿는다.
Hence, let's also do that! As always, there are many possibilities for assembling this.
그리하여, 해보자! 항상, 그곳에는 이것을 assembling 하는데 많은 가능성이 있다.
We can even assemble a patch here without needing to dig into the call to patch the initial setting for AL in the call (which would be intermediate/advanced patching, see later).
우리는 이곳 patch 를 call 에서 AL 의 initial setting 을 patch 하기 위해 깊게 들어가지 않아도 assemble 할 수 있다.(중간/진화 patching, 나중에 보자)



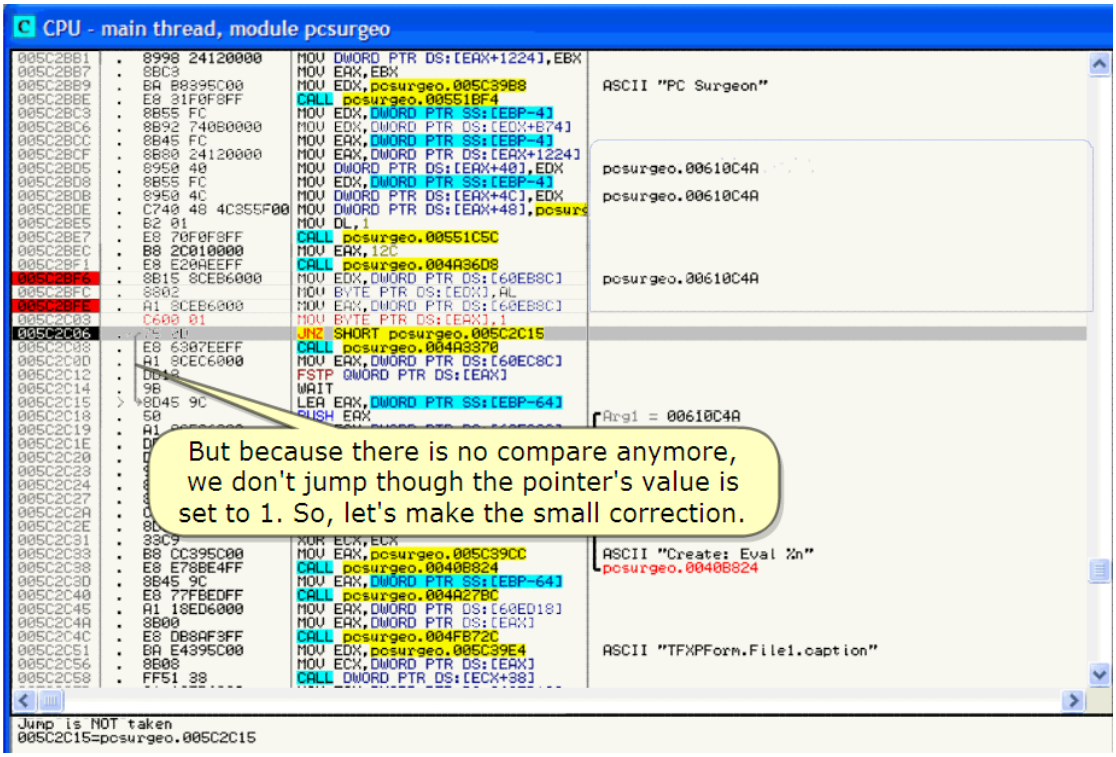
;)

We are about to write "True" in the pointer's value.

우리는 pointer value 에 "True"를 쓰기 위해 있다.

Step F8 to do it!

F8 로 해.

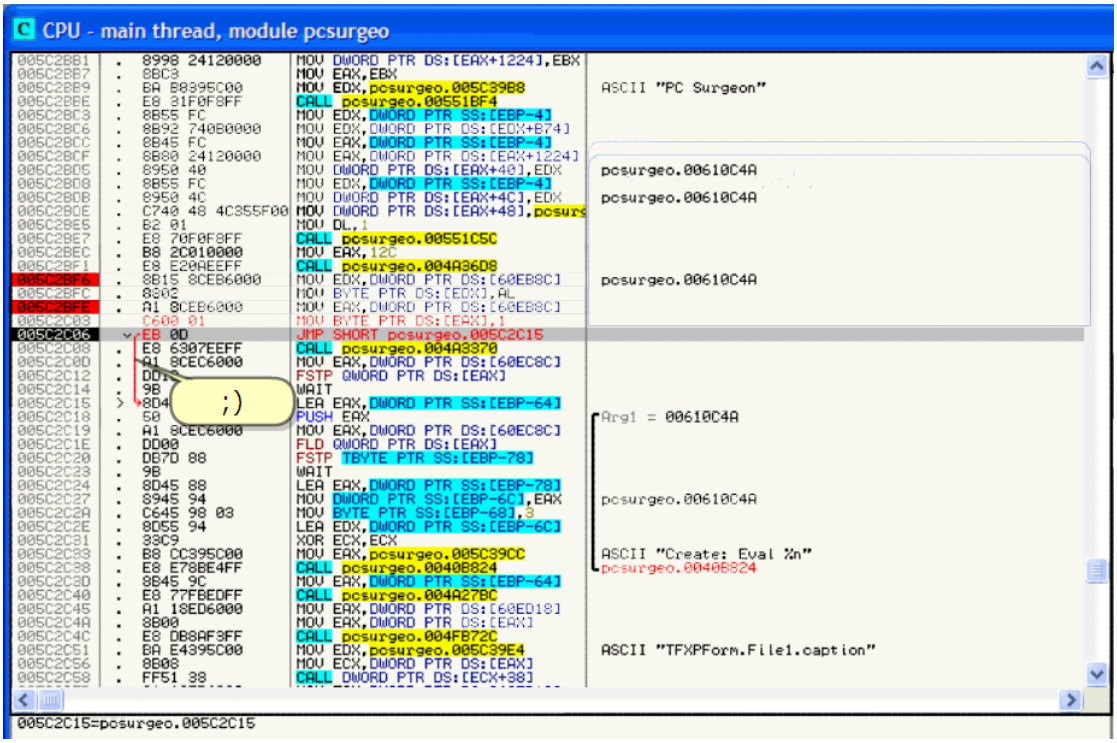


But because there is no compare anymore, we don't jump though the pointer's value is set to 1.

그러나 그곳은 더 이상 비교하는 게 없다. 우리는 pointer's value 가 1 로 set 되서 jump 할 수 없다.

So, let's make the small correction.

작게 교정하자.



;)

There are still a lot of BP's set. Normally, one would now run to break and verify for the pointer's correct setting (and also doublechecks in the code).

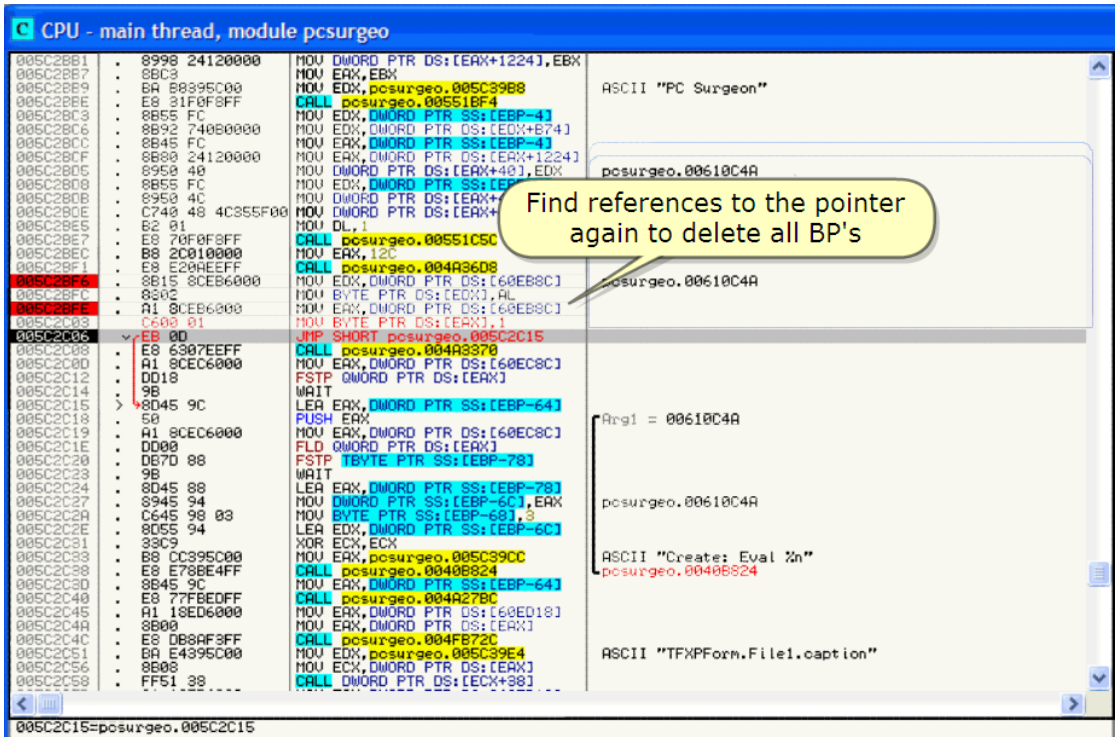
그곳에는 아직까지 많은 BP's 가 설정되어 있다. 보통, 하나가 멈추기 위해 실행될 것이다. 그리고 poitner's 교정 setting 이 맞는지 검증한다(code 내에서 doublecheck)

In this software are no doublechecks and stuff.

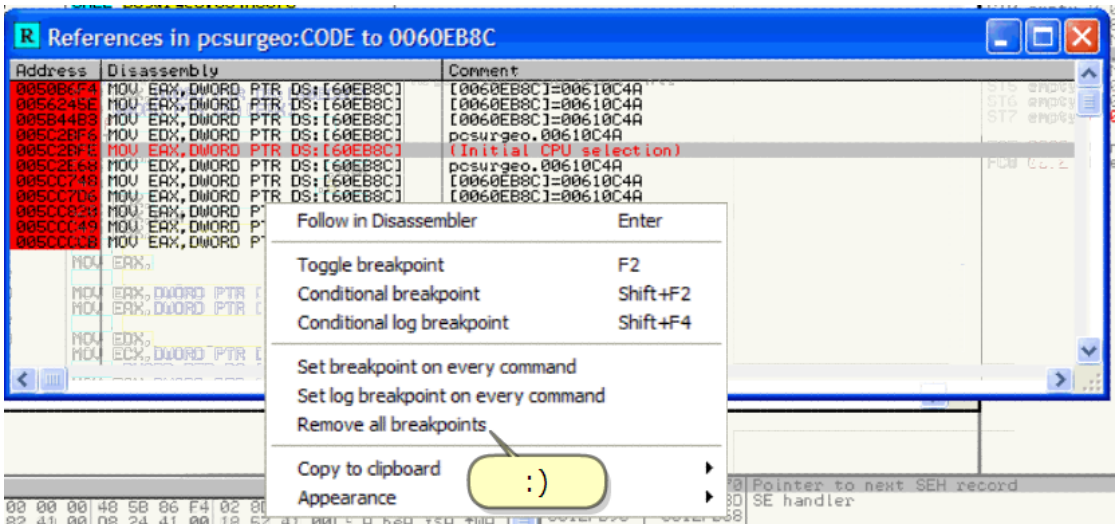
이 software 에서는 doublecheck 가 없다.

Hence, let's just remove all BP's and run to see the result of these easy patches.

그리하여, all BP's 를 삭제하자. 그리고 그것들을 쉽게 patch 한 결과가 어떤지 보자.



Find references to the pointer again to delete all BP's



:)

And test the patches...

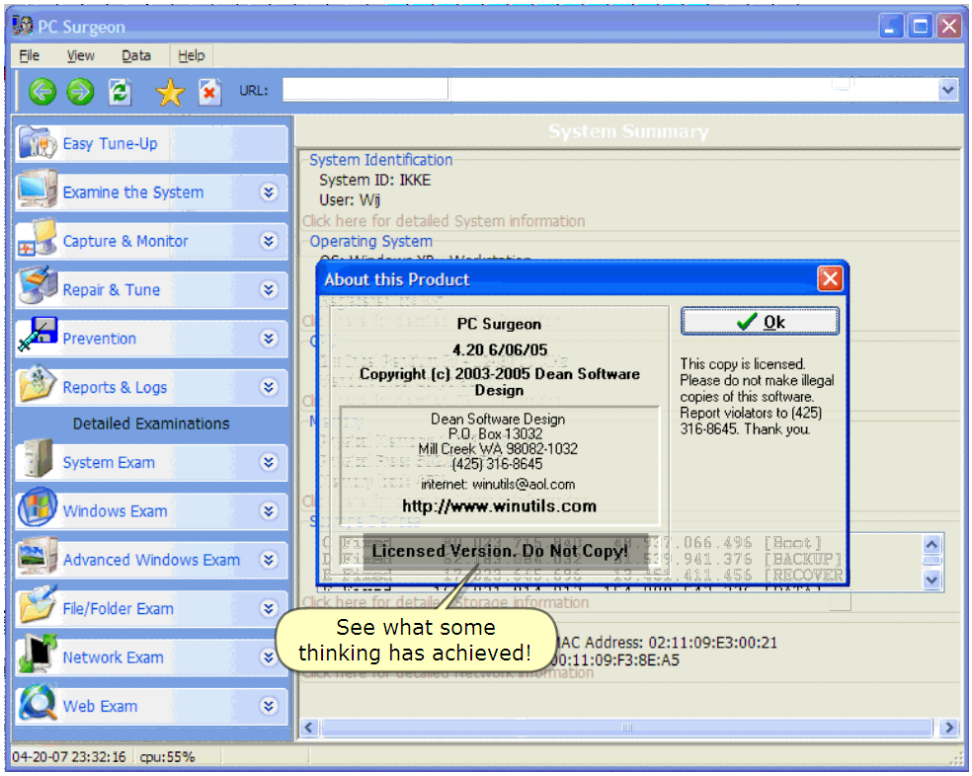
Patch 된 것을 test 하자.

We break in the leftover BP but see that we will jump the setting for "unregistered" on the main window!

우리는 BP 의 잔재에 멈췄다. 그러나 "unregistered"를 위해 우리는 setting 하여 jump 할 수 있다.

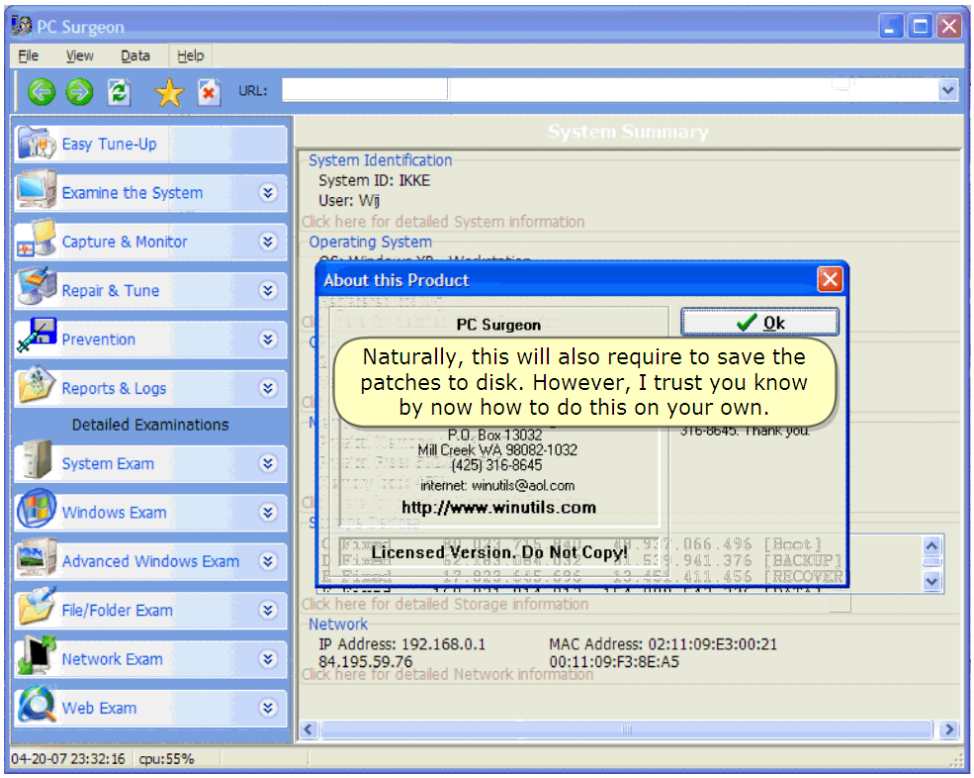
;)

See About Box



See what some thinking has achieved!

무엇을 이루었는지 봐.



Naturally, this will also require to save the patches to disk. However, I trust you know by now how to do this on your own.

보통, 이것은 patch 된 것을 disk 로 저장하기 위해 요구할 것이다. 그러나 나는 네가 어떻게 이것을 하는지 이해할 것이라 믿는다.

7. Conclusion

In this program, it was really easy to find the patches to register the application.
이 program 에서는, application 을 등록하는 patch 를 찾는 게 정말 쉽다.
It sufficed to patch it "The plain stupid method" to turn it into fully registered.
이것은 "The plain stupid method"를 바꿔 충분히 등록되게 patch 하기 위해 충분하다.
We have found that there really isn't much necessary to patch such a program.
우리는 program 을 patch 하기 위해 그것이 정말 필요하지 않다는 것을 찾았다.
The author would better see this to find a better protection for his work.
저자는 그의 작업에 대한 더 나은 보호 기능을 찾으려면 이 곳을 볼 것이다.
In this part 6 of this reversing series, the primary goal was to study the behavior of a program when applying "The Plain Stupid Method" of patching.
이번 part 6 reversing series 에서는, 중요한 목표는 "The plain Stupid method" patch 를 지원할 때 program 의 behaviour 를 배우는 것이다.
I hope you understood everything fine and I also hope someone somewhere learned something from this. See me back in part 06 ;)
모든 것을 잘 이해했기를 바란다. 그리고 나는 누구든지 어느 곳에서든지 이것에서 무엇이든지 배웠기를 바란다. Part 06 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.