

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

2012년 1월 28일 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 6에 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\*864로 설정해 그리고 full screen으로 movie를 보기 위해 F11를 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 screen을 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.

Movie는 text와 다음 screen을 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 button과 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 text와 다음 screen을 보기 위해 hotspot을 click해.

During the whole movie you can click this spot to leave immediately

이 movie 어디에서나 즉시 떠나기 위해 이 spot을 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 6에서, 우리는 무언가 기본적인 patch 를 배우기 위해 "real" application 을 reverse 할 것이다.

기본적인 patching 은 "분명히 명청한 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.  
Again, let's do that together.

| Address  | OpCode       | Instruction                 |
|----------|--------------|-----------------------------|
| 0060A8EC | 55           | PUSH EBP                    |
| 0060A8ED | 8BEC         | MOV EBP,ESP                 |
| 0060A8EF | B9 15 000000 | MOV ECX,15                  |
| 0060A8F4 | > 6A 00      | PUSH 0                      |
| 0060A8F6 | 6A 00        | PUSH 0                      |
| 0060A8F8 | 49           | DEC ECX                     |
| 0060A8F9 | 75 F9        | JNZ SHORT pcsurgeo.0060A8F4 |
| 0060A8FB | E9           | LEA ECX,[...]               |
| 0060A8FC |              |                             |
| 0060A8FD |              |                             |
| 0060A8FE |              |                             |
| 0060A903 |              |                             |
| 0060A908 |              |                             |
| 0060A90A |              |                             |
| 0060A90B |              |                             |
| 0060A910 |              |                             |
| 0060A913 |              |                             |
| 0060A916 |              |                             |
| 0060A919 |              |                             |
| 0060A91B |              |                             |
| 0060A920 | 8B45 E8      | MOV EAX,[ELOCAL.6]          |
| 0060A923 | 8D55 EC      | LEA EDX,[ELOCAL.5]          |
| 0060A926 | E8 8901E0FF  | CALL pcsurgeo.0040AA84      |
| 0060A92B | 8D45 EC      | LEA EAX,[ELOCAL.5]          |

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. -->

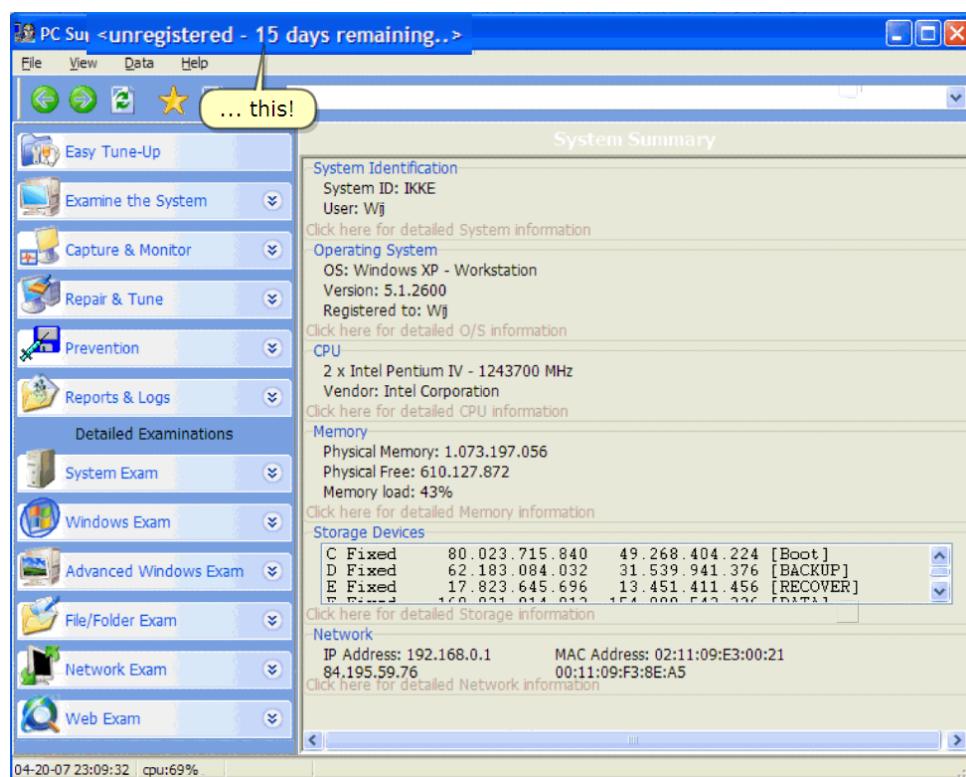
미등록이거나 등록 !!!

(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.

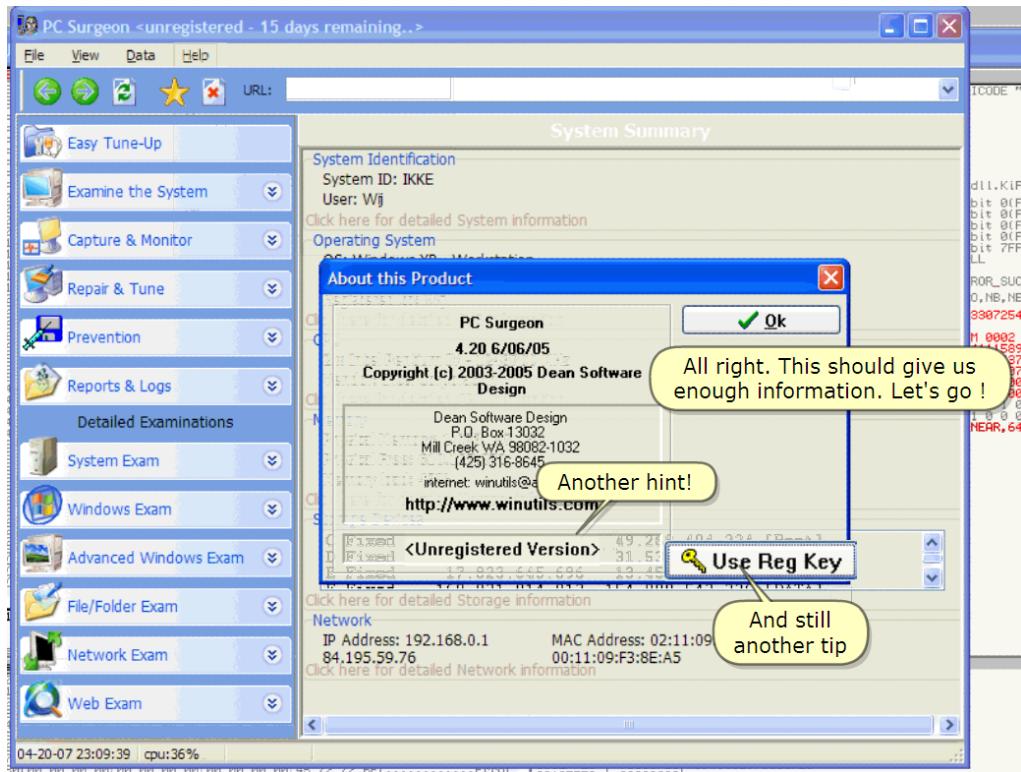


... this !

이곳

Let's find some more info

좀 더 많은 정보를 찾아보자.



Another hint

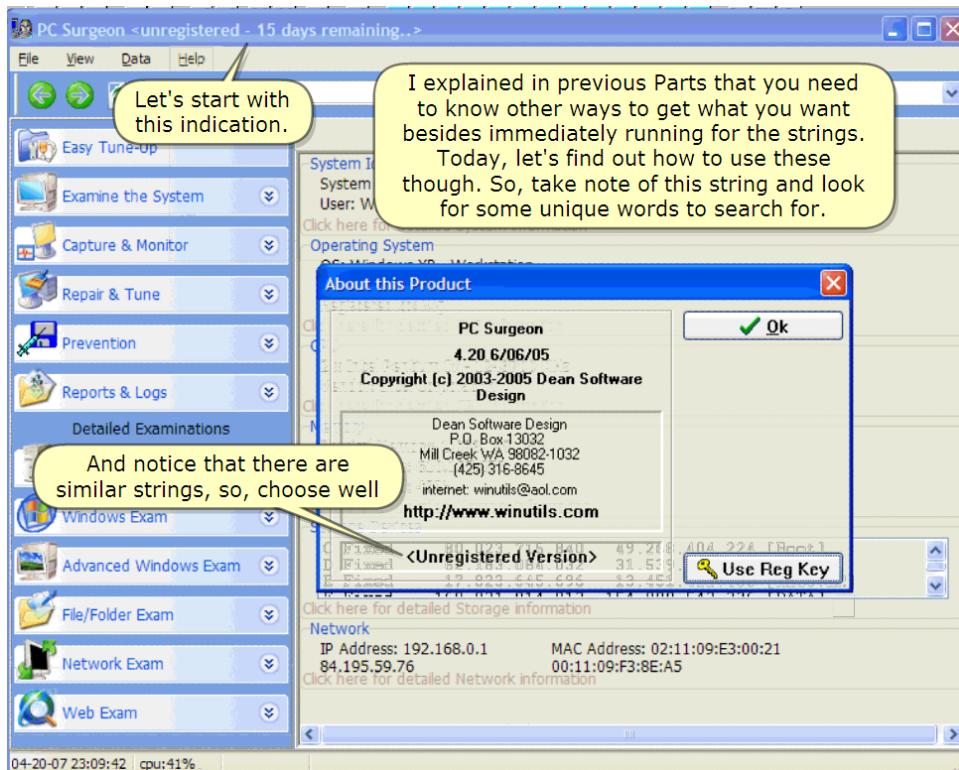
다른 hint

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 part에서 설명했다. 다른 방법을 알고 string을 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 string을 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 string들을 적자. 그래서 잘 선택해야 된다.

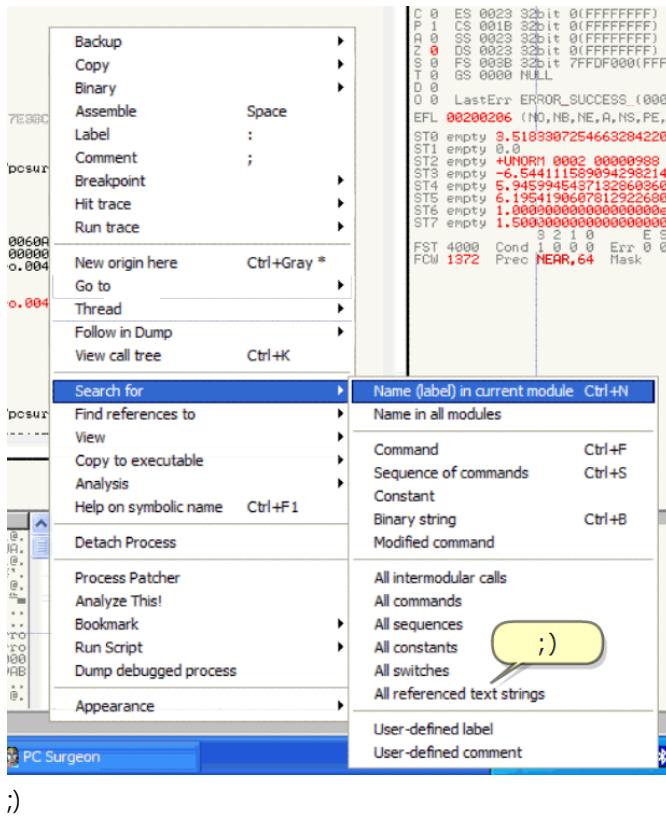
"about" box를 닫고 Olly로 돌아가자

Olly로 돌아가자.

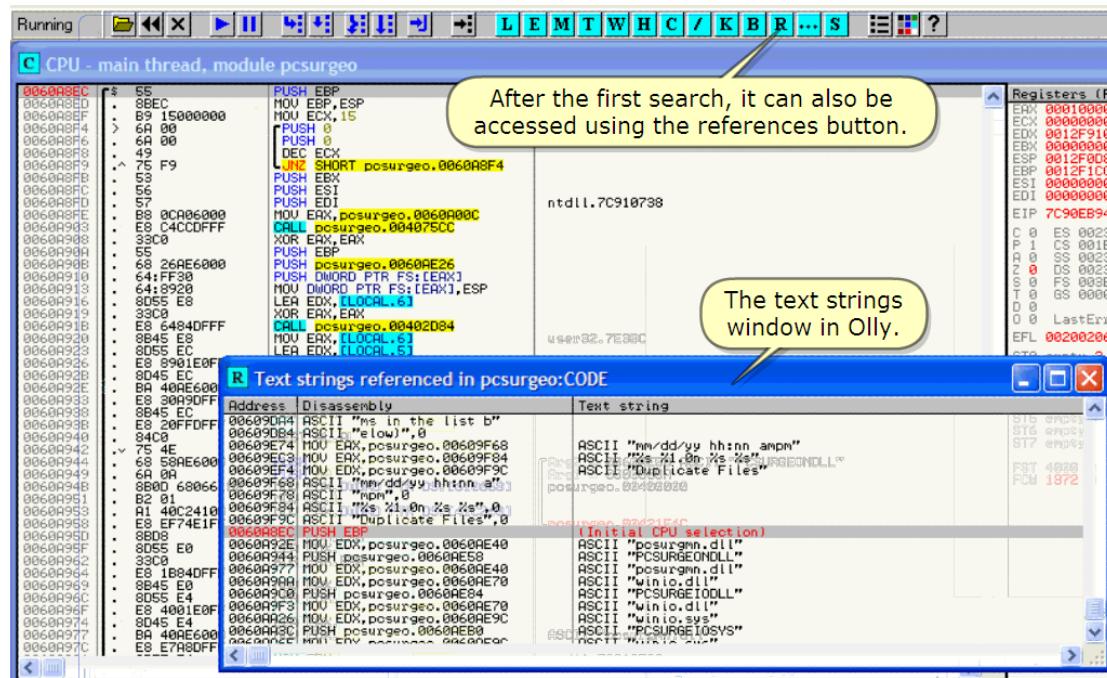
Rightclick in Cpu window to bring up the pop-up menu

CPU window에서 Rightclick 해. POP-up menu 가 뜰 거야.

#### 4. Finding the patches



;)



The text strings window in Olly.

Olly에서 Text strings window 다.

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

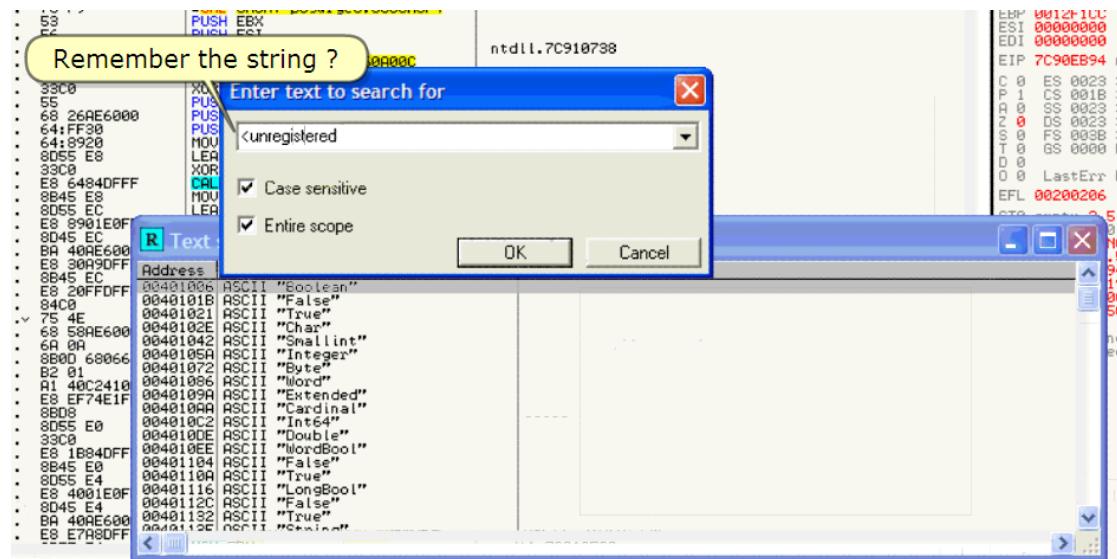
먼저 찾자, 이것 또한 참조 button 을 사용하여 access 할 수 있다.

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 을 끌어. (Olly 는 top에서 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.

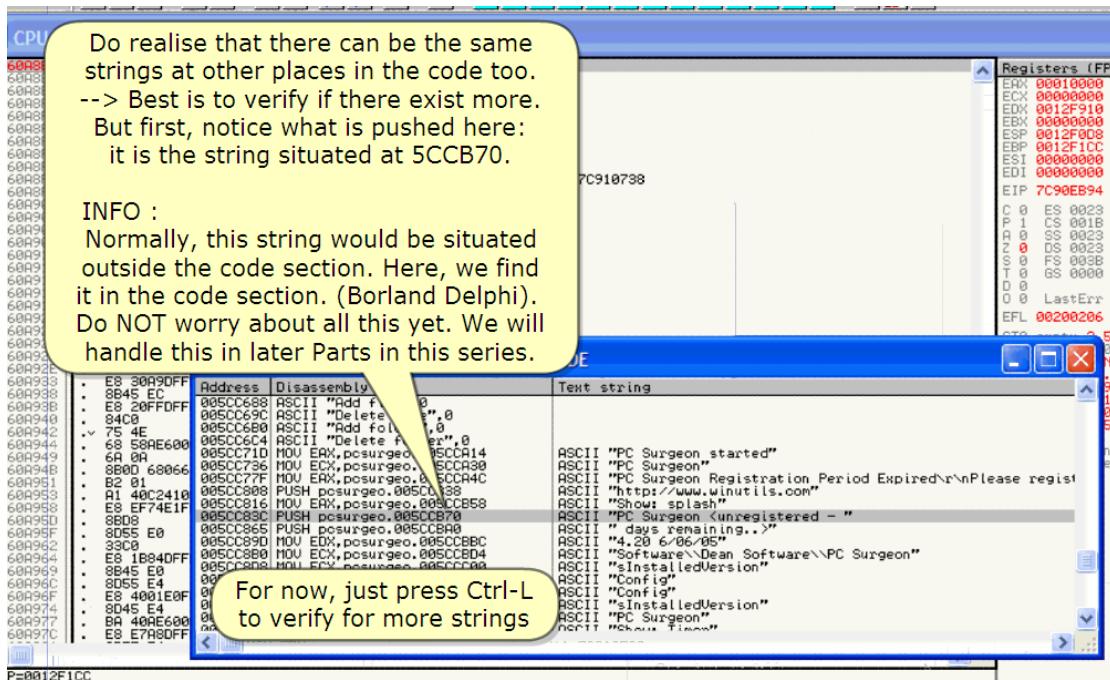
전체 범위에서 대소문자 구별없이 찾는게 가장 좋다. 우리는 어떤 것도 실수하지 않는다.

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 box에 check 하고 실행한다.

Found!

찾았다!



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

깨달아라. Code에서 같은 string은 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 string은 5CCB70에 위치해 있다.

**INFO :**

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

보통, string은 code section 바깥에 위치해 있다. 여기, 우리는 code section에서 찾았다. (Borland Delphi)

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

아직 걱정하지마. 우리는 이 series의 나중 Part에서 조종할 수 있다.

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

이제, Ctrl-L을 눌러 좀 더 많은 string을 검증하자.

Immunity Debugger Screenshot showing search results for the text string "There are none".

| Address  | Disassembly                | Text string   |
|----------|----------------------------|---|
| 005CCB6C | ASCII "Delete file type"   | 0012FFC4 7C816FD7 RETURN to kernel32.7C816FD7<br>ntdll.7C916738 |
| 005CC66C | ASCII 0                    | 0012FFC8 FFFFFFFF   |
| 005CC678 | ASCII "enabled",0          | 0012FFD0 FFFF0000   |
| 005CC688 | ASCII "Add file",0         | 0012FFD4 005446ED   |
| 005CC698 | ASCII "Delete file",0      | 0012FFD8 005446E0   |
| 005CC6B0 | ASCII "Add folder",0       | 0012FFDC 82959EC8   |
| 005CC6C4 | ASCII "Delete folder",0    | 0012FFE0 FFFFFFFF   |
| 005CC710 | MOV EAX, posurgeo.005CCA14 | 0012FFE4 End of SEH chain                                       |
| 005CC736 | MOV ECX, posurgeo.005CCA30 | 0012FFE8 ZCB895FF SE handle                                     |
| 005CC77F | MOV EAX, posurgeo.005CCA4C | 0012FFEB Kernel32.7C816FE8                                      |
| 005CC808 | PUSH posurgeo.005CCB38     | 0012FFEC 00000000   |
| 005CC816 | MOV EAX, posurgeo.005CCB58 | 0012FFE8 00000000   |
| 005CC83C | PUSH posurgeo.005CCB70     | 0012FFFC posurgeo.<ModuleEntryPoint>                            |
| 005CC865 | PUSH posurgeo.005CCBA0     | 0012FFFC 00000000   |
| 005CC89D | MOV EDX, posurgeo.005CCBBC | 0012FFFC 00000000   |
| 005CC8B0 | MOV ECX, posurgeo.005CCBD4 | 0012FFFC 00000000   |
| 005CC8B8 | MOV ECX, posurgeo.005CCC08 | 0012FFFC 00000000   |
| 005CC8D0 | MOV EDX, posurgeo.005CCC1C | 0012FFFC 00000000   |
| 005CC8E4 | Movl ENV posurgeo.005CCB1C | 0012FFFC 00000000   |

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.

Immunity Debugger Screenshot showing search results for the text string "There are none".

| Address  | Disassembly                | Text string   |
|----------|----------------------------|---|
| 005CC65C | ASCII "Delete file type"   | 0012FFC4 7C816FD7 RETURN to kernel32.7C816FD7<br>ntdll.7C916738 |
| 005CC66C | ASCII 0                    | 0012FFC8 FFFFFFFF   |
| 005CC678 | ASCII "enabled",0          | 0012FFD0 FFFF0000   |
| 005CC688 | ASCII "Add file",0         | 0012FFD4 005446ED   |
| 005CC698 | ASCII "Delete file",0      | 0012FFD8 005446E0   |
| 005CC6B0 | ASCII "Add folder",0       | 0012FFDC 82959EC8   |
| 005CC6C4 | ASCII "Delete folder",0    | 0012FFE0 FFFFFFFF   |
| 005CC710 | MOV EAX, posurgeo.005CCA14 | 0012FFE4 End of SEH chain                                       |
| 005CC736 | MOV ECX, posurgeo.005CCA30 | 0012FFEB Kernel32.7C816FE8                                      |
| 005CC77F | MOV EAX, posurgeo.005CCA4C | 0012FFEC 00000000   |
| 005CC808 | PUSH posurgeo.005CCB38     | 0012FFFC 00000000   |
| 005CC816 | MOV EAX, posurgeo.005CCB58 | 0012FFFC 00000000   |
| 005CC83C | PUSH posurgeo.005CCB70     | 0012FFFC 00000000   |
| 005CC865 | PUSH posurgeo.005CCBA0     | 0012FFFC 00000000   |
| 005CC89D | MOV EDX, posurgeo.005CCBBC | 0012FFFC 00000000   |
| 005CC8B0 | MOV ECX, posurgeo.005CCBD4 | 0012FFFC 00000000   |
| 005CC8B8 | MOV ECX, posurgeo.005CCC08 | 0012FFFC 00000000   |
| 005CC8D0 | MOV EDX, posurgeo.005CCC1C | 0012FFFC 00000000   |
| 005CC8E4 | Movl ENV posurgeo.005CCB1C | 0012FFFC 00000000   |

Follow in the code by doubleclicking the line

Code에서 이 line을 doubleclick 하여 따라가자.

C CPU - main thread, module pcsurgeo

```

005CC7E6 58 FBC75C00 PUSH posurgeo.005CC7FB
005CC7EB 8B45 FB MOU EDX, DWORD PTR SS:[EBP-10]
005CC7EE 8B 7D76E9FF CALL posurgeo.00404070
005CC7F3 83 C4 01 RETN
005CC7F4 .^ EB 0080E3FF JMP posurgeo.00404094
005CC7F9 .^ 75 14 JMP SHORT posurgeo.005CC7EB
005CC7FB .^ 75 14 JNZ SHORT posurgeo.005CC816
005CC800 .^ 6A 01 PUSH 1
005CC802 .^ 6A 00 PUSH 0
005CC804 .^ 6A 00 PUSH 0
005CC806 .^ 6A 00 PUSH 0
005CC808 .^ 6A 00 PUSH posurgeo.005CC888
005CC80A .^ 6A 00 PUSH 0
005CC80C .^ 6A 00 PUSH 0
005CC80E .^ 6A 00 CALL K JMP-&shell32.ShellExecuteA
005CC810 > B8 5805C00 MOU EAX, posurgeo.005CCB58
005CC812 .^ EB 905FEDFF CALL posurgeo.0040A278C_FSP
005CC816 .^ A1 9CE60000 MOU EAX, DWORD PTR DS:[60EB8C]
005CC818 .^ 8B38 00 CMP BYTE PTR DS:[EAX], 0
005CC820 .^ 74 00 JNE posurgeo.005CC935
005CC822 .^ A1 DCE16000 MOU EAX, DWORD PTR DS:[60F1DC]
005CC824 .^ 8B38 00 CMP BYTE PTR DS:[EAX], 0
005CC826 .^ 74 00 JNE posurgeo.005CC935
005CC828 .^ EB 70C5C00 PUSH posurgeo.005CCB70
005CC830 .^ D90513CCAC00 FLD DWORD PTR DS:[5CC43C]
005CC832 .^ A1 8EC6000 MOU EAX, DWORD PTR DS:[60EC8C]
005CC834 .^ D020 EC FSUB QWORD PTR DS:[EAX]
005CC836 .^ 83C4 1F40FF ADD ESP,-10H
005CC838 .^ DB3C24 FSTP TBVTE PTR SS:[ESP]
005CC83A .^ 9B 4E MAINT
005CC83C .^ 8D55 E4409 LER EDV, DWORD PTR SS:[EBP-10]
005CC83E .^ B8 94C5C00 MOU EAX, posurgeo.005CCB94
005CC840 .^ EB82F4E3FF CALL posurgeo.0040C2A4
005CC842 .^ FF75 14 PUSH posurgeo.005CCB94
005CC844 .^ 8B 40C5C00 LER EDX, DWORD PTR SS:[EBP-10]
005CC846 .^ B8 D3000000 MOU EDX, 0
005CC848 .^ EB 958E3FF CALL posurgeo.00405326
005CC84A .^ 8B55 E3 HOU EDX, DWORD PTR SS:[EBP-10]
005CC84C .^ BB45 FC10FD MOU EAX, DWORD PTR DS:[EBP-41]
005CC84E .^ EB 222FEAF CALL posurgeo.00473DA
005CC850 .^ 8B45 20120000 MOU EDX, DWORD PTR DS:[EBP+1220]
005CC852 .^ B8 20030000 01 MOU EDX, DWORD PTR DS:[EBP+2003]
005CC854 .^ B810 00000001 MOU EDX, DWORD PTR DS:[EBP+1]
005CC856 .^ EB92 EC000000 CALL DWORD PTR DS:[EDX+EC]

```

IsShown = 1  
BefDir=&HUBS  
Parameters = NULL  
FileName = "http://www.winutils.com"  
Operation = NULL  
hInfd = NULL  
ShellExecuteA  
ASCII "Show: splash"

Ok. We land at the right place.

ASCII "PC Surgeon Unregistered - "

posurgeo.0040C2A4  
ntdll.7C910738  
ASCII "days remaining: 0"  
ASCII "PC Surgeon Registration Period Expired\n" |  
ASCII "http://www.winutils.com"\r  
ASCII "Show splash?"

ASCII " days remaining: 0"\r  
ASCII "28 6/05/25"\r  
ASCII "posurgeo\\Dean Software\\VPC Surgeon"\r  
ASCII "InstalledVersion"\r  
ASCII "Config"\r  
ASCII " " |

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.)

**C CPU - main thread, module pcsurgeo**

```

005CC990 > FF91 CC000000 00 CALL DWORD PTR DS:[ECX+CC]
005CC996 803D 495F6300 00 CMP BYTE PTR DS:[635F49],0
005CC99D .> 74 13 JE SHORT pcsurgeo.005CC9B2
005CC99F . 8B45 FC MOU EAX,DWORD PTR SS:[EBP-4]
005CC9A0 . 8B00 1C100000 MOU EAX,DWORD PTR DS:[EAX+101C]
005CC9A1 . B2 01 MOU DL,1
005CC9A2 . 8B08 MOU ECX,DWORD PTR DS:[ECX]
005CC9A3 . FF91 CC000000 CALL DWORD PTR DS:[ECX+CC]
005CC9A4 . 8B45 FC CMP BYTE PTR DS:[635F4F],0
005CC9A5 .> 74 13 JE SHORT pcsurgeo.005CC9CE
005CC9A6 . 8B45 FC MOU EAX,DWORD PTR SS:[EBP-4]
005CC9A7 . 8B00 20100000 MOU EAX,DWORD PTR DS:[EAX+1020]
005CC9A8 . B2 01 MOU DL,1
005CC9A9 . 8B08 MOU ECX,DWORD PTR DS:[ECX]
005CC9A98 . FF91 CC000000 CALL DWORD PTR DS:[ECX+CC]
005CC9A9C . 8B45 FC MOU EAX,DWORD PTR SS:[EBP-4]
005CC9A9D . 8B00 20000000 MOU EAX,DWORD PTR DS:[EAX+B20]
005CC9A9E . B2 01 MOU DL,1
005CC9A9F . E8 AE1CE7FF CALL pcsurgeo.0043E68C
005CC9B0 . 39C0 XOR EAX,EAX
005CC9B1 . 5A POP EDX
005CC9B2 . 59 POP ECX
005CC9B3 . 64:8910 MOU DWORD PTR FS:[EAX],EDX
005CC9B4 . 68 08CA5C00 PUSH pcsurgeo.005CCA88
005CC9B5 .> 8045 E0 LEA EAX,DWORD PTR SS:[EBP-28]
005CC9B6 . B8 00000000 MOU EDX,3
005CC9B7 . E8 0085E3FF CALL pcsurgeo.00404F88
005CC9B8 . 8B45 F4 LEA EAX,DWORD PTR SS:[EBP-C]
005CC9B9 . E8 0485E3FF CALL pcsurgeo.00404F84
005CC9B90 RETN
005CC9B91 . 00
005CC9B92 .> E9 FE70E3FF JNP pcsurgeo.00404804
005CC9B93 . EB E3 JMP SHORT pcsurgeo.005CC9EB
005CC9B94 . 8BES MOU ESP,EBP
005CC9B95 . SD POP EBP
005CC9B96 . C3 RETN
005CC9B97 . FFFFFFFF DD FFFFFFFF
005CC9B98 . 000000012 DD 00000012
005CC9B99 . 12000000 ASCII "PC Surgeon start"
005CC9B9A . 50 43 20 53 75 7 ASCII "ed",0
005CC9B9B . 65 64 00 DB 00
005CC9B9C . 00 DD FFFFFFFF
005CC9B9D . FFFFFFFF DD 00000000
005CC9B9E . 00000000 DD 00
005CC9B9F . 50 43 20 53 75 7 ASCII "PC Surgeon",0
005CC9B9A0 . 00 DB 00
005CC9B9A1 . 00007041 DD FLOAT 15.00000

```

We arrive at the end of the routine.  
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.

**C CPU - main thread, module pcsurgeo**

```

005CC722 > E8 715EE0FF CALL pcsurgeo.004A2598
005CC727 . 39C9 XOR ECX,ECX
005CC729 . B8 01000000 MOU EDX,1
005CC72E . 8B45 FC MOU EAX,DWORD PTR SS:[EBP-4]
005CC731 . E8 4235FFFF CALL pcsurgeo.005BCF78
005CC736 . B9 30CH5C00 MOU ECX,pcsurgeo.005CCA80
005CC738 . B8 03000000 MOU EDX,8
005CC739 . 8B45 FC MOU EAX,DWORD PTR SS:[EBP-4]
005CC740 . E8 3035FFFF CALL pcsurgeo.005BCF78
005CC743 . A1 8CE66000 MOU EAX,DWORD PTR DS:[60EB8C]
005CC744 . 8B88 00 CMP BYTE PTR DS:[EAX],0
005CC745 .> 60 68 C0000000 JE pcsurgeo.005CC816
005CC746 . B1 8CEC6000 MOU EDX,DWORD PTR DS:[60EC0C]
005CC747 . D000 FIDWORD PTR DS:[EAX]
005CC748 . D81D 3CC5C00 FCOMD DWORD PTR DS:[5CC43C]
005CC749 . DE00 FSTSW AX
005CC750 . 9E SAHF
005CC751 .> 0F86 AP000000 JE pcsurgeo.005CC816
005CC752 . A1 DCF16000 MOU EAX,DWORD PTR DS:[60F1DC]
005CC753 . C600 01 MOU BYTE PTR DS:[EAX],1
005CC754 . E9 00 PUSH 0
005CC755 . 66:8B00 40C5C00 MOU CX,WORD PTR DS:[5CC4A0]
005CC756 . B2 01 MOU DL,1
005CC757 . B8 4CC45C00 MOU EAX,pcsurgeo.005CCA4C
005CC758 . E8 07E6E6FF CALL pcsurgeo.0043D090
005CC759 . 66:8945 FA MOU WORD PTR SS:[EBP-6],AX
005CC760 . 66:8970 FA 06 CMP MDR PTR SS:[EBP-6],6
005CC761 .> 75 67 JNZ SHORT pcsurgeo.005CC7FB
005CC762 . 8B00 7CF06000 MOU ECX,DWORD PTR DS:[60F07C]
005CC763 . 8B09 MOU ECX,DWORD PTR DS:[ECX]
005CC764 . B2 01 MOU DL,1
005CC765 . A1 38234A00 MOU EAX,DWORD PTR DS:[4A2338]
005CC766 . E8 1063ECCF CALL pcsurgeo.004930B8
005CC767 . 8945 F0 MOU DMORD PTR SS:[EBP-18],EAX
005CC768 . 33C0 XOR EAX,EAX
005CC769 . 65 PUSH EBP
005CC770 . 68 F4C75C00 PUSH pcsurgeo.005CC7F4
005CC771 . 64:FF30 PUSH DWORD PTR FS:[EAX]
005CC772 . 8B45 F0 MOU EAX,DWORD PTR FS:[EAX],ESP
005CC773 . 8B10 MOU EDX,DWORD PTR DS:[EBP-10]
005CC774 . FF92 EC000000 CALL DWORD PTR DS:[EDX+ECI]
005CC775 . A1 70EC6000 MOU EAX,DWORD PTR DS:[60EC70]
005CC776 . 8B88 00 CMP BYTE PTR DS:[EAX],0
005CC777 .> 74 10 JE SHORT pcsurgeo.005CC7DE
005CC778 . A1 DCF16000 MOU EAX,DWORD PTR DS:[60F1DC]

```

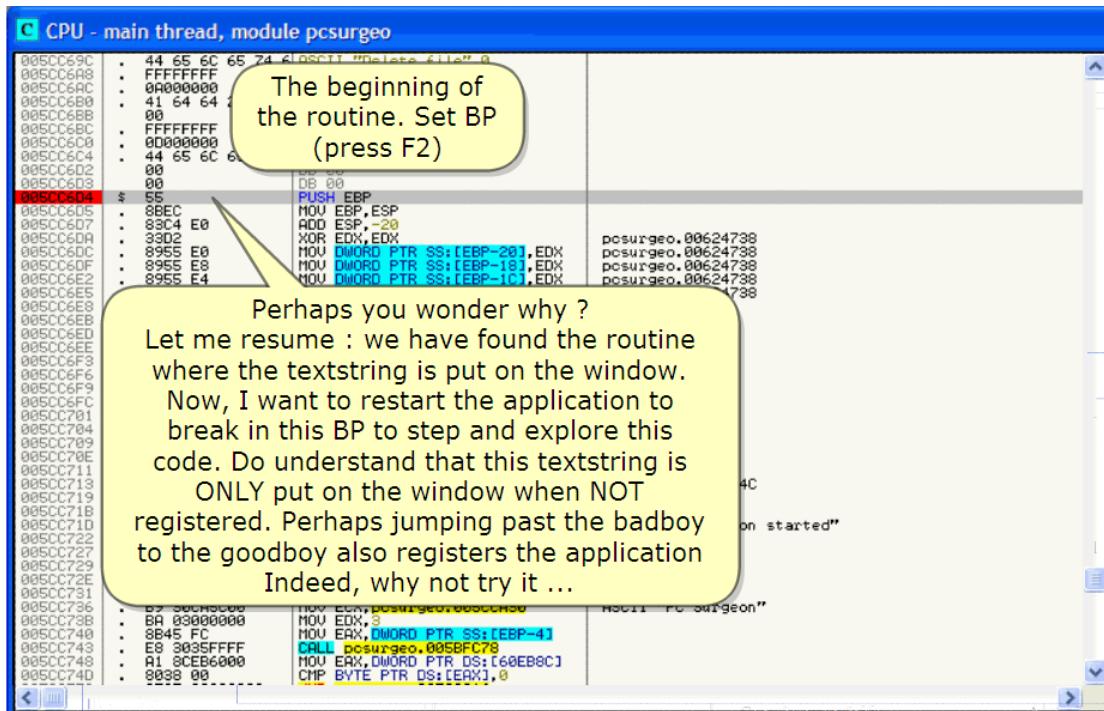
Oh! And we don't want to run into this, right?  
Let's scroll further up.

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 드저히하지 않지?

Restart

제11장

Pun

시행

#### Fine Olly breaks in the BP

좋아. Olly 가 BP 안에서 멈췄다.

Start exploring and trace F8

탐험을 하자. F8 로 추적해.

CPU - main thread, module pcsergeo

| Address  | OpCode        | Instruction                     | Comments                   |
|----------|---------------|---------------------------------|----------------------------|
| 005CC6D4 | 55            | PUSH EBP                        |                            |
| 005CC6D5 | 8BEC          | MOV EBP, ESP                    |                            |
| 005CC6D7 | 83C4 E0       | ADD ESP, -20                    |                            |
| 005CC6D8 | 33D2          | XOR EDX, EDX                    |                            |
| 005CC6D9 | 8955 E0       | MOV DWORD PTR SS:[EBP-20], EDX  | pcsergeo.006105CC          |
| 005CC6DC | 8955 E8       | MOV DWORD PTR SS:[EBP-18], EDX  | pcsergeo.006105CC          |
| 005CC6DF | 8955 E4       | MOV DWORD PTR SS:[EBP-1C], EDX  | pcsergeo.006105CC          |
| 005CC6E2 | 8955 F4       | MOV DWORD PTR SS:[EBP-C], EDX   | pcsergeo.006105CC          |
| 005CC6E5 | 8945 FC       | MOV DWORD PTR SS:[EBP-4], EDX   | pcsergeo.00610C48          |
| 005CC6E8 | 33C9          | XOR EAX, EAX                    | pcsergeo.00610C48          |
| 005CC6EB |               |                                 |                            |
| 005CC6ED |               |                                 |                            |
| 005CC6EE |               |                                 |                            |
| 005CC6F3 |               |                                 |                            |
| 005CC6F5 |               |                                 |                            |
| 005CC6F9 |               |                                 |                            |
| 005CC6FC | EB 0F         | JMP SHORT [PC+1]                |                            |
| 005CC701 | 8B45 FC       | MOV EAX, DWORD PTR DS:[EBP-4]   |                            |
| 005CC704 | E8 93C7FFFF   | CALL pcsergeo.005C8E9C          |                            |
| 005CC709 | A1 BCF36000   | MOV EAX, DWORD PTR DS:[60F3BC]  |                            |
| 005CC70F | 8038 00       | CMP BYTE PTR DS:[EAX], 0        |                            |
| 005CC711 | 74 14         | JE SHORT pcsergeo.005CC727      | pcsergeo.00610C4C          |
| 005CC713 | BB15 E0EB6000 | MOV EDX, DWORD PTR DS:[60EB6E8] | ASCII "PC Surgeon started" |
| 005CC719 | BB12          | MOV EDX, DWORD PTR DS:[EDX]     |                            |
| 005CC71B | 33C9          | XOR ECX, ECX                    |                            |
| 005CC71D | B8 14CA5C00   | MOV EAX, 14CA5C00               |                            |
| 005CC722 | E8 715EEDFF   | CALL D [PC+1]                   |                            |
| 005CC727 | 33C9          | XOR ECX, ECX                    |                            |
| 005CC729 | BA 01000000   | MOV EDX, 1                      |                            |
| 005CC72E | BB45 FC       | MOV EAX, DWORD PTR SS:[EBP-4]   |                            |
| 005CC731 | E8 4235FFFF   | CALL pcsergeo.005BFC78          | ASCII "PC Surgeon"         |
| 005CC736 | B9 30CASC00   | MOV ECX, pcsergeo.005CCA90      |                            |
| 005CC73B | BA 03000000   | MOV EDX, 3                      |                            |
| 005CC740 | BB45 FC       | MOV EAX, DWORD PTR SS:[EBP-4]   |                            |
| 005CC743 | E8 2035FFFF   | CALL pcsergeo.005BFC78          |                            |
| 005CC748 | A1 8CEB6000   | MOV EAX, DWORD PTR DS:[60EB8C]  |                            |
| 005CC74D | 8038 00       | CMP BYTE PTR DS:[EAX], 0        |                            |

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.

C CPU - main thread, module pcsurgeo

```

005CC6D4 $ 55 PUSH EBP
005CC6D5 . 8EC MOV EBP,ESP
005CC6D7 . 89C4 E0 ADD ESP,-20
005CC6DA . 33D2 XOR EDX,EDX
005CC6DC . 8955 E0 MOV DWORD PTR DS:[ESP+14],EDX
005CC6DF . 8955 E8 MOV DWORD PTR DS:[ESP+14],EDX
005CC6E2 . 8955 E4 MOV DWORD PTR DS:[ESP+14],EDX
005CC6E5 . 8955 F4 MOV DWORD PTR DS:[ESP+14],EDX
005CC6E8 . 8945 FC MOV DWORD PTR DS:[ESP+14],EDX
005CC6E9 . 5080 XOR EAX,EAX
005CC6ED . 5080 PUSH EBP
005CC6EE . 68 010A5C00 PUSH pcsurgeo
005CC6F3 . 64:FF30 PUSH DWORD PTR DS:[ESP+14]
005CC6F6 . 64:8920 MOV EAX,DWORD PTR DS:[ESP+14]
005CC6F9 . 8845 FC MOV EAX,DWORD PTR DS:[ESP+14]
005CC6FC . E8 0FAFFFFF CALL pcsurgeo
005CC701 . 8845 FC MOV EAX,DWORD PTR DS:[ESP+14]
005CC704 . E8 93C7FFFF CALL pcsurgeo
005CC709 . A1 BCF36000 MOV EAX,DWORD PTR DS:[ESP+14]
005CC70E . 8B33 00 CMP BYTE PTR DS:[ESP+14],0
005CC711 . 74 14 JE SHORT pcsurgeo.005CC712
005CC713 . 8B15 E0EB6000 MOV EDX,DWORD PTR DS:[60EB6000]
005CC714 . 8B15 E0EB6000 MOV EDX,DWORD PTR DS:[60EB6000]
005CC71B . 33C9 XOR ECX,ECX
005CC71D . B8 14CA5C00 MOV EAX,pcsurgeo.005CCA14
005CC722 . E8 715EEDFF CALL pcsurgeo.004A2598
005CC727 . 33C9 XOR ECX,ECX
005CC729 . B8 01000000 MOV EAX,pcsurgeo.005CCA14
005CC72E . 8845 FC MOV EAX,DWORD PTR DS:[ESP+14]
005CC731 . E8 4235FFFF CALL pcsurgeo.005CCA14
005CC736 . B9 30CA5C00 MOV ECX,pcsurgeo.005CCA14
005CC738 . B8 00000000 MOV EDX,3
005CC740 . 8845 FC MOV EAX,DWORD PTR DS:[ESP+14]
005CC743 . E8 3035FFFF CALL pcsurgeo.005CCA14
005CC748 . A1 8CE6000 MOV EAX,DWORD PTR DS:[ESP+14]
005CC74D . 8933 00 CMP EAX,DWORD PTR DS:[ESP+14]
005CC74F . 8B45 C00000000 JNZ pcsurgeo.005CCA14
005CC756 . A1 8CEC6000 MOV EAX,DWORD PTR DS:[ESP+14]
005CC75B . D000 FLD DWORD PTR DS:[ESP+14]
005CC75D . D810 30CA5C00 FCOMC DWORD PTR DS:[ESP+14]
005CC763 . DFE0 FSTSW AX
005CC765 . 9E SAHF
005CC766 . v 0F86 AA000000 JBE pcsurgeo.005CCA14
005CC76C . A1 DCF16000 MOV EAX,DWORD PTR DS:[ESP+14]
005CC771 . C600 01 MOV ECX,DWORD PTR DS:[ESP+14]
005CC774 . 6A 00 PUSH 0

```

Jump is taken  
005CC727=pcsurgeo.005CC727

| Address   | Hex dump   |
|-----------|--|
| 005CB8000 | 00 00 00 00 00 00 00 00 00 00 00 00 58 CB EB D1 02 |
| 005CB8010 | 00 20 41 00 00 00 00 00 00 00 00 00 00 00 00 00    |

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 명청한 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.

Let's find out if just patching these jumps  
(== Plain stupid patching system) also registers the application (or not). 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. This will be subject for and explained in later Parts and very briefly at the end of this Part06.

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

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

**C CPU - main thread, module pcsurgeo**

```

005CC604: 55      PUSH EBP
005CC605: 8BEC    MOU EBP,ESP
005CC607: 89C4 E0 ADD ESP,-20
005CC608: 32D2    XOR EDX,EDX
005CC60E: 8955 E0 MOU DWORD PTR SS:[EBP-20],EDX
005CC60F: 8955 E8 MOU DWORD PTR SS:[EBP-18],EDX
005CC622: 8955 E4 MOU DWORD PTR SS:[EBP-10],EDX
005CC665: 8955 F4 MOU DWORD PTR SS:[EBP-C],EDX
005CC668: 8945 FC MOU DWORD PTR SS:[EBP-4],EDX
005CC66B: 33C0    XOR EAX,EAX
005CC66D: 55      PUSH EBP
005CC66E: 6A 01CA5C00 PUSH pcsurgeo.005CCA01
005CC6F3: 64:FF30 PUSH DWORD PTR FS:[ERX]
005CC6F6: 64:8920 MOU DWORD PTR FS:[ERX],ESP
005CC6F9: 8845 FC MOU EAX,DWORD PTR SS:[EBP-4]
005CC6FC: E8 0F46FFFF CALL pcsurgeo.005C6010
005CC701: 8845 FC MOU EAX,DWORD PTR SS:[EBP-4]
005CC704: E8 93C7FFFF CALL pcsurgeo.005C8E9C
005CC709: A1 8CF36900 MOU EAX,DWORD PTR DS:[60F3BC]
005CC70E: 8038 00 CMP BYTE PTR DS:[ERX],0
005CC711: 74 14 JE SHORT pcsurgeo.005CC727
005CC713: 8B15 E8EB6000 MOU EDX,DWORD PTR DS:[68EBE8]
005CC719: 8B12    MOU EAX,DWORD PTR DS:[ERX]
005CC71B: 33C9    XOR ECX,ECX
005CC71D: E8 14CA5C00 MOU EAX,pcsurgeo.005CCA01
005CC722: E8 715EE0FF CALL pcsurgeo.004A01
005CC725: 33C9    XOR ECX,ECX
005CC729: 6A 01000000 MOU EDX,1
005CC72E: 8845 FC MOU EAX,DWORD PTR DS:[ERX]
005CC731: E8 4235FFFF CALL pcsurgeo.005BF000
005CC736: 80 30CA5C00 MOU ECX,pcsurgeo.005C8E9C
005CC73B: 6A 03000000 MOU EDX,3
005CC740: 8845 FC MOU EAX,DWORD PTR SS:[EBP-4]
005CC743: E8 3035FFFF CALL pcsurgeo.005BFC78
005CC748: A1 8CEB6000 MOU EAX,DWORD PTR DS:[68E800]
005CC74D: 8038 00 CMP BYTE PTR DS:[ERX],0
005CC750: 74 85 C0000000 JNZ pcsurgeo.005C8816
005CC756: A1 8CEC6000 MOU EAX,DWORD PTR DS:[60EC8C]
005CC768: D000    FLD QWORD PTR DS:[EAX]
005CC76D: D810 3CCA5C00 FCOMQ DWORD PTR DS:[SCCA3C]
005CC763: DFE0    FSTSW AX
005CC765: 9C     SAHF
005CC766: 74 86 AA000000 JBE pcsurgeo.005CC816
005CC76C: A1 DCF16000 MOU EAX,DWORD PTR DS:[60F1DC]
005CC771: C600 01 MOU BYTE PTR DS:[ERX],1
005CC774: 6A 00 PUSH 0
005CC776: 66:8800 40CA5C00 MOU CX,WORD PTR DS:[SCCA40]
005CC77D: B2 01 MOU DL,1
005CC77F: 40CA5C00 MOU EAX,pcsurgeo.005CCA40
005CC784: E8 07E6E6FF CALL pcsurgeo.0043D908
005CC789: 66:6635 FA MOU EAX,DWORD PTR DS:[ERX]
005CC790: 66:8370 FA 06 MOU EDX,DWORD PTR DS:[ERX-1],EDX
005CC792: 95 67 JNZ SHORT pcsurgeo.005C27E8
005CC794: 8B00 7CF06000 MOU ECX,DWORD PTR DS:[60F07C]
005CC794: 8B09    MOU ECX,DWORD PTR DS:[ECX]
005CC796: B2 01 MOU DL,1
005CC79E: A1 38234A00 MOU EAX,DWORD PTR DS:[4A2338]
005CC798: E8 1069ECFF CALL pcsurgeo.004990B8
005CC7A8: 8845 F0 XOR EAX,ERX
005CC7B8: 33C0    PUSH EBP
005CC7D0: 55      PUSH EBP
005CC7E8: 68 F4C75C00 PUSH DWORD PTR .005CC7F4
005CC7B3: 64:FF30 PUSH DWORD PTR FS:[ERX]
005CC7B6: 64:8920 MOU DWORD PTR FS:[ERX],ESP
005CC7B9: 8845 F0 MOU EDX,DWORD PTR DS:[ERX]
005CC7B8: 8B10    CALL DWORD PTR DS:[EDX+4]
005CC7B8: FF92 EC000000 MOU EAX,DWORD PTR DS:[60ECE70]
005CC7B4: A1 8CEC6000 CMP BYTE PTR DS:[ERX],0
005CC7B9: 8038 00 JE SHORT pcsurgeo.005CC70E
005CC7C0: 74 10

```

Again a similar pointer that will decide about jumping or not.  
Remark that it's not the same.

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.

```

005CC748: A1 8CEB6000 MOU EAX,DWORD PTR DS:[60EB8C]
005CC74D: 8038 00 CMP BYTE PTR DS:[ERX],0
005CC750: 74 85 C0000000 JNZ pcsurgeo.005C8816
005CC756: A1 8CEC6000 MOU EAX,DWORD PTR DS:[60EC8C]
005CC75B: D000    FLD QWORD PTR DS:[EAX]
005CC75D: D810 3CCA5C00 FCOMQ DWORD PTR DS:[SCCA3C]
005CC763: DFE0    FSTSW AX
005CC765: 9C     SAHF
005CC766: 74 86 AA000000 JBE pcsurgeo.005CC816
005CC76C: A1 DCF16000 MOU EAX,DWORD PTR DS:[60F1DC]
005CC771: C600 01 MOU BYTE PTR DS:[ERX],1
005CC774: 6A 00 PUSH 0
005CC776: 66:8800 40CA5C00 MOU CX,WORD PTR DS:[SCCA40]
005CC77D: B2 01 MOU DL,1
005CC77F: 40CA5C00 MOU EAX,pcsurgeo.005CCA40
005CC784: E8 07E6E6FF CALL pcsurgeo.0043D908
005CC789: 66:6635 FA MOU EAX,DWORD PTR DS:[ERX]
005CC790: 66:8370 FA 06 MOU EDX,DWORD PTR DS:[ERX-1],EDX
005CC792: 95 67 JNZ SHORT pcsurgeo.005C27E8
005CC794: 8B00 7CF06000 MOU ECX,DWORD PTR DS:[60F07C]
005CC794: 8B09    MOU ECX,DWORD PTR DS:[ECX]
005CC796: B2 01 MOU DL,1
005CC79E: A1 38234A00 MOU EAX,DWORD PTR DS:[4A2338]
005CC798: E8 1069ECFF CALL pcsurgeo.004990B8
005CC7A8: 8845 F0 XOR EAX,ERX
005CC7B8: 33C0    PUSH EBP
005CC7D0: 55      PUSH EBP
005CC7E8: 68 F4C75C00 PUSH DWORD PTR .005CC7F4
005CC7B3: 64:FF30 PUSH DWORD PTR FS:[ERX]
005CC7B6: 64:8920 MOU DWORD PTR FS:[ERX],ESP
005CC7B9: 8845 F0 MOU EDX,DWORD PTR DS:[ERX]
005CC7B8: 8B10    CALL DWORD PTR DS:[EDX+4]
005CC7B8: FF92 EC000000 MOU EAX,DWORD PTR DS:[60ECE70]
005CC7B4: A1 8CEC6000 CMP BYTE PTR DS:[ERX],0
005CC7B9: 8038 00 JE SHORT pcsurgeo.005CC70E
005CC7C0: 74 10

```

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.

**C CPU - main thread, module posurgeo**

```

005C736 . B9 30CA5C00 MOV ECX, posurgeo.005CCA80
005C738 . BA 03000000 MOV EDX, 3
005C740 . 8B45 FC MOV EDX, DWORD PTR SS:[EBP-4]
005C743 . E8 303FFFFF CALL posurgeo.005BFC78
005C745 . A1 8CEB6000 MOV EDX, DWORD PTR DS:[60EB8C]
005C74D . 8039 00 CMP BYTE PTR DS:[EAX], 0
005C753 . 74 00 JE posurgeo.005CC081A
005C755 . A1 8CEC6000 MOV EDX, DWORD PTR DS:[60EC8C]
005C758 . D049 FLD QWORD PTR DS:[EAX]
005C75D . 0810 0CC45C00 FCOMP QWORD PTR DS:[60C48C]
005C763 . DFE0 FSTSW AX
005C765 . 9E SAHF
005C766 . A8E8 00000000 JNE posurgeo.005CC081A
005C76C . A1 DCF16000 MOV EDX, DWORD PTR DS:[60F1DC]
005C771 . C600 01
005C774 . 6A 00
005C776 . 6618880 40CA51
005C77D . B2 01
005C77F . B8 40CA5C00
005C784 . E8 07E6EFFF
005C789 . 6618945 FA
005C78D . 661897D FA 06
005C792 . 75 67 JNE SHORT posurgeo.005CC7FB
005C794 . 8800 00000000 MOV ECX, DWORD PTR DS:[60F07C]
005C798 . 8809
005C799 . B2 02
005C79E . A1 03
005C7A3 . E8 10
005C7A5 . 8945
005C7A8 . 3008
005C7A9 . 65
005C7A9 . 68 F
005C7B3 . 64:FB
005C7B6 . 64:89
005C7B9 . 8845 F0
005C7BC . 8810
005C7BE . FF92 EC000000
005C7C4 . A1 70EC6000
005C7C9 . 8088 00
005C7CE . 74 10 JE SHORT posurgeo.005CC7DE
005C7CE . A1 DCF16000
005C7D3 . C600 00
005C7D6 . A1 8CEB6000
005C7DB . C600 01
005C7DE . > 39C0 XOR EDX, EAX
005C7DE . 5H POP EDX

```

ASCII "PC Surgeon"

However, we always want to jump here or else we may run into expired at a later date

Hence, we need to remember to patch one of these cond jumps to always jump. For now, set BP

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!

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.

**C CPU - main thread, module pcsurgeo**

```

005CC800 .~ 75 14 JNZ SHORT pcsurgeo.005CC816
005CC802 .~ 6A 01 PUSH 1
005CC804 .~ 6A 00 PUSH 0
005CC806 .~ 6A 00 PUSH 0
005CC808 .~ 68 38CB5C00 PUSH pcsurgeo.005CCB38
005CC80D .~ 6A 00 PUSH 0
005CC80F .~ 6A 00 PUSH 0
005CC811 .~ E8 1289E6FF CALL <JMP.&shell32.ShellExecuteA>
005CC816 .> B8 58CB5C00 MOU EAX,pcsurgeo.005CCB58
005CC818 .~ E8 9C5FE0FF CALL pcsurgeo.00442780
005CC820 .~ A1 8CCE6000 MOU EAX,DUORD PTR DS:[60EB8C]
005CC825 .~ 8038 00 CMP BYTE PTR DS:[EAX],0
005CC828 .~ 0F85 07010000 JNZ posurgeo.005CC935
005CC82E .~ A1 DCF16000 MOU EAX,DWORD PTR DS:[60F1DC]
005CC833 .~ 8038 00 CMP BYTE PTR DS:[EAX],0
005CC836 .~ 0F85 F9000000 JNZ posurgeo.005CC935
005CC83C .~ 68 70CB5C00 PUSH pcsurgeo.005CCB70
005CC841 .~ D905 3CCAC00 FLD DWORD PTR DS:[5CCA9C]
005CC847 .~ A1 8CCE6000 MOV EAX,DWORD PTR DS:[60EC8C]
005CC84C DC20 FSUB QWORD PTR DS:[EAX]
005CC84E .~ 88C4 F4 ADD ESP,-8C
005CC851 .~ DB3C24 FSTP TBYTE PTR SS:[ESP]
005CC854 .~ 98 HLT

```

IsShown = 1  
DefDir = NULL  
Parameters = NULL  
FileName = "http://www.winutils.com"  
Operation = NULL  
hInfd = NULL  
ShellExecuteA  
ASCII "Show: splash"

Remark that the same pointer is deciding again!

Remark that the same pointer is deciding again!

주목. 같은 pointer 가 다시 결정한다.

**C CPU - main thread, module pcsurgeo**

```

005CC800 .~ 75 14 JNZ SHORT pcsurgeo.005CC816
005CC802 .~ 6A 01 PUSH 1
005CC804 .~ 6A 00 PUSH 0
005CC806 .~ 6A 00 PUSH 0
005CC808 .~ 68 38CB5C00 PUSH pcsurgeo.005CCB38
005CC80D .~ 6A 00 PUSH 0
005CC80F .~ 6A 00 PUSH 0
005CC811 .~ E8 1289E6FF CALL <JMP.&shell32.ShellExecuteA>
005CC816 .~ B8 58CB5C00 MOU EAX,pcsurgeo.005CCB58
005CC818 .~ E8 9C5FE0FF CALL pcsurgeo.00442780
005CC820 .~ A1 8CCE6000 MOU EAX,DUORD PTR DS:[60EB8C]
005CC825 .~ 8038 00 CMP BYTE PTR DS:[EAX],0
005CC828 .~ 0F85 07010000 JNZ posurgeo.005CC935
005CC82E .~ A1 DCF16000 MOU EAX,DWORD PTR DS:[60F1DC]
005CC833 .~ 8038 00 CMP BYTE PTR DS:[EAX],0
005CC836 .~ 0F85 F9000000 JNZ posurgeo.005CC935
005CC83C .~ 68 70CB5C00 PUSH pcsurgeo.005CCB70
005CC841 .~ D905 3CCAC00 FLD DWORD PTR DS:[5CCA9C]
005CC847 .~ A1 8CCE6000 MOV EAX,DWORD PTR DS:[60EC8C]
005CC84C DC20 FSUB QWORD PTR DS:[EAX]
005CC851 .~ 88C4 F4 ADD ESP,-8C
005CC854 .~ DB3C24 FSTP TBYTE PTR SS:[ESP]
005CC854 .~ 9B WAIT
005CC855 .~ 8055 E4 LEA EDX,DWORD PTR SS:[EBP-1C]
005CC858 .~ B8 94CB5C00 MOU EAX,pcsurgeo.005CCB94
005CC859 .~ E8 42FAE3FF CALL pcsurgeo.0040C2A4
005CC862 .~ FF75 E4 PUSH DWORD PTR SS:[EBP-1C]
005CC865 .~ 68 A0CB5C00 PUSH pcsurgeo.005CCB90
005CC869 .~ 8D45 E8 LEA EAX,DWORD PTR SS:[EBP-18]
005CC86D .~ B8 03000000 MOU EDX,3
005CC872 .~ E8 A98AE3FF CALL pcsurgeo.00405320
005CC877 .~ 8B55 E8 MOU EDX,DWORD PTR SS:[EBP-18]
005CC879 .~ 6A45 E8 MOU EAX,QWORD PTR SS:[EBP-18]
005CC87D .~ 00 DS:[00610C4A]=00
005CC882 .~ 00
005CC885 .~ 00
005CC888 .~ 00
005CC892 .~ 00
005CC894 .~ 00
005CC899 .~ 00
005CC89D .~ 00
005CC89A .~ 00
005CC8A2 .~ 00
005CC8A4 .~ 00
005CC8A6 .~ 00
005CC8B0 .~ 00

```

ASCII "PC Surgeon <unregister>"

INFO:  
I'll come back to this later in more detail but remark already that the deciding pointer's value [60EB8C] is 00610C4A. But [00610C4] is zero though, hence, we will not jump.

pcsurgeo.0040C2A4  
ASCII " days remaining..?"  
5/05"

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 00610C4A이다.

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

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

Step F8

F8 눌러

C CPU - main thread, module pcsurgeo

```

005CC800 .~ 75 14 JNZ SHORT posurgeo.005CC816
005CC802 . 6A 01 PUSH 1
005CC804 . 6A 00 PUSH 0
005CC806 . 6A 00 PUSH 0
005CC808 . 68 38C85C00 PUSH posurgeo.005CCB38
005CC80D . 6A 00 PUSH 0
005CC80F . 6A 00 PUSH 0
005CC811 . E8 12B9E6FF CALL <JMP.&shell32.ShellExecuteA>
005CC816 > B8 58C85C00 MOV EAX, posurgeo.005CCB58
005CC818 E8 905FEDFF CALL posurgeo.004A27BC
005CC820 . A1 8CE65000 MOU EAX, DWORD PTR DS:[60EB8C]
005CC825 . 8038 00 CMP BYTE PTR DS:[EAX], 0
005CC828 .~ 0F85 07010000 JNZ posurgeo.005CC935
005CC82E . A1 DCF16000 MOV EAX, DWORD PTR DS:[60F1DC]
005CC833 . 8038 00 CMP BYTE PTR DS:[EAX], 0
005CC836 .~ 0F85 F9000000 JNZ posurgeo.005CC935
005CC83C . 68 70C85C00 PUSH posurgeo.005CCB70
005CC841 D905 3C0A5C00
005CC847 A1 8CEC6000
005CC84C DC20
005CC84E 83C4 F4
005CC851 DB3C24
005CC854 9B
005CC855 BD5E E4
005CC858 B8 94C85C00
005CC85D E8 42FAE3FF CALL posurgeo.0040C2A4
005CC862 FF75 E4 PUSH DWORD PTR SS:[EBP-1C]
005CC865 . 68 A0C85C00 PUSH posurgeo.005CCBA0
005CC86A . 8045 E8 LEA EAX, DWORD PTR SS:[EBP-18]
005CC86D . BA 03000000 MOU EDX, 3
005CC872 . E8 A98AE3FF CALL posurgeo.00405320
005CC877 . 8B55 E8 MOV EDX, DWORD PTR SS:[EBP-18]

```

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.

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.

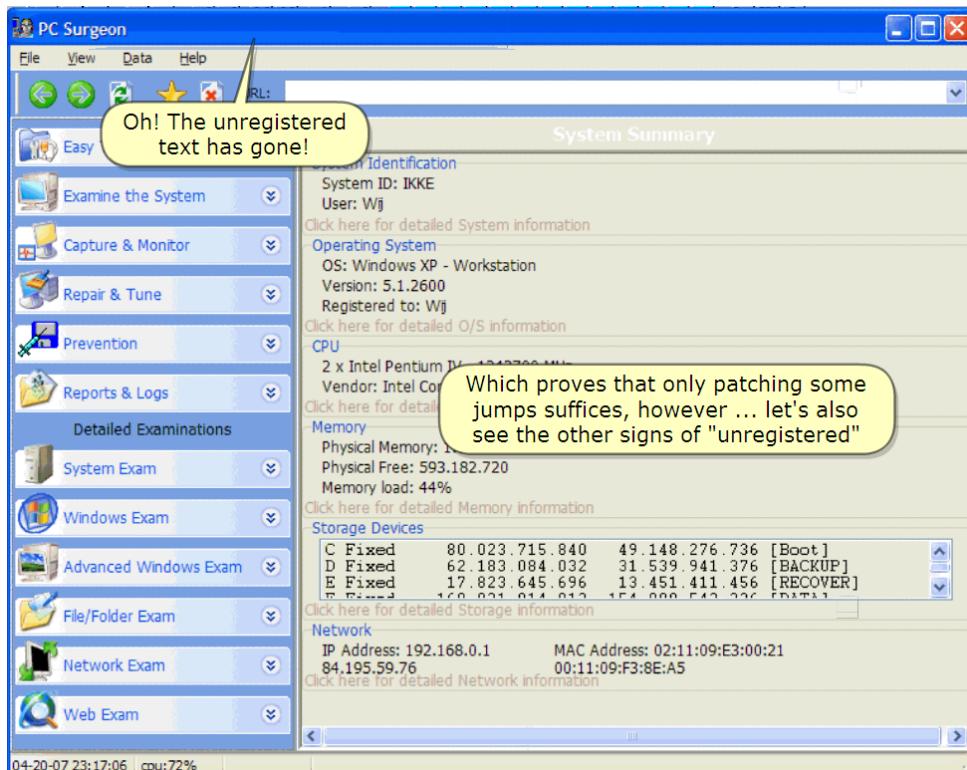
;)

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.

;) )

| Address  | Disassembly                | Text string                                 |
|----------|----------------------------|---|
| 00562381 | ASCII "BitBttn1"           | ASCII "<Unregistered Version>"              |
| 0056238E | ASCII "Label6"             | ASCII " Version. Do Not Copy!"              |
| 0056239C | ASCII "Label7"             | ASCII "Licensed Version. Do Not Copy!"      |
| 005623AB | ASCII "BitBttn4C"          | ASCII "Software\\Dean Software\\PC Surgeon" |
| 005623BE | ASCII "FormShow"           | ASCII "sUserF"                              |
| 005623CD | ASCII "BitBttn1C"          | ASCII "Config"                              |
| 005623D0 | ASCII "TAboutForm"         | ASCII "This copy is licensed to: "          |
| 005623FA | ASCII "TAboutForm"         | ASCII "sUserF"                              |
| 0056240F | ASCII "About"              | ASCII "Config"                              |
| 0056247B | MOU EDX,pcsurgeo.005626B8  | ASCII "All rights reserved."                |
| 005624AF | MOU ECX,pcsurgeo.005626D8  |   |
| 005624C0 | MOU EDX,pcsurgeo.005626F8  |   |
| 00562507 | MOU ECX,pcsurgeo.00562720  |   |
| 00562532 | MOU ECX,pcsurgeo.0056274C  |   |
| 00562537 | MOU EDX,pcsurgeo.0056275C  |   |
| 00562573 | PUSH ECX,pcsurgeo.0056276C |   |
| 00562581 | MOU ECX,pcsurgeo.0056274C  |   |
| 00562586 | MOU EDX,pcsurgeo.0056275C  |   |
| 00562600 | MOU ECX,pcsurgeo.0056279C  |   |

Let's go see it in the code : doubleclick

Code 에서 보자 : doubleclick

;)

C CPU - main thread, module pcsurgeo

| Address  | OpCode | Mnemonic                         | Operands                         |
|----------|--------|----------------------------------|----------------------------------|
| 00562421 | . . .  | PUSH EBX                         |                                  |
| 00562422 | . . .  | XOR EDX, EDX                     |                                  |
| 00562424 | . . .  | MOV DWORD PTR SS:[EBP-114], EDX  |                                  |
| 00562428 | . . .  | MOV DWORD PTR SS:[EBP-118], EDX  |                                  |
| 00562430 | . . .  | MOV DWORD PTR SS:[EBP-120], EDX  |                                  |
| 00562436 | . . .  | MOV DWORD PTR SS:[EBP-224], EDX  |                                  |
| 0056243C | . . .  | MOV DWORD PTR SS:[EBP-110], EDX  |                                  |
| 00562442 | . . .  | MOV DWORD PTR SS:[EBP-C], EDX    |                                  |
| 00562448 | . . .  | MOV DWORD PTR SS:[EBP-4], EDX    |                                  |
| 0056244E | . . .  | XOR EBX, EBX                     |                                  |
| 00562450 | . . .  | XOR EAX, EAX                     |                                  |
| 00562452 | . . .  | PUSH EBP                         |                                  |
| 00562453 | . . .  | PUSH pcsurgeo.005626A8           |                                  |
| 00562458 | . . .  | PUSH DWORD PTR FS:[EAX]          |                                  |
| 0056245B | . . .  | MOU EDWORD PTR FS:[EAX], ESP     |                                  |
| 0056245E | . . .  | MOU EAX, EDWORD PTR DS:[60EB8C]  |                                  |
| 00562463 | . . .  | CMP BYTE PTR DS:[EAX], 0         |                                  |
| 00562466 | . . .  | JNZ SHORT pcsurgeo.0056248A      |                                  |
| 00562468 | . . .  | MOU EAX, EDWORD PTR DS:[EBX+308] |                                  |
| 0056246E | . . .  | MOU DL, 1                        |                                  |
| 00562470 | . . .  | E8 1F78F1FF                      | CALL pcsurgeo.00479C94           |
| 00562475 | . . .  | E8 883 24030000                  | MOU EAX, EDWORD PTR DS:[EBX+824] |
| 0056247B | . . .  | E8 883 34030000                  | MOU EDX, EDWORD PTR DS:[EBX+368] |
| 00562480 | . . .  | E8 1F79F1FF                      | CALL pcsurgeo.00479D94           |
| 00562485 | . . .  | E9 E0010000                      | JMP pcsurgeo.0056266A            |
| 00562488 | . . .  | E8 883 00030000                  | MOU EAX, EDWORD PTR DS:[EBX+308] |
| 00562490 | . . .  | SIGP                             | XOR EDX, EDX                     |
| 00562492 | . . .  | E8 F077F1FF                      | CALL pcsurgeo.00479C94           |
| 00562497 | . . .  | E8 883 34030000                  | MOU EDX, EDWORD PTR DS:[EBX+334] |
| 0056249D | . . .  | E8 BE2DEAFF                      | CALL pcsurgeo.00405268           |
| 005624A2 | . . .  | 65C8                             | TEST EAX, EAX                    |
| 005624A4 | . . .  | 7E 23                            | JLE SHORT pcsurgeo.005624C9      |
| 005624A6 | . . .  | E8 883 34030000                  | MOU EDX, EDWORD PTR DS:[EBX+334] |
| 005624A9 | . . .  | 6045 F4                          | LEA EAX, EDWORD PTR SS:[EBP-C]   |
| 005624AF | . . .  | E9 D8255600                      | MOU ECX, pcsurgeo.005626D8       |
| 005624B4 | . . .  | E8 F32DEAFF                      | CALL pcsurgeo.0040529C           |
| 005624B9 | . . .  | 6855 F4                          | MOU EDX, EDWORD PTR SS:[EBP-C]   |
| 005624BC | . . .  | E8 883 24030000                  | MOU EAX, EDWORD PTR DS:[EBX+324] |
| 005624C2 | . . .  | E8 D078F1FF                      | CALL pcsurgeo.00479D94           |
| 005624C7 | . . .  | E8 10                            | JMP SHORT pcsurgeo.00562409      |
| 005624C9 | . . .  | E8 883 24030000                  | MOU EAX, EDWORD PTR DS:[EBX+324] |
| 005624CF | . . .  | E8 F8265600                      | MOU EDX, pcsurgeo.005626F8       |
| 005624D4 | . . .  | E8 CB78F1FF                      | CALL pcsurgeo.00479D94           |
| 005624D9 | . . .  | E8 883 34030000                  | MOU EAX, EDWORD PTR DS:[EBX+334] |
| 005624DF | . . .  | E8 7C2DEAFF                      | CALL pcsurgeo.00405268           |

00562688=pcsurgeo.005626B8 (ASCII "<Unregistered Version>")  
EDX=00000000

ASCII "<Unregistered Version>"

ASCII " Version. Do Not Copy!"

Oh! And see this!

Goodboy

ASCII "Licensed Version. Do Not Copy!"

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.

C CPU - main thread, module pcsecureo

|          |                 |                                  |
|----------|-----------------|----------------------------------|
| 00562414 | 00              | DB 00                            |
| 00562415 | 00              | DB 00                            |
| 00562416 | 88C0            | MOV EAX, EAX                     |
| 00562418 | 55              | PUSH EBP                         |
| 00562419 | 8BEC            | MOU ECX, ESP                     |
| 0056241B | 81C4 DCFDFFFF   | ADD ESP, 224                     |
| 00562421 | 53              | PUSH EBX                         |
| 00562422 | 33D2            | XOR EDX, EDX                     |
| 00562424 | 8995 ECFFFFFFFF | MOU DWORD PTR SS:[EBP-114], EDX  |
| 0056242A | 8995 E8FFFFFF   | MOU DWORD PTR SS:[EBP-114], ECX  |
| 00562430 | 8995 E0FFFFFF   | MOU DWORD PTR SS:[EBP-114], EDI  |
| 00562436 | 8995 DCFDFFFF   | MOU DWORD PTR SS:[EBP-114], EDX  |
| 0056243C | 8995 E4FFFFFF   | MOU DWORD PTR SS:[EBP-114], ECX  |
| 00562442 | 8995 F0FFFFFF   | MOU DWORD PTR SS:[EBP-114], EDI  |
| 00562448 | 8995 F4         | MOU DWORD PTR SS:[EBP-114], EDX  |
| 0056244B | 8995 FC         | MOU DWORD PTR SS:[EBP-114], ECX  |
| 0056244E | 8B08            | MOU EBX, EAX                     |
| 00562450 | 33C0            | XOR EAX, EAX                     |
| 00562452 | 55              | PUSH EBP                         |
| 00562453 | 68 A3265600     | PUSH pcsecureo.005626A8          |
| 00562458 | 64:F30          | PUSH DWORD PTR FS:[EAX]          |
| 0056245B | 64:8920         | MOU EDX, DWORD PTR DS:[EAX], ESP |
| 0056245E | A1 8CEB6000     | MOU EAX, DWORD PTR DS:[60EBE8C]  |
| 00562463 | 8038 00         | CMP BTLE PTR DS:[EAX], 0         |
| 00562465 | 75 22           | JNZ SHORT pcsecureo.0056248A     |
| 00562468 | 8B83 08030000   | MOU EAX, DWORD PTR DS:[EBX+308]  |
| 0056246E | B8 01           | MOU DL, 1                        |
| 00562470 | E8 1F78F1FF     | CALL pcsecureo.00479C94          |
| 00562475 | 8B83 24030000   | MOU EAX, DWORD PTR DS:[EBX+324]  |
| 00562478 | B8 82656000     | MOU EDX, pcsecureo.00562688      |
| 00562480 | E9 1F79F1FF     | CALL pcsecureo.00479D04          |
| 00562483 | E9 E0010000     | JP pcsecureo.005626A8            |
| 0056248A | > 8B83 08030000 | MOU EAX, DWORD PTR DS:[EBX+308]  |
| 00562490 | 33D2            | XOR EDX, EDX                     |
| 00562492 | E8 FD77F1FF     | CALL pcsecureo.00479C94          |
| 00562497 | 8B83 24030000   | MOU EAX, DWORD PTR DS:[EBX+324]  |
| 0056249D | E8 B22DEAFF     | CALL pcsecureo.00405268          |
| 005624A2 | 8C00            | TEST EAX, EAX                    |
| 005624A4 | 7E 23           | JLE SHORT pcsecureo.005624C9     |
| 005624A6 | 8B93 34030000   | MOU EDX, DWORD PTR DS:[EBX+334]  |
| 005624AC | 8045 F4         | LEA EAX, DWORD PTR SS:[EBP-1C]   |
| 005624AF | B9 0B2656000    | MOU ECX, pcsecureo.00562608      |
| 005624B4 | E8 F32DEAFF     | CALL pcsecureo.004052AC          |
| 005624B9 | 8B85 F4         | MOU EDX, DWORD PTR SS:[EBP-C]    |
| 005624BC | 8B83 24030000   | MOU EAX, DWORD PTR DS:[EBX+324]  |
| 005624C2 | E8 DD78F1FF     | CALL pcsecureo.00479D04          |

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

ASCII " (Unregistered Version)"

ASCII " Version. Do Not Copy!"

## 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 pointer를 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 routine에 대하여 그것들의 string으로부터 setting을 찾았다. 재시작하고 모든 필요한 patch를 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.

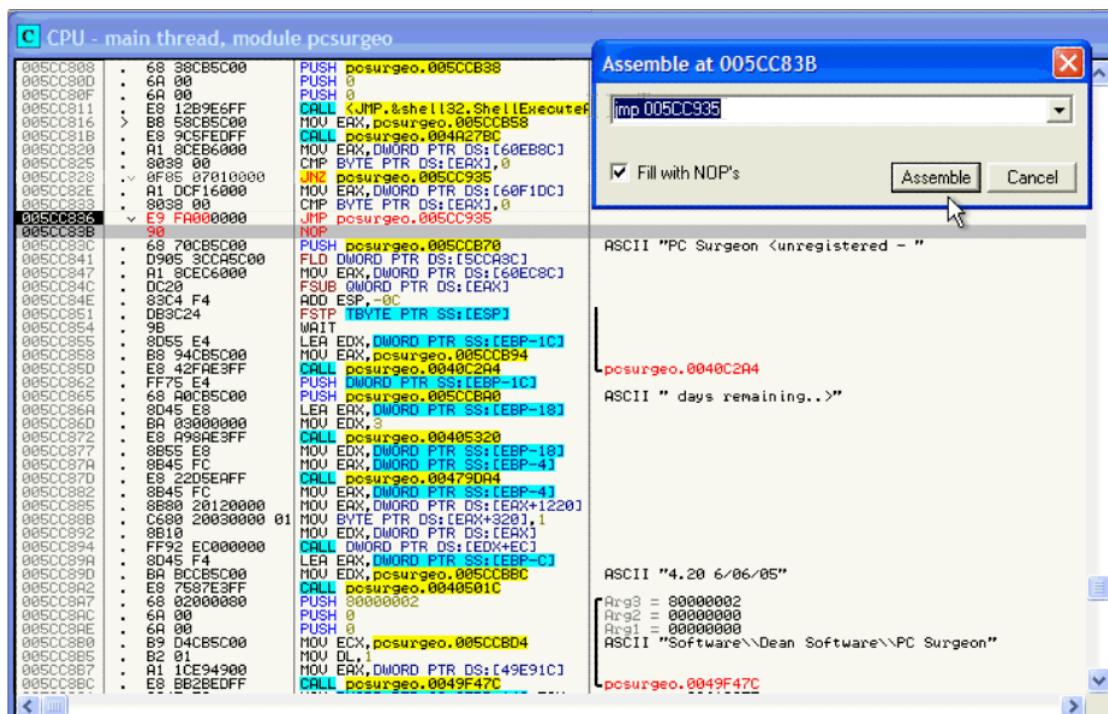
만약에 re-assembled 된 line 의 software BP 를 남겨놓았다면, Application 을 재시작 한 후에 Olly 는 혼란을 겪는다. 왜냐하면 Olly 는 software breakpoint 가 필요하다. 그러고나서 "충돌"

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.

C CPU - main thread, module pcsurgeo

```

00562419    s 55      PUSH EBP
00562419    . 8BEC    MOU EBP,ESP
00562419    . 81C4  DCFDFFFF ADD ESP,-224
00562419    . 53      PUSH EBX
00562422    . 33D2    XOR EDX,EDX
00562424    . 8995  ECFFFFFF MOU DWORD PTR SS:[EBP-114],EDX
00562424    . 8995  E0FEFFFF MOU DWORD PTR SS:[EBP-118],EDX
00562430    . 8995  E0FEFFFF MOU DWORD PTR SS:[EBP-120],EDX
00562436    . 8995  DCFFFFFF MOU DWORD PTR SS:[EBP-224],EDX
0056243C    . 8995  E4FFFFFF MOU DWORD PTR SS:[EBP-11C],EDX
00562442    . 8995  F0FFFFFF MOU DWORD PTR SS:[EBP-110],EDX
00562448    . 8955  F4      MOU DWORD PTR SS:[EBP-C],EDX
0056244B    . 8955  FC      MOU DWORD PTR SS:[EBP-4],EDX
0056244E    . 8BD8    MOU EBX,EAX
00562450    . 33C0    XOR EAX,EAX
00562452    . 55      PUSH EBP
00562453    . 68 A3265600 PUSH pcsurgeo.00562648
00562453    . 64:FF30    PUSH DWORD PTR FS:[EAX]
0056245B    . 64:8920    MOU DWORD PTR FS:[EAX],ESP
0056245B    . A1 8CE60000 MOU EAX,DWORD PTR DS:[60EB8C]
00562453    . 8038  00      CMP BYTE PTR DS:[EAX],0

00562466    > 75 22    JNZ SHORT pcsurgeo.0056248A
00562468    . 8B83  0000000 MOU EAX,DWORD PTR DS:[EBX+308]
0056246E    . B2 01      MOV DL,1
00562470    . E8 1F78F1FF CALL pcsurgeo.00479C94
00562470    . 8B83  24030000 MOU EAX,DWORD PTR DS:[EBX+324]
00562478    . 8B83  0265600 MOU EDX,pcsurgeo.00562688
00562480    . E8 1F78F1FF CALL pcsurgeo.00479D94
00562485    > E9 E0      TEST EAX,EBX
00562485    > 8B83  0000000 MOU EAX,DWORD PTR DS:[EBX+308]
00562490    . 83D0  00      TEST EBX,EBX
00562492    . E8 FD77F1FF CALL pcsurgeo.00479C94
00562492    . 8B83  34030000 MOU EAX,DWORD PTR DS:[EBX+334]
0056249D    . E8 BE2DEAFF CALL pcsurgeo.00405268
0056249D    . 89C5  00      TEST EAX,EBX
005624A2    > 75 23    JE SHORT pcsurgeo.005624C9
005624A6    . 8B95  34030000 MOU EDX,DWORD PTR DS:[EBX+334]
005624A6    . E8 D265600 MOU ECX,pcsurgeo.00562608
005624A6    . 8B45  F4      MOU EDX,DWORD PTR SS:[EBP-C]
005624A6    . B9 D8265600 MOU ECX,pcsurgeo.00405240
005624B4    . E8 BE2DEAFF CALL pcsurgeo.00405240
005624B8    . 8B55  F4      MOU EDX,DWORD PTR SS:[EBP-C]
005624B8    . 8B83  24030000 MOU EAX,DWORD PTR DS:[EBX+324]
005624C2    . E8 DD78F1FF CALL pcsurgeo.00479D94
005624C2    > E9 10      JNP SHORT pcsurgeo.005624D9
005624C7    > 8B83  24030000 MOU EAX,DWORD PTR DS:[EBX+324]
005624C7    . B9 F8265600 MOU EDX,pcsurgeo.005626F8

Not jumping

```

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

ASCII "Unregistered Version"

...though we want to jump this

ASCII "Version. Do Not Copy"

But we want to arrive here

ASCII "Licensed Version. Do Not Copy!"

```

00562424    . 8995  ECFFFFFF MOU DWORD PTR SS:[EBP-114],EDX
00562424    . 8995  E0FEFFFF MOU DWORD PTR SS:[EBP-118],EDX
00562430    . 8995  E0FEFFFF MOU DWORD PTR SS:[EBP-120],EDX
00562436    . 8995  DCFFFFFF MOU DWORD PTR SS:[EBP-224],EDX
0056243C    . 8995  E4FFFFFF MOU DWORD PTR SS:[EBP-11C],EDX
00562442    . 8995  F0FFFFFF MOU DWORD PTR SS:[EBP-110],EDX
00562448    . 8955  F4      MOU DWORD PTR SS:[EBP-C],EDX
0056244B    . 8955  FC      MOU DWORD PTR SS:[EBP-4],EDX
0056244E    . 8BD8    MOU EBX,EAX
00562450    . 33C0    XOR EAX,EAX
00562452    . 55      PUSH EBP
00562453    . 68 A3265600 PUSH pcsurgeo.00562648
00562453    . 64:FF30    PUSH DWORD PTR FS:[EAX]
0056245B    . 64:8920    MOU DWORD PTR FS:[EAX],ESP
0056245B    . A1 8CE60000 MOU EAX,DWORD PTR DS:[60EB8C]
00562453    . 8038  00      CMP BYTE PTR DS:[EAX],0

00562466    > 74 22    JE SHORT pcsurgeo.0056248A
00562468    . 8B83  0000000 MOU EAX,DWORD PTR DS:[EBX+308]

Assemble at 00562468
JZ SHORT 0056248A
Fill with NOP's
Assemble Cancel

```

C CPU - main thread, module pcsurgeo

```

00562418 $ 55 PUSH EBP
00562419 • 8BEC MOU EBP,ESP
0056241B • 81C4 DCFDFFFF ADD ESP,-224
00562421 • 53 PUSH EBX
00562422 • 33D2 XOR EDX,EDX
00562424 • 8995 ECFFFFFF MOU DWORD PTR SS:[EBP-114],EDX
00562429 • 8995 E8FFFFFF MOU DWORD PTR SS:[EBP-118],EDX
00562430 • 8995 E0FFFFFF MOU DWORD PTR SS:[EBP-120],EDX
00562436 • 8995 DCFFFFFF MOU DWORD PTR SS:[EBP-224],EDX
0056243C • 8995 E4FFFFFF MOU DWORD PTR SS:[EBP-11C],EDX
00562442 • 8995 F0FFFFFF MOU DWORD PTR SS:[EBP-110],EDX
00562448 • 8955 F4 MOU DWORD PTR SS:[EBP-C],EDX
0056244B • 8955 FC MOU DWORD PTR SS:[EBP-4],EDX
0056244E • 8BD8 MOU EBX,ERX
00562450 • 33C0 XOR EAX,ERX
00562452 • 55 PUSH EBP
00562453 • 68 A3265600 PUSH posurgeo.00562693
00562458 • 64:FF30 PUSH DWORD PTR FS:[ERX]
0056245B • 64:8920 MOU DWORD PTR FS:[ERX],ESP
0056245E • A1 SCEB6000 MOU ERX,DWORD PTR DS:[60EB8C]
00562463 • 8038 00 CMP BYTE PTR DS:[ERX],0
00562466 . 74 22 JE SHORT posurgeo.0056248A
00562468 • 8B83 00000000 MOU EAX, DWORD PTR DS:[EBX+308]
0056246E . B2 01 MOU DL,1
00562470 . E8 1F78F1FF CALL posurgeo.00479C94
00562475 . 8B83 24030000 MOU ERX, DWORD PTR DS:[EBX+324]
00562478 . BA B8265600 MOU EDX, posurgeo.00562688
00562480 . E8 1F79F1FF CALL posurgeo.00479D44
00562485 . E9 E0010000 JMP posurgeo.00479D44
00562489 .> 8B83 00000000 MOU ERX, DWORD PTR DS:[EBX+308]
00562490 . 33D2 XOR EDX,EDX
00562491 . E8 F8265600 CALL posurgeo.00562693
00562493 .> 8B83 00000000 MOU EDX, posurgeo.00562688
00562495 .> 8B83 24030000 MOU ERX, DWORD PTR DS:[EBX+324]
00562498 .> 8B83 24030000 MOU ERX, DWORD PTR DS:[EBX+324]
0056249B .> 8B83 24030000 MOU ERX, DWORD PTR DS:[EBX+324]
0056249E .> 8B83 24030000 MOU ERX, DWORD PTR DS:[EBX+324]
005624A1 .> 8B83 24030000 MOU ERX, DWORD PTR DS:[EBX+324]
005624A4 .> 8B83 24030000 MOU EDX, posurgeo.00562688
005624A6 .> 8B83 34030000 MOU EDX, DWORD PTR DS:[EBX+384]
005624AC .> 8D45 F4 LEA EDX,DWORD PTR SS:[EBP-C]
005624B0 .> 89 D82655 MOU EDX, posurgeo.00562688
005624B4 .> 8B F32D8 TEST EDX,EDX
005624B9 .> 8B83 24030000 MOU ERX, DWORD PTR DS:[EBX+324]
005624C2 .> 8B83 24030000 MOU ERX, DWORD PTR DS:[EBX+324]
005624C5 .> 8B83 24030000 MOU ERX, DWORD PTR DS:[EBX+324]
005624C8 .> 8B83 24030000 MOU ERX, DWORD PTR DS:[EBX+324]
005624D1 .> 8B83 24030000 MOU ERX, DWORD PTR DS:[EBX+324]
005624D4 .> 8B83 24030000 MOU ERX, DWORD PTR DS:[EBX+324]
005624D7 .> 8B83 24030000 MOU ERX, DWORD PTR DS:[EBX+324]
005624D9 .> 8B83 24030000 MOU ERX, DWORD PTR DS:[EBX+324]
005624F0 .> 8B83 24030000 MOU EDX, posurgeo.00562688

```

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

ASCII "<Unregistered Version>"

ASCII "Licensed Version. Do Not Copy!"

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.

C CPU - main thread, module pcsurgeo

```

00562418 $ 55 PUSH EBP
00562419 • 8BEC MOU EBP,ESP
0056241B • 81C4 DCFDFFFF ADD ESP,-224
00562421 • 53 PUSH EBX
00562422 • 33D2 XOR EDX,EDX
00562424 • 8995 ECFFFFFF MOU DWORD PTR SS:[EBP-114],EDX
00562429 • 8995 E8FFFFFF MOU DWORD PTR SS:[EBP-118],EDX
00562430 • 8995 E0FFFFFF MOU DWORD PTR SS:[EBP-120],EDX
00562436 • 8995 DCFFFFFF MOU DWORD PTR SS:[EBP-224],EDX
0056243C • 8995 E4FFFFFF MOU DWORD PTR SS:[EBP-11C],EDX
00562442 • 8995 F0FFFFFF MOU DWORD PTR SS:[EBP-110],EDX
00562448 • 8955 F4 MOU DWORD PTR SS:[EBP-C],EDX
0056244B • 8955 FC MOU DWORD PTR SS:[EBP-4],EDX
0056244E • 8BD8 MOU EBX,ERX
00562450 • 33C0 XOR EAX,ERX
00562452 • 55 PUSH EBP
00562453 • 68 A3265600 PUSH posurgeo.00562693
00562458 • 64:FF30 PUSH DWORD PTR FS:[ERX]
0056245B • 64:8920 MOU DWORD PTR FS:[ERX],ESP
0056245E • A1 SCEB6000 MOU ERX,DWORD PTR DS:[60EB8C]
00562463 • 8038 00 CMP BYTE PTR DS:[ERX],0
00562466 . 74 22 JE SHORT posurgeo.0056248A
00562468 • 8B83 00000000 MOU EAX, DWORD PTR DS:[EBX+308]
0056246E . B2 01 MOU DL,1
00562470 . E8 1F78F1FF CALL posurgeo.00479C94
00562475 . 8B83 24030000 MOU ERX, DWORD PTR DS:[EBX+324]
00562478 . BA B8265600 MOU EDX, posurgeo.00562688
00562480 . E8 1F79F1FF CALL posurgeo.00479D44
00562485 . E9 E0010000 JMP posurgeo.00479D44
00562489 .> 8B83 00000000 MOU ERX, DWORD PTR DS:[EBX+308]
00562490 . 33D2 XOR EDX,EDX
00562491 . E8 F8265600 CALL posurgeo.00562693
00562493 .> 8B83 00000000 MOU EDX, posurgeo.00562688
00562495 .> 8B83 24030000 MOU ERX, DWORD PTR DS:[EBX+324]
00562498 .> 8B83 24030000 MOU ERX, DWORD PTR DS:[EBX+324]
0056249B .> 8B83 24030000 MOU ERX, DWORD PTR DS:[EBX+324]
0056249E .> 8B83 24030000 MOU ERX, DWORD PTR DS:[EBX+324]
005624A1 .> 8B83 24030000 MOU ERX, DWORD PTR DS:[EBX+324]
005624A4 .> 8B83 24030000 MOU EDX, posurgeo.00562688
005624A6 .> 8B83 34030000 MOU EDX, DWORD PTR DS:[EBX+384]
005624AC .> 8D45 F4 LEA EDX,DWORD PTR SS:[EBP-C]
005624B0 .> 89 D82655 MOU EDX, posurgeo.00562688
005624B4 .> 8B F32D8 TEST EDX,EDX
005624B9 .> 8B83 24030000 MOU ERX, DWORD PTR DS:[EBX+324]
005624C2 .> 8B83 24030000 MOU ERX, DWORD PTR DS:[EBX+324]
005624C5 .> 8B83 24030000 MOU ERX, DWORD PTR DS:[EBX+324]
005624C8 .> 8B83 24030000 MOU ERX, DWORD PTR DS:[EBX+324]
005624D1 .> 8B83 24030000 MOU ERX, DWORD PTR DS:[EBX+324]
005624D4 .> 8B83 24030000 MOU ERX, DWORD PTR DS:[EBX+324]
005624D7 .> 8B83 24030000 MOU ERX, DWORD PTR DS:[EBX+324]
005624D9 .> 8B83 24030000 MOU ERX, DWORD PTR DS:[EBX+324]
005624F0 .> 8B83 24030000 MOU EDX, posurgeo.00562688

```

Aha! We jump to the goodboy!

ASCII " Version. Do Not Copy!"

ASCII "Licensed Version. Do Not Copy!"

Aha! We jump to the goodboy !

아하! 우리는 goodboy로 jump 한다.

!

Scroll down to see the way ahead ;)

Scroll 미리 앞 부분을 보기 내려봐.

C CPU - main thread, module pcsurgeo

```
005624C7 > EB 10 JMP SHORT pcsurgeo.005624D9
005624C9 > 9B83 24030000 MOU EAX,DIWORD PTR DS:[EBX+824]
005624CF > BA F8265600 MOU EDX,DIWORD PTR DS:[EBX+828]
005624D4 > E8 CB78F1FF CALL pcsurgeo.00479DA4
005624D9 > 8B83 38030000 MOU EAX,DIWORD PTR DS:[EBX+838]
005624DF > E8 7C2DEAFF CALL pcsurgeo.00405268
005624E4 > 85C0 TEST EAX,EAX
005624E6 > 7E 16 JLE SHORT pcsurgeo.005624FE
005624E8 > 8B93 38030000 MOU EDX,DIWORD PTR DS:[EBX+838]
005624EE > 8B83 38030000 MOU EAX,DIWORD PTR DS:[EBX+830]
005624F4 > E8 AB78F1FF CALL pcsurgeo.00479DA4
005624F9 > E9 6C010000 JMP pcsurgeo.0056266A
005624FE > 68 02000000 PUSH 80000002
00562503 > 6A 00 PUSH 0
00562505 > 6A 00 PUSH 0
00562507 > B9 20275600 MOU ECX,pcsurgeo.00562720
0056250C > B2 01 MOU DL,1
0056250E > A1 1CE94900 MOU EAX,DIWORD PTR DS:[49E91C]
00562513 > E8 64CF3FF CALL pcsurgeo.0049F470
00562518 > 8945 F8 MOU DIWORD PTR SS:[EBP-8],EAX
0056251B > 33C0 XOR EAX,EAX
0056251D > 55 PUSH EBP
0056251E > 68 63265600 PUSH pcsurgeo.00562663
00562523 > 64:FF30 PUSH DIWORD PTR FS:[EAX]
00562526 > 64:8920 MOU DIWORD PTR FS:[EAX],ESP
00562529 > 6A 00 PUSH 0
0056252B > 8D85 F0FFFFF LEA EAX,DIWORD PTR SS:[EBP-110]
00562531 > 50 PUSH EAX

00562532 > 0056274C
00562537 > 00562750
0056253C > SS:[EBP-8]
0056253F > SS:[EBP-110]
00562544 > 9F5FC
00562549 > SS:[EBP-10C]
00562550 > A3018
00562555 > SS:[EBP-10C]
0056255B > SS:[EBP-4]
0056255E > 4051E8
00562563 > EB45 FL
00562566 > E8 F52CEAFF CALL pcsurgeo.00405268
0056256B > 85C0 TEST EAX,EAX
0056256D > 0F8E C2000000 CALL pcsurgeo.00562695
00562573 > 68 6C275600 PUSH pcsurgeo.00562760
00562578 > 6A 00 PUSH 0
0056257A > 8D85 E4FFFFF LEA EAX,DIWORD PTR SS:[EBP-110]
00562580 > 50 PUSH EAX

But this will need further investigation. So keep stepping F8 instead of pressing run
```

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

ASCII "Licensed Version. Do Not Copy!"  
Arg3 = 80000002  
Arg2 = 00000000  
Arg1 = 00000000  
ASCII "Software\\Dean Software\\PC Surgeon"  
pcsurgeo.0049F47C

ASCII "sUserF"  
ASCII "Config"

ASCII "This copy is licensed to: "

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.

C CPU - main thread, module pcsurgeo

```

005624C7 > EB 10 JMP SHORT pcsurgeo.005624D9
005624C9 . EB F8265900 MOU ECX,DMORD PTR DS:[EBX+324]
005624CF . EB CB79F1FF CALL pcsurgeo.00479D44
005624D4 > EB 05 38030000 MOU ECX,DMORD PTR DS:[EBX+338]
005624D9 . EB 05 C2DEFF CALL pcsurgeo.00405268
005624E4 . EB 00 SS00 TEST ERM,ERZ
005624E5 . EB 16 JLE SHORT pcsurgeo.005624FE
005624E8 > EB 93 39030000 MOU EDX,DMORD PTR DS:[EBX+338]
005624EE . EB 93 39030000 MOU ECX,DMORD PTR DS:[EBX+338]
005624F4 > EB AB79F1FF CALL pcsurgeo.00479D44
005624F9 . EB E6C010000 JMP pcsurgeo.0056266A
005624FE > EB 00 02000000 PUSH 00000002
00562503 . EB 00 PUSH 0
00562505 . EB 00 PUSH 0
00562507 > EB 20275600 MOU ECX,pcsurgeo.00562720
0056250C . EB 01 MOU DL,1
0056250E > A1 1CE94900 MOU ECX,DMORD PTR DS:[49E91C]
00562513 . EB 64CFF3FF CALL pcsurgeo.0049F47C
00562518 > EB 8945 F8 MOU DMORD PTR SS:[EBP-8],EAX
0056251B . EB 93C0 XOR ECX,EAX
0056251D . EB 00 PUSH EBP
0056251E > EB 63265600 PUSH pcsurgeo.00562668
00562523 . EB 4:FF90 PUSH DMORD PTR FS:[ERX]
00562526 > EB 4:8920 MOU DMORD PTR FS:[ERX],ESP
00562529 . EB 00 PUSH 0
0056252B > EB 0D85 F0FFFFF LEA ECX,DMORD PTR SS:[EBP-110]
00562531 . EB 50 MOV ECX,pcsurgeo.0056274C
00562532 > EB 4C275600 ASCII "sUser"
00562537 . BA 5C275600 CALL ECX,pcsurgeo.004951E8
0056253C . EB 8945 F8 MOU ECX,DMORD PTR SS:[EBP-4]
0056253D . EB 8900FF CALL pcsurgeo.00405268
00562544 > EB 8885 F0FF TEST ERM,ERZ
00562549 . EB 8905 F4FF CALL ECX,pcsurgeo.00562635
00562550 > EB 8905 30FF LEA ECX,DMORD PTR SS:[EBP-110]
00562555 . EB 0D95 F4FF CALL ECX,pcsurgeo.0056274C
00562556 > EB 8945 FC MOU ECX,DMORD PTR SS:[EBP-4]
00562557 . EB 8945 F52CEAFF PUSH pcsurgeo.0056276C
00562558 > EB 8945 F52CEAFF CALL pcsurgeo.00405268
00562559 . EB 50 TEST ECX,EAX
00562560 > EB 0F88 C2000000 ASCII "This copy is licensed to: "
00562573 . EB 6C275600 PUSH pcsurgeo.0056276C
00562578 > EB 00 PUSH 0
0056257A > EB 0D85 E4FEFFFF LEA ECX,DMORD PTR SS:[EBP-110]
00562580 . EB 50 PUSH ECX

```

It means there was another verification in this call. Let's help the application in the right direction.

That's not what we want!

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 patch를 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 file로 저장하는 것이 필요하다.

Olly로 돌아가자.

## 5. Testing the patched application

We've made/found several patches so far.

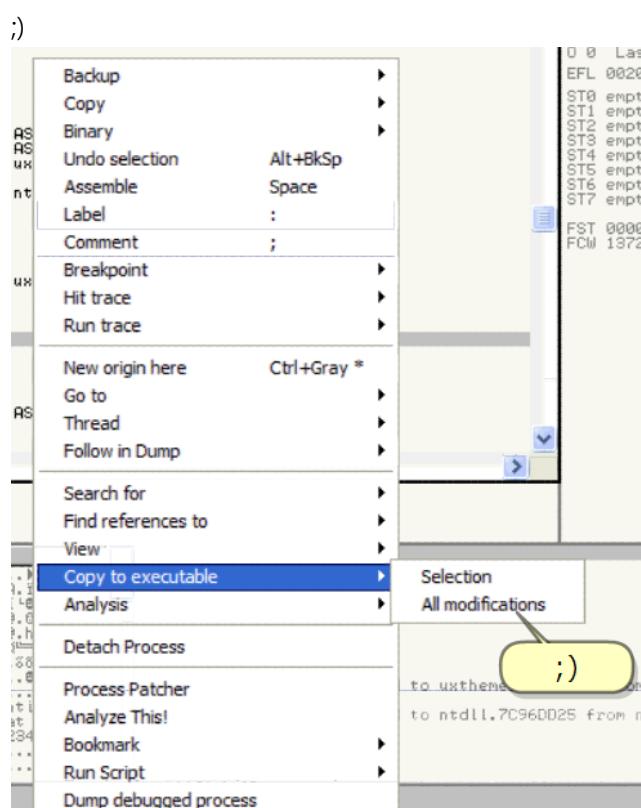
우리는 지금까지 몇가지 patch를 만들고/찾았다.

So, we will have to save "All changes" to file.

그래서, 우리는 file로 저장하기 위한 "All changes"가 있다.

Right-click to do so.

rightclick하고 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.

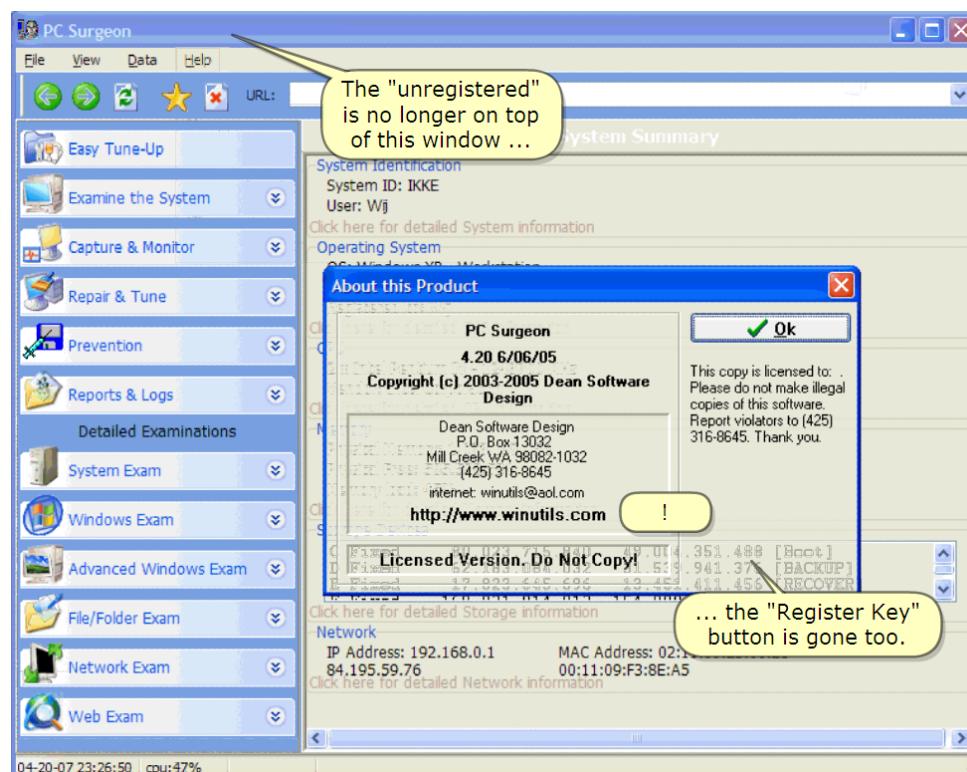
No more unregistered text ...

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 명청한 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.

```
;) ;)
005CC800  .  0H 00    PUSH 0
005CC801  .  0A 00    PUSH 0
005CC811  >  B8 12B9E6FF  CALL <JMP.&shell32.ShellExecuteA
005CC816  .  BB 58C85C00  MOU EAX, pcSurgeo, 005CC858
005CC81B  .  B9 9C5FEDFF  CALL pcSurgeo, 004A2780
005CC820  .  B1 8CEC6000  MOU EAX, DWORD PTR DS:[60EB8C]
005CC825  .  0B88 00    CMP BYTE PTR DS:[EAX], 0
005CC828  .  <FB8 07010000  JNZ pcSurgeo, 005CC935
005CC82E  .  B9 01 DCF16000  MOU EAX, DWORD PTR DS:[60F1DC]
005CC833  .  0B88 00    CMP BYTE PTR DS:[EAX], 0
005CC836  .  <FB8 F9000000  JNZ pcSurgeo, 005CC935
005CC83C  .  B9 68 70C85C00  MOU EAX, DWORD PTR DS:[5CC83C]
005CC841  .  D905 3CC85C00  FLD DWORD PTR DS:[5CC83C]
005CC847  .  A1 8CEC6000  MOU EAX, DWORD PTR DS:[60EC8C]
005CC84C  .  DC28        FSUB QWORD PTR DS:[EAX]
005CC84E  .  B9 9C4 F4    ADD ESP, -8C
005CC851  .  DB4C24    FSTP BYTE PTR SS:[ESP]
005CC854  .  9B          WAIT
005CC855  .  B9 55 E4    LEA EDX, DWORD PTR SS:[LEBP-1C]
005CC858  .  B8 94C85C00  MOU EAX, pcSurgeo, 005CB94
005CC85D  .  B9 42F8E3FF  CALL pcSurgeo, 0040C294
005CC862  .  FF75 E4    PUSH DWORD PTR SS:[LEBP-1C]
005CC865  .  B9 A0C85C00  PUSH pcSurgeo, 005CCB80
005CC86A  .  B9 D45 E8    LEA EAX, DWORD PTR SS:[LEBP-18]
005CC86D  .  B9 03000000  MOU EDX, 3
005CC872  .  B9 R98E3FF  CALL pcSurgeo, 00405320
005CC877  .  B9 55 E8    MOU EDX, DWORD PTR SS:[LEBP-18]
005CC879  .  B9 45 FC    MOU EAX, DWORD PTR SS:[LEBP-4]
005CC87D  .  B9 22D5EAFF  CALL pcSurgeo, 00479DA4
005CC882  .  B9 45 FC    MOU EAX, DWORD PTR SS:[LEBP-4]
005CC885  .  B9B0 28120000  MOU EAX, DWORD PTR DS:[EAX+1220]
005CC88B  .  C680 20030000  B1 01    BYTE PTR DS:[EAX+320], 1
005CC892  .  B9B10        MOU EDX, DWORD PTR DS:[ERX]
005CC894  .  FF92 EC000000  CALL DWORD PTR DS:[EDX+E8]
005CC899  .  B9 D45 F4    LEA EAX, DWORD PTR SS:[LEBP-C]
005CC89D  .  B9 BCC85C00  MOU EDX, pcSurgeo, 005CBBC
005CC89E  .  B9 7587E3FF  CALL pcSurgeo, 0040501C
005CC8A0  .  00 00        ASCII "4.20 6/06/08"

DS[10060EB8C]=00610C4A
ERX=00610C4A (pc)
Copy pane to clipboard
Modify data
Address  Hex dump
Follow address in Dump
Follow value in Dump
Appearance ;)



|          | ASCII                                           |       |
|----------|-------------------------------------------------|-------|
| 0060B800 | 00 00 00 00                                     | 0012F |
| 0060B810 | C0 20 00 00                                     | 0012F |
| 0060B820 | 40 5E 00 00                                     | 0012F |
| 0060B830 | 00 80 00 00                                     | 0012F |
| 0060B840 | 3C 00 00 00                                     | 0012F |
| 0060B850 | 00 CB 00 00                                     | 0012F |
| 0060B860 | 00 DE DF E0 E1 E3 00 E4 E5 80 40 00 00 00 00 00 | 0012F |
| 0060B870 | 00 00 00 00 01 00 00 00 02 00 00 00 03 00 00 00 | 0012F |
| 0060B880 | 00 00 00 00 00 00 00 00 04 72 72 6F             | 0012F |
| 0060B890 | 72 00 BB C0 52 75 6E 74 69 6D 65 20 65 72 72 6F | 0012F |


```

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.

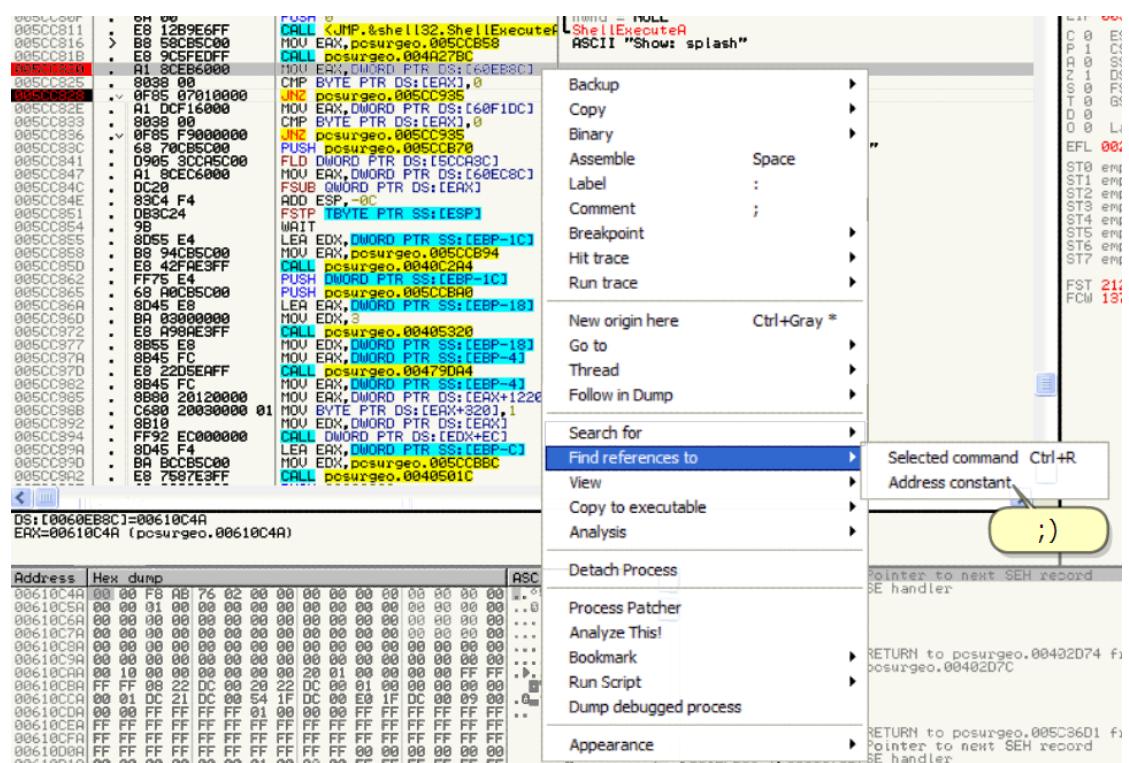
One way is using a HW or MEM BP on write (or access) on this byte, but let's leave this for later (it's a little more complicated) because we have another way that is very easy:

한 가지 방법은 HW 나 Memory BP 를 byte 에 사용하는 것이다. 그러나 나중을 위해 이것을

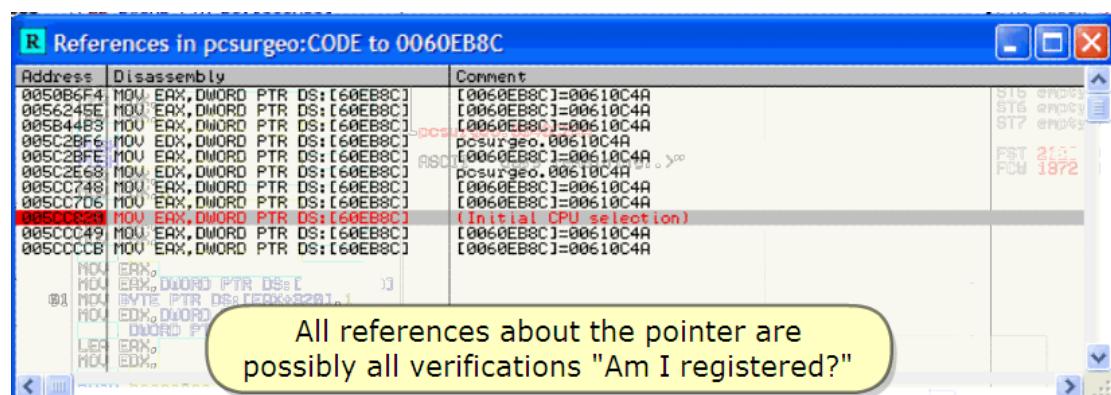
남긴다.(꽤 복잡하다) 왜냐하면 우리는 다른 방법을 가지고 있는데 그것은 매우 쉽다. :

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:

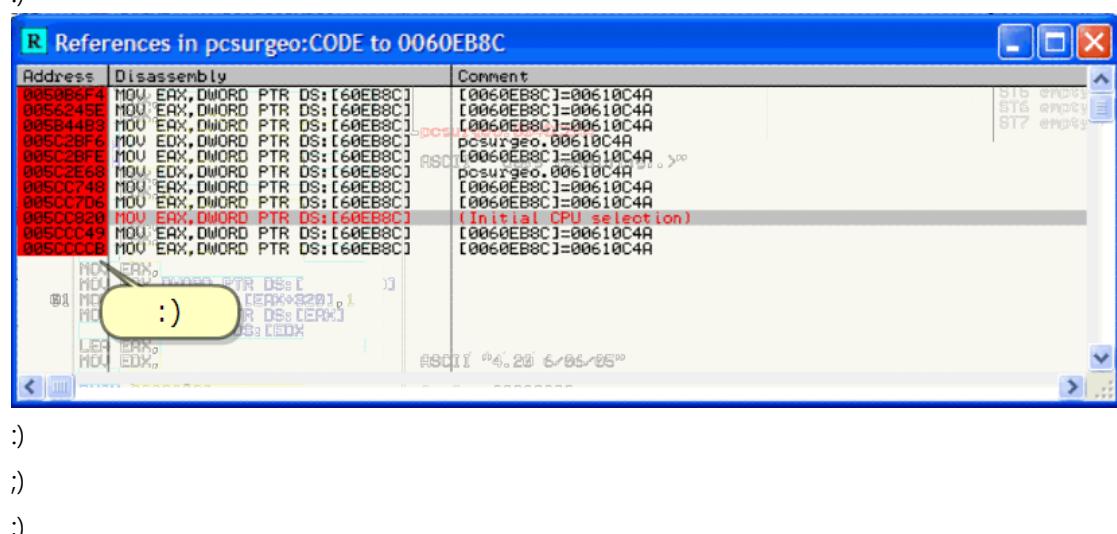
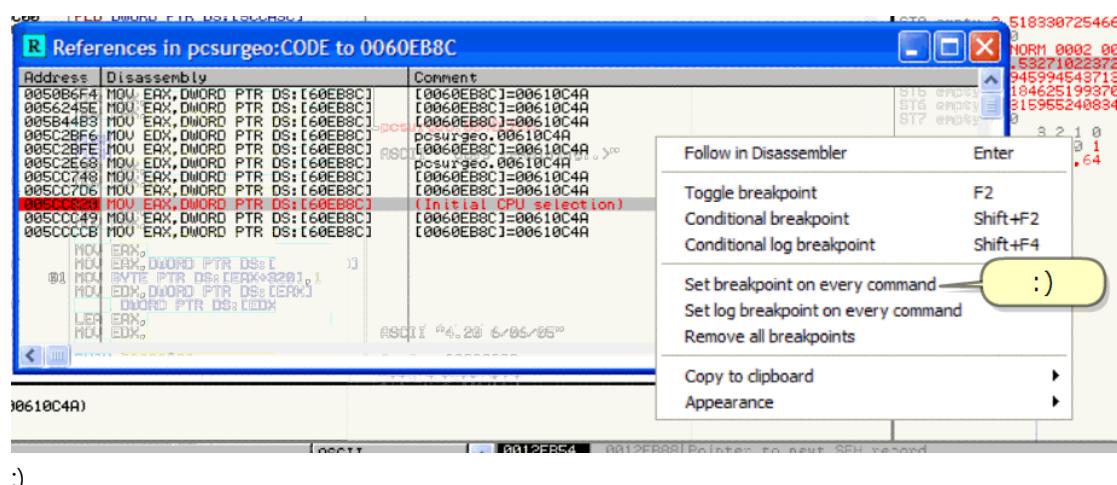
The idea is now to set BP's on every reference to the pointer so that we can break when restarting.

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

**C CPU - main thread, module pcsurgeo**

```

005C2B81 . 8998 24120000 MOV DWORD PTR DS:[EAX+1224],EBX
005C2B87 . 8BC3 MOU EAX, EBX
005C2B89 . BA 88395C00 MOU EDX, posurgeo.005C3988
005C2B8E . E8 31F0F8FF CALL posurgeo.00551BF4
005C2B93 . 8B55 FC MOU EDX, DWORD PTR SS:[EBP-4]
005C2B97 . BB92 74000000 MOU EDX, DWORD PTR DS:[EBP+874]
005C2B9C . 805C2B00
005C2B05 . 805C2B08
005C2B0D . 805C2B0E
005C2B0B . C740 48 4C355F00 MOU DL, 1
005C2B0E . B2 01 MOU DL, 1
005C2B0F . E8 70F0F8FF CALL posurgeo.00551C5C
005C2BEC . BB 2C010000 MOU EAX, 12C
005C2B01 . E8 E20AEEFF CALL posurgeo.004A3608
005C2B06 . 8015 8CE60000 MOU EDX, DWORD PTR DS:[60EB8C]
005C2B0F . 8000 00 MOU BYTE PTR DS:[EDX], AL
005C2B0E . A1 8CEC6000 MOU EAX, DWORD PTR DS:[60EB8C]
005C2B03 . 8098 00 CMP BYTE PTR DS:[EAX], 0
005C2B06 . 75 00 JNZ SHORT posurgeo.004A3370
005C2B08 . E8 6307EEFF CALL posurgeo.004A3370
005C2B0D . A1 8CEC6000 MOU EAX, DWORD PTR DS:[60EC8C]
005C2B12 . DD18 FSTP QWORD PTR DS:[EAX]
005C2B14 . 98 WAIT
005C2B18 > 8D45 9C LEA EAX, DWORD PTR SS:[EBP-64]
005C2C15 . 50 PUSH EAX
005C2C19 . A1 8CEC6000 MOU EAX, DWORD PTR DS:[60EC8C]
005C2C1E . DD00 FLD QWORD PTR DS:[EAX]
005C2C20 . DB70 88 FSTP TBYTE PTR SS:[EBP-78]
005C2C23 . 98 WAIT
005C2C24 . 8D45 88 LEA EAX, DWORD PTR SS:[EBP-78]
005C2C27 . 8945 94 MOU DWORD PTR SS:[EBP-6C], EAX
005C2C29 . C645 98 03 MOU BYTE PTR SS:[EBP-68], 3
005C2C31 . 8D55 94 LEA EDX, DWORD PTR SS:[EBP-60]
005C2C33 . 33C9 XOR ECX, ECX
005C2C30 . BB CC395C00 MOU EAX, posurgeo.005C39CC
005C2C38 . 805C2B00
005C2C39 . FF51 38 CALL DWORD PTR DS:[ECX+98] ...

```

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

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

Arg1 = 0012FB00

ASCII "Create: Eval %n"  
posurgeo.0040B824

ASCII "TFXPForm.File1.caption"

DS:[0060EB8C]=00610C4A (posurgeo.00610C4A)  
EDX=00000000

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

Famous pointer 의 Value 는 첫 번째로 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)

**C CPU - main thread, module pcsurgeo**

```

005C2B81 . 8998 24120000 MOV DWORD PTR DS:[EAX+1224],EBX
005C2B87 . 8BC3 MOU EAX, EBX
005C2B89 . BA 88395C00 MOU EDX, posurgeo.005C3988
005C2B8E . E8 31F0F8FF CALL posurgeo.00551BF4
005C2B93 . 8B55 FC MOU EDX, DWORD PTR SS:[EBP-4]
005C2B97 . BB92 74000000 MOU EDX, DWORD PTR DS:[EBP+874]
005C2B9C . 805C2B00
005C2B05 . 805C2B08
005C2B0D . 805C2B0E
005C2B0B . C740 48 4C355F00 MOU DL, 1
005C2B0E . B2 01 MOU DL, 1
005C2B0F . E8 70F0F8FF CALL posurgeo.00551C5C
005C2BEC . BB 2C010000 MOU EAX, 12C
005C2B01 . E8 E20AEEFF CALL posurgeo.004A3608
005C2B06 . 8015 8CE60000 MOU EDX, DWORD PTR DS:[60EB8C]
005C2B0F . 8000 00 MOU BYTE PTR DS:[EDX], AL
005C2B0E . A1 8CEC6000 MOU EAX, DWORD PTR DS:[60EB8C]
005C2B03 . 8098 00 CMP BYTE PTR DS:[EAX], 0
005C2B06 . 75 00 JNZ SHORT posurgeo.004A3370
005C2B08 . E8 6307EEFF CALL posurgeo.004A3370
005C2B0D . A1 8CEC6000 MOU EAX, DWORD PTR DS:[60EC8C]
005C2B12 . DD18 FSTP QWORD PTR DS:[EAX]
005C2B14 . 98 WAIT
005C2B18 > 8D45 9C LEA EAX, DWORD PTR SS:[EBP-64]
005C2C15 . 50 PUSH EAX
005C2C19 . A1 8CEC6000 MOU EAX, DWORD PTR DS:[60EC8C]
005C2C1E . DD00 FLD QWORD PTR DS:[EAX]
005C2C20 . DB70 88 FSTP TBYTE PTR SS:[EBP-78]
005C2C23 . 98 WAIT
005C2C24 . 8D45 88 LEA EAX, DWORD PTR SS:[EBP-78]
005C2C27 . 8945 94 MOU DWORD PTR SS:[EBP-6C], EAX
005C2C29 . C645 98 03 MOU BYTE PTR SS:[EBP-68], 3
005C2C31 . 8D55 94 LEA EDX, DWORD PTR SS:[EBP-60]
005C2C33 . BB CC395C00 MOU EAX, posurgeo.005C39CC
005C2C38 . E8 E788E4FF CALL posurgeo.0040B824
005C2C3D . 8B45 9C MOU EAX, DWORD PTR SS:[EBP-64]
005C2C40 . E8 77FBEDFF CALL posurgeo.004A278C
005C2C45 . A1 18ED6000 MOU EAX, DWORD PTR DS:[60ED18]
005C2C46 . 805C2B00 MOU EDX, DWORD PTR DS:[EBX]
005C2C47 . 805C2B00 MOU posurgeo.004FB720
005C2C48 . 805C2B00 MOU posurgeo.005C39E4
005C2C49 . 805C2B00 MOU ECX, DWORD PTR DS:[EAX]
005C2C4A . 805C2B00 MOU EDX, DWORD PTR DS:[ECX+98] ...

```

The value from AL which is zero ...

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

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

Arg1 = 0012FB00

ASCII "Create: Eval %n"  
posurgeo.0040B824

ASCII "TFXPForm.File1.caption"

AL=00  
DS:[00610C4A]=00

The value from AL which is zero ...

AL 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.

C CPU - main thread, module pcsurgeo

```
005C2B81  . 8998 24120000 MOU DWORD PTR DS:[EAX+1224],EBX
005C2B87  . 8BC3 MOU EAX, EBX
005C2B89  . BA B8395C00 MOU EDX, posurgeo.005C3988
005C2B8E  . E8 31F0F8FF CALL posurgeo.00551BF4
005C2B93  . 8B55 FC MOU EDX, DWORD PTR SS:[EBP-4]
005C2B96  . 8B92 740B0000 MOU EDX, DWORD PTR DS:[EAX+B74]
005C2BCC  . 8B45 FC MOU EAX, DWORD PTR SS:[EBP-4]
005C2BDF  . 8B88 24120000 MOU EAX, DWORD PTR DS:[EAX+1224]
005C2BD5  . 8950 40 MOU EDX, DWORD PTR DS:[EAX+40],EDX
005C2BDB  . 8B55 FC MOU EDX, DWORD PTR SS:[EBP-4]
005C2BDB  . 8950 4C MOU EDX, DWORD PTR DS:[EAX+4C],EDX
005C2BDE  . C740 48 4C355F00 MOU EDX, DWORD PTR DS:[EAX+4C]
005C2BE5  . B2 01 MOU DL,1
005C2BE7  . E8 70F0F8FF CALL posurgeo.
005C2BEC  . B8 2C010000 MOU EAX, 12C
005C2BF1  . E8 E20EEEFF CALL posurgeo.004A93608
005C2BFD  . BB15 8CEB6000 MOU EDX, DWORD PTR DS:[E0EB8C]
005C2BFC  . 8B02 MOU BYTE PTR DS:[ECX],AL
005C2BFE  . A1 8CEC6000 MOU EAX, DWORD PTR DS:[E0EB8C]
005C2C03  . 8038 00 CMP BYTE PTR DS:[EAX],0
005C2C06  . V7 0D JNZ SHORT posurgeo.005C2C15
005C2C08  . E8 6307EEFF CALL posurgeo.004A93370
005C2C0D  . A1 8CEC6000 MOU EAX, DWORD PTR DS:[E0EC8C]
005C2C12  . DD18 FSTP QWORD PTR DS:[EAX]
005C2C14  . 9B WAIT
005C2C15  > 8D45 9C LEA EAX, DWORD PTR SS:[EBP-64]
005C2C18  . 50 PUSH EAX
005C2C19  . A1 8CEC6000 MOU EAX, DWORD PTR DS:[E0EC8C]
005C2C1E  . D000 FLD QWORD PTR DS:[EAX]
005C2C20  . DB7D 88 FSTP TBVTE PTR SS:[EBP-78]
005C2C23  . 9B WAIT
005C2C24  . 8D45 88 LEA EAX, DWORD PTR SS:[EBP-78]
005C2C27  . 8945 94 MOU EDX, DWORD PTR SS:[EBP-6C],EAX
005C2C2H  . C645 98 03 MOU BYTE PTR SS:[EBP-68],3
005C2C2E  . 8D55 94 LEA EDX, DWORD PTR SS:[EBP-6C]
005C2C31  . 83C9 XOR ECX,ECX
005C2C33  . B8 CC395C00 MOU EAX,posurgeo.005C3900
005C2C38  . E8 E788E4FF CALL posurgeo.004A93824
005C2C3D  . 8B45 9C MOU EAX, DWORD PTR SS:[EBP-64]
005C2C40  . E8 77F8EDFF CALL posurgeo.004A9278C
005C2C45  . A1 18ED6000 MOU EAX, DWORD PTR DS:[E0ED18]
005C2C4H  . 8B00 MOU EDX, DWORD PTR DS:[EAX]
005C2C4C  . E8 DB8AF3FF CALL posurgeo.004F72C
005C2C51  . BA E4395C00 MOU EAX, posurgeo.005C39E4
005C2C56  . 8B08 MOU ECX, DWORD PTR DS:[EAX]
005C2C59  . FF51 38 CALL DWORD PTR DS:[ECX+98]
005C2C59  . FF51 38
```

The pointer is copied in EAX

Arg1 = 0012FB00

DS:[0060EB8C]=00610C4A  
EAX=0012FB00

ASCII "PC Surgeon"

posurgeo.00720063  
posurgeo.00610C4A  
posurgeo.00610C4A

ASCII "Create: Eval %n"

posurgeo.00408824

ASCII "TFXPForm.File1.caption"

The pointer is copied in EAX

Pointer 값이 EAX 로 복사됐다.

C CPU - main thread, module pcsurgeo

```
005C2B61 . 8998 24120000 MOV DWORD PTR DS:[EAX+1224],EBX
005C2B67 . 8BC3 MOV EAX,EBX
005C2B69 . BA B8995C00 MOV EDX, posurgeo.005C3988
005C2B6E E8 31F0FBFF CALL posurgeo.00551BF4
005C2B73 . 8B55 FC MOV EDX, DWORD PTR SS:[EBP-4]
005C2B76 . B892 740B0000 MOV EDX, DWORD PTR DS1[EDX+B74]
005C2BCC . 8B45 FC MOV EDX, DWORD PTR SS:[EBP-4]
005C2B7F . B8B0 24120000 MOV EAX, DWORD PTR DS:[EAX+1224]
005C2B85 . 8958 48 MOV DWORD PTR DS:[EAX+48],EDX
005C2B88 . 8B55 FC MOV EDX, DWORD PTR SS:[EBP-4]
005C2B8D . 8958 4C MOV DWORD PTR DS:[EAX+4C],EDX
005C2B90 . C748 48 4C355F00 MOV DWORD PTR DS:[EAX+48],pcsurgeo.00610C4A
005C2B9E . B2 01 MOV DL,1
005C2B9F . E8 70F0FBFF CALL posurgeo.00551C5C
005C2BCE . B8 2C010000 MOV EAX,12C
005C2BFD . E8 E204EFFE CALL posurgeo.004A3608
005C2BF6 . BB15 8C6E6000 MOV EDX, DWORD PTR DS:[60EB8C]
005C2BF9 . 8802 MOV BYTE PTR DS:[EAX],AL
005C2BFA . A1 8C6B6000 MOV EAX,DWORD PTR DS:[66EB8C]
005C2BFB . 8933 00 CMP BYTE PTR DS:[EAX],0
005C2C06 . 75 00 JNC SHORT posurgeo.005C2C15
005C2C08 . E8 6307EEFF CALL posurgeo.004A3376
005C2C0D . A1 8C6C6000 MOV EAX,DWORD PTR DS:[60EC8C]
005C2C12 . DD18 FSTP QWORD PTR DS:[EAX]
005C2C14 . 9B HLT
005C2C15 . 8D45 50
005C2C18 . 50
005C2C19 . A1 00
005C2C1E . D000
005C2C20 . DB70
005C2C23 . 9B WHI1
005C2C24 . 8D45 88 LEA EAX,DWORD PTR SS:[EBP-78]
005C2C27 . 8945 94 MOV DWORD PTR SS:[EBP-6C],ERX
005C2C2A . C645 98 03 MOV BYTE PTR SS:[EBP-68],3
005C2C2E . 8055 94 LEA EDX,DWORD PTR SS:[EBP-6C]
005C2C31 . 39C9 XOR ECX,ECX
005C2C33 . BB CC995C00 MOV EAX, posurgeo.005C39CC
005C2C38 . E8 E78BE4FF CALL posurgeo.004B8824
005C2C3D . 8B45 9C MOV EDX, DWORD PTR SS:[EBP-64]
005C2C40 . E8 77FEBDEF CALL posurgeo.004A27B8C
005C2C45 . A1 8ED6000 MOV EAX,DWORD PTR DS:[60ED18]
005C2C48 . 8800 MOV EAX,DWORD PTR DS:[EAX]
005C2C4C . E8 DB8AF3FF CALL posurgeo.004FB120
005C2C51 . BB E4395C00 MOV EDX, posurgeo.005C39E4
005C2C56 . 8B85 HOU ECA,DWORD PTR DS:[EAX]
005C2C58 . FFS1 38 CALL DWORD PTR DS:[EAX+98] ...
```

ASCII "PC Surgeon"

posurgeo.00610C4A

posurgeo.00610C4A

posurgeo.00610C4A

And its value is compared to zero

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

ASCII "Create: Eval %n"

posurgeo.004B8824

ASCII "TFXPForm.File1.caption"

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(AL이 zero) 우리는 다음 step으로 jump 할 수 없다.

C CPU - main thread, module pcsurgeo

005C2B81 . 8998 24120000 MOV DWORD PTR [EAX], ECX  
005C2B82 . 8B83 . MOU EDX, ECX  
005C2B89 . BA 8395C00 MOU EDX, pcursurgo  
005C2B8E . E8 31F0FF CALL pcursurgo  
005C2B93 . BB55 FC MOU EDX, 00  
005C2B96 . BB92 74000000 MOV EDX, DQ  
005C2BC0 . BB45 FC MOV EDX, DQ  
005C2BCF . BB80 24120000 MOU EDX, DQ  
005C2BD5 . B950 40 MOU EDX, DQ  
005C2BDE . BB55 FC MOU EDX, C0  
005C2BDB . B950 40 MOU EDX, DQ  
005C2BDE . C748 48 4C355F00 MOU EDX, DQ  
005C2BE5 . B9 91 00 00 00 MOU EDX, DQ  
005C2BEE . B8 70F0FF CALL pcursurgo  
005C2BEC . B8 2C010000 MOU EDX, 12C  
005C2BED . B8 220AEFF CALL pcursurgo, 004A96D0  
005C2BEC1 . BB15 8CE60000 MOU EDX, DQ  
005C2BEC6 . B8 03 00 00 00 MOU EDX, DQ  
005C2BEC7 . A1 SCEB6000 MOU EDX, DQ  
005C2BEC8 . B8 03 00 00 00 CMP BYTE PTR [EDX], ECX  
005C2BEC9 . 75 0D JNZ SHORT pcursurgo  
005C2BECB . E8 6307EFF CALL pcursurgo  
005C2BEC0 . A1 8CEC6000 MOU EDX, DQ  
005C2BEC0 . DD18 FSTP QWORD PTR [EDX]  
005C2BEC1 . 9B WAIT  
005C2BEC1 . > 0D45 9C LEA EAX, DQWORD  
005C2BEC1 . 50 PUSH EAX  
005C2BEC1 . A1 8CEC6000 MOU EDX, DQ  
005C2BEC1 . DD08 FLD QWORD PTR [EDX]  
005C2BEC0 . DB70 88 FSTP TBYTE PTR [EDX]  
005C2BEC1 . 9B WAIT  
005C2BEC1 . 8D45 88 LEA EAX, DQWORD  
005C2BEC1 . 8945 94 MOU EDX, DQWORD PTR [EDX]  
005C2BEC1 . C645 98 03 MOV BTYE PTR [EDX], ECX  
005C2BEC1 . 8D55 94 LEA EDX, DQWORD  
005C2BEC1 . 33C9 XOR ECX, ECX  
005C2BEC1 . B8 CC395C00 MOU EDX, pcursurgo  
005C2BEC1 . E8 E788E4FF CALL pcursurgo  
005C2BEC1 . 8B45 9C MOU EDX, DQWORD PTR [EDX]  
005C2BEC1 . E8 77FBE0FF CALL pcursurgo  
005C2BEC1 . A1 18ED6000 MOU EDX, DQWORD PTR [EDX]  
005C2BEC1 . 8B00 MOU EDX, DQWORD PTR [EDX]  
005C2BEC1 . C748 004FEC20 CALL pcursurgo, 004FEC20  
005C2BEC1 . B8 E4395C00 MOU EDX, DQWORD PTR DS:[EAX]  
005C2BEC1 . 8B00 MOU EDX, DQWORD PTR DS:[ECX+98]  
005C2BEC1 . FF51 38 CALL DQWORD PTR DS:[ECX+98]  

Hence, let's also do that! As always, there are many possibilities for assembling this. 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).

Is it 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).

INFO:

Completely correct would be "By changing the byte's value to which the dword pointer's value points at". This is rather cumbersome and incomprehensible. Hence, I'm making it a little shorter. I trust all is clear though?

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)

The screenshot shows the CPU pane of Immunity Debugger with assembly code for the main thread module 'pcsurgeo'. The assembly code includes various instructions like MOVs, CALLs, and FLDs. An 'Assemble at' dialog box is open at address 005C2C06, containing the instruction 'mov BYTE PTR DS:[EAX],1'. A checkbox labeled 'Fill with NOP's' is checked. The arguments section shows 'Arg1 = 00610C4A'. Below the assembly pane, a note says 'Jump is NOT taken 005C2C15=pcsurgeo.005C2C15 ;)'.

```

CPU - main thread, module pcsurgeo
005C2B61: E9 98 24120000 MOV DWORD PTR DS:[EBX+1224], EBX
005C2B62: B8 03 MOU EAX, EBX
005C2B63: E9 31F0F9FF MOU EDX, pcosurgeo.005C29B8
005C2B64: C9 CALL pcosurgeo.00551BF4
005C2B65: B8 55 FC MOU EDX, DWORD PTR SS:[EBP-4]
005C2B66: E9 92 740B0000 MOU EDX, DWORD PTR DS:[EDX+B74]
005C2B67: B8 45 FC MOU EAX, DWORD PTR SS:[EBP-4]
005C2B68: E9 88 24120000 MOU EAX, DWORD PTR DS:[EBX+1224]
005C2B69: B8 90 40 MOU EDX, DWORD PTR DS:[EBX+40], EDX
005C2B6A: B8 55 FC MOU EDX, DWORD PTR SS:[EBP-4]
005C2B6B: B8 90 4C MOU EDX, DWORD PTR DS:[EBX+4C], EDX
005C2B6C: C740 48 4C355F00 MOU EDX, DWORD PTR DS:[EBX+48], pcosurgeo.00551C5C
005C2B6D: B8 01 MOU DL, 1
005C2B6E: E9 79F0F9FF MOU EAX, 12C
005C2B6F: B8 20010000 MOU EAX, DWORD PTR DS:[ECX], AL
005C2B70: E9 E20EEFF MOU EAX, DWORD PTR DS:[EBX+8C]
005C2B71: B8 15 8CE6000 MOU EAX, DWORD PTR DS:[EBX+8C]
005C2B72: C9 C0 MOU EDX, pcosurgeo.004A3608
005C2B73: B8 02 00 MOU BYTE PTR DS:[ECX], AL
005C2B74: C9 C0 MOU EAX, DWORD PTR DS:[EBX+8C]
005C2B75: C9 C0 MOU EAX, BYTE PTR DS:[ECX], 1
005C2B76: C9 C0 JNZ SHORT pcosurgeo.005C2C15
005C2B77: E9 6307EFF CALL pcosurgeo.004A3370
005C2B78: A1 8CEC6000 MOU EAX, DWORD PTR DS:[60EC8C]
005C2B79: D018 FSTP QWORD PTR DS:[EAX]
005C2B7A: 9B WAIT
005C2B7B: >8D45 9C LEA EAX, DWORD PTR SS:[EBP-64]
005C2B7C: 50 PUSH EAX
005C2B7D: A1 8CEC6000 MOU EAX, DWORD PTR DS:[60EC8C]
005C2B7E: DD00 FLD QWORD PTR DS:[EAX]
005C2B7F: DB70 88 FSTP TBYTE PTR SS:[EBP-78]
005C2B80: 9B WAIT
005C2B81: 8D45 88 LEA EAX, DWORD PTR SS:[EBP-78]
005C2B82: 8945 94 MOU DWORD PTR SS:[EBP-6C], EAX
005C2B83: E645 98 03 MOU BYTE PTR SS:[EBP-68], 3
005C2B84: 8D55 94 LEA EDX, DWORD PTR SS:[EBP-6C]
005C2B85: 83C9 XOR ECX, ECX
005C2B86: B8 CC395C00 MOU EAX, pcosurgeo.005C29C0
005C2B87: E9 E78BE4FF MOU EAX, pcosurgeo.00408824
005C2B88: B8 45 9C MOU EAX, DWORD PTR SS:[EBP-64]
005C2B89: E9 77FBE0FF MOU EAX, pcosurgeo.004A27BC
005C2B8A: B8 00 00 MOU EAX, DWORD PTR DS:[60ED18]
005C2B8B: C9 C0 MOU EAX, DWORD PTR DS:[EBX]
005C2B8C: E9 DB80F3FF MOU EAX, pcosurgeo.004F8720
005C2B8D: B8 E4395C00 MOU EAX, pcosurgeo.005C29E4
005C2B8E: B8 08 MOU ECX, DWORD PTR DS:[EAX]
005C2B8F: FF51 39 CALL DWORD PTR DS:[ECX+98]

```

ASCII "PC Surgeon"  
**Assemble at 005C2C06**  
 mov BYTE PTR DS:[EAX],1  
 Fill with NOP's  
 Assemble Cancel  
 Arg1 = 00610C4A  
 pcosurgeo.00610C4A  
 ASCII "Create: Eval %n"  
 pcosurgeo.00408824  
 ASCII "TFXPForm.File1.caption"  
 ASCII "TFXPForm.File1.caption"

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.

C CPU - main thread, module posurgeo

005C2B81 : E998 24120000 MOU DWORD PTR DS:[EAX+1224],EBX  
005C2B87 : EBC3 MOU EAX,EBX  
005C2B89 : EB 88395C00 MOU EDX, posurgeo.005C29B8  
005C2B8E : E8 31F0F9FF CALL posurgeo.00551FB4  
005C2B93 : E855 FC MOU EDX, DMORD PTR SS:[EBP-4]  
005C2B96 : E892 740B0000 MOU EDX, DMORD PTR DS:[EDX+B74]  
005C2B9C : E845 FC MOU EAX, DMORD PTR SS:[EBP-4]  
005C2B9F : E880 24120000 MOU EAX, DMORD PTR DS:[EAX+1224]  
005C2B95 : E858 40 MOU DWORD PTR DS:[EAX+40],EDX  
005C2B98 : E855 FC MOU EDX, DMORD PTR SS:[EBP-4]  
005C2B9D : E850 4C MOU DWORD PTR DS:[EAX+4C],EDX  
005C2B9E : C740 48 4C355F00 MOU DWORD PTR DS:[EAX+48],posurgeo  
005C2B95 : E821 01 MOU DL,1  
005C2B97 : E8 70F0F9FF CALL posurgeo.00551C5C  
005C2BEC : B8 20010000 MOU EAX,12C  
005C2BF1 : E8 E20EEEFF CALL posurgeo.004A3608  
005C2BF6 : 8815 8CEB6000 MOU EDX, DMORD PTR DS:[60EB8C1]  
005C2BFC : 8802 MOU BYTE PTR DS:[EDX],AL  
005C2BFE : A1 8CE66000 MOU EAX, DMORD PTR DS:[60EB8C1]  
005C2C03 : C600 01 MOU BYTE PTR DS:[EAX],1  
005C2C06 : C7 40 01 JNZ SHORT posurgeo.005C2C15  
005C2C08 : E8 63B7EEFF CALL posurgeo.004A3370  
005C2C0D : A1 8CEC6000 MOU EAX, DMORD PTR DS:[60EC8C1]  
005C2C12 : DB12 FSTP QWORD PTR DS:[EAX]  
005C2C14 : 9B WAIT  
005C2C15 : >8045 9C LEA EAX, DMORD PTR SS:[EBP-64]  
005C2C18 : 50 PUSH EAX  
005C2C19 : A1  
005C2C1E : D  
005C2C20 : D  
005C2C23 : D  
005C2C24 : D  
005C2C27 : D  
005C2C2H : D  
005C2C2E : 80  
005C2C31 : 33C9 MOV ECX,ECX  
005C2C33 : B8 CC395C00 MOU EAX, posurgeo.005C39C0  
005C2C38 : E8 E786E4FF CALL posurgeo.0040B824  
005C2C40 : 8845 9C MOU EAX, DMORD PTR SS:[EBP-64]  
005C2C48 : E8 77FBEDFF CALL posurgeo.004A27B0  
005C2C45 : A1 1SED6000 MOU EAX, DMORD PTR DS:[60ED18]  
005C2C44 : 800000 MOU EAX, DMORD PTR DS:[EAX]  
005C2C4C : E8 0B80F3FF CALL posurgeo.004FB72C  
005C2C51 : EB E4395C00 MOU EDX, DMORD PTR DS:[EAX+9E4]  
005C2C56 : 8808 MOU ECX, DMORD PTR DS:[EAX]  
005C2C58 : FF51 38 CALL DMORD PTR DS:[ECX+98]  
  
But because there is no compare anymore,  
we don't jump though the pointer's value is  
set to 1. So, let's make the small correction.  
  
ASCII "PC Surgeon"  
posurgeo.00610C4A  
posurgeo.00610C4A  
  
Arg1 = 00610C4A  
  
ASCII "Create: Eval %n"  
posurgeo.0040B824  
  
ASCII "TFXPForm.File1.caption"  
  
Jump is NOT taken  
005C2C15=posurgeo.005C2C15

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 1로 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.

C CPU - main thread, module pcsurgeo

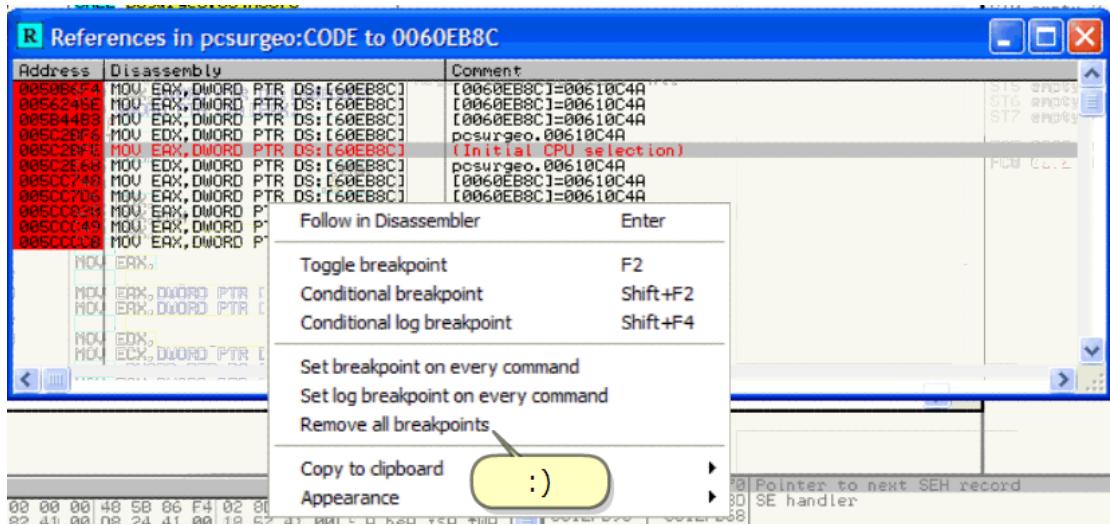
```

005C2B81  . 8998 24120000 MOU DWORD PTR DS:[EAX+1224],EBX
005C2B87  . 8BC3 MOU EDX, EBX
005C2B89  . BA 8895C00 MOU EDX, posurgeo.005C3988
005C2B8E  . E8 81F0F8FF CALL posurgeo.00551B74
005C2B93  . 8B55 FC MOU EDX, DWORD PTR SS:[EBP-1]
005C2B96  . 8B93 74000000 MOU EDX, QWORD PTR DS:[EDX+B74]
005C2B9D  . 8B45 FC MOU EDX, QWORD PTR SS:[EBP-1]
005C2B9F  . 8B00 24120000 MOU EDX, QWORD PTR DS:[EAX+1224]
005C2B05  . 8958 40 MOU EDX, DWORD PTR DS:[EAX+40],EDX
005C2B08  . 8B55 FC MOU EDX, QWORD PTR SS:[EBP-1]
005C2B0B  . 8958 40 MOU DWORD PTR DS:[EAX+40]
005C2B0E  . C740 48 4C355F00 MOU DWORD PTR DS:[EAX+40]
005C2B05  . B2 81 DL,1
005C2B87  . E8 70F0F8FF CALL posurgeo.00551C5C
005C2BBC  . B8 2C010000 MOU EAX, 12C
005C2B81  . E8 E204EEFF CALL posurgeo.004A9608
005C2B86  . 8B15 8CE66000 MOU EDX, DWORD PTR DS:[60EB8C]
005C2B8C  . 8B02 00 MOU BYTE PTR DS:[EDX],AL
005C2B8E  . A1 8CEC6000 MOU EAX, DWORD PTR DS:[60EB8C]
005C2B83  . C600 01 MOU BYTE PTR DS:[EAX],1
005C2B06  . EB 00 JMP SHORT, posurgeo.005C2C15
005C2C08  . E8 6307EEFF CALL posurgeo.004A9379
005C2C0D  . A1 8CEC6000 MOU EAX, DWORD PTR DS:[60EC8C]
005C2C12  . DD18 FSTP QWORD PTR DS:[EAX]
005C2C14  . 9B WAIT
005C2C15  > 8D45 9C LEA EAX, DWORD PTR SS:[EBP-64]
005C2C18  . 5B PUSH EAX
005C2C19  . A1 8CEC6000 MOU EAX, DWORD PTR DS:[60EC8C]
005C2C1E  . DD00 FLD QWORD PTR DS:[EAX]
005C2C20  . DB70 88 FSTP TBYTE PTR SS:[EBP-78]
005C2C23  . 9B WAIT
005C2C24  . 8D45 88 LEA EAX, DWORD PTR SS:[EBP-78]
005C2C27  . 8945 94 MOU DWORD PTR SS:[EBP-6C],EAX
005C2C2A  . C645 98 03 MOU BYTE PTR SS:[EBP-68],3
005C2C2E  . 8D55 94 LEA EDX, DWORD PTR SS:[EBP-6C]
005C2C31  . 33C9 XOR ECX, ECX
005C2C33  . B8 CC395C00 MOU EAX, posurgeo.005C3900
005C2C38  . E8 E78BE4FF CALL posurgeo.0040B824
005C2C3D  . 8B45 9C MOU EAX, DWORD PTR SS:[EBP-64]
005C2C40  . E8 77FBEDFF CALL posurgeo.004A2780
005C2C45  . A1 18ED6000 MOU EAX, DWORD PTR DS:[60ED18]
005C2C48  . 8B00 MOU EAX, DWORD PTR DS:[EAX]
005C2C4C  . E8 DB8AF3FF CALL posurgeo.004FB720
005C2C51  . BA E4395C00 MOU EDX, posurgeo.005C39E4
005C2C56  . 8B08 MOU ECX, DWORD PTR DS:[EAX]
005C2C59  . FF51 3B CALL DWORD PTR DS:[EAX+98]...
005C2C15=pcsurgeo.005C2C15

```

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

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



:)

And test the patches...

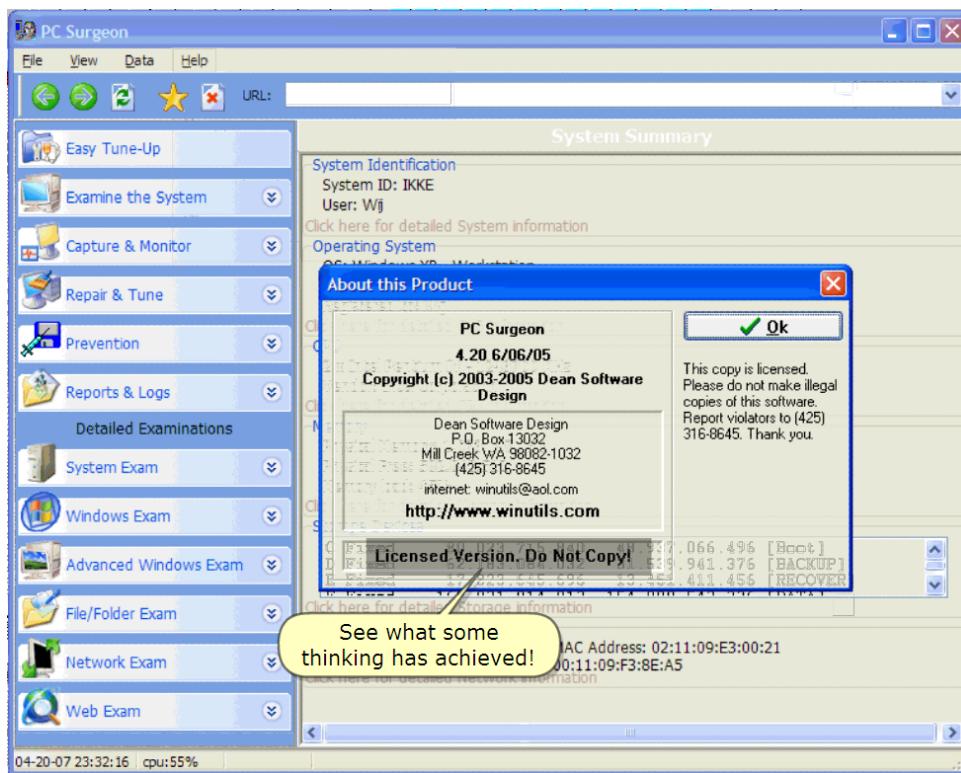
Patch 된 것을 test 하자.

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

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

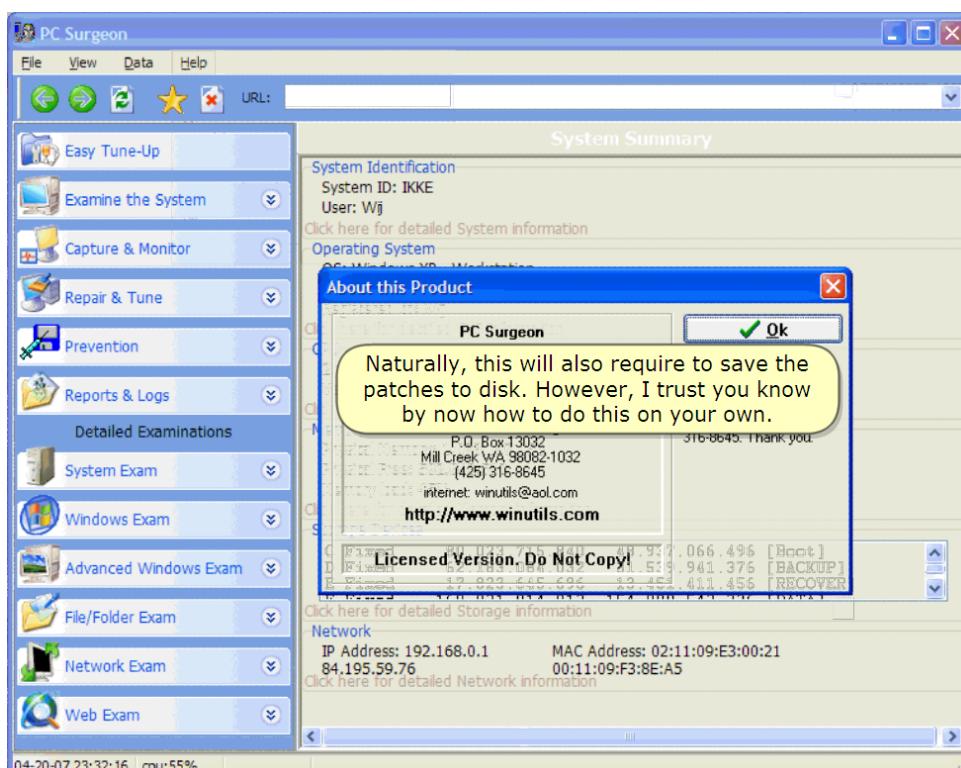
:)

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.