04.Basic+aesthetic patching

2012 년 1 월 28 일 토요일
오전 10:11

기본문

Hello everybody.
Welcome to this Part 4 in my series about reversing for newbies/beginners.
This "saga" is intended for complete starters in reversing, also for those without any programming experience at all.
Lena151 (2006)

1. Abstract

In this Part 4, we will reverse a 'real" application. That is because indeed, the best practice is found in the real applications.
As always, the goal is NOT to study this particular case but to try to teach some basic techniques. This series is purely meant to bring you the ideas for continued reversing, to give you the basics from where you can start exploring and developing your skills.
In my search not to harm anybody, I came across Pixtopianbook. This is a program not updated since 1/2001 and thus probably abandoned. This application is targeted because it is ideal for this tutorial in reversing and as such is targeted for educational purposes only. I hope you will exploit your newly acquired knowledge in a positive way.
In this matter, I also want to refer to Part 1.

이것도 똑같음
Set your screen resolution to 1152*864 and press F11 to see the movie full screen !!!
Again, I have made this movie interactive. 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. Then the movie will skip the text and continue with the next screen. If something is not clear or goes too fast, you can always use the control buttons and the slider below on this screen.
He, try it out and click on the hotspot to skip this text and to go to the next screen now !!!
Click here as soon as you finished reading (on each screen!)
During the whole movie you can click this spot to leave immediately

2. Tools and Target

이것도 똑같음

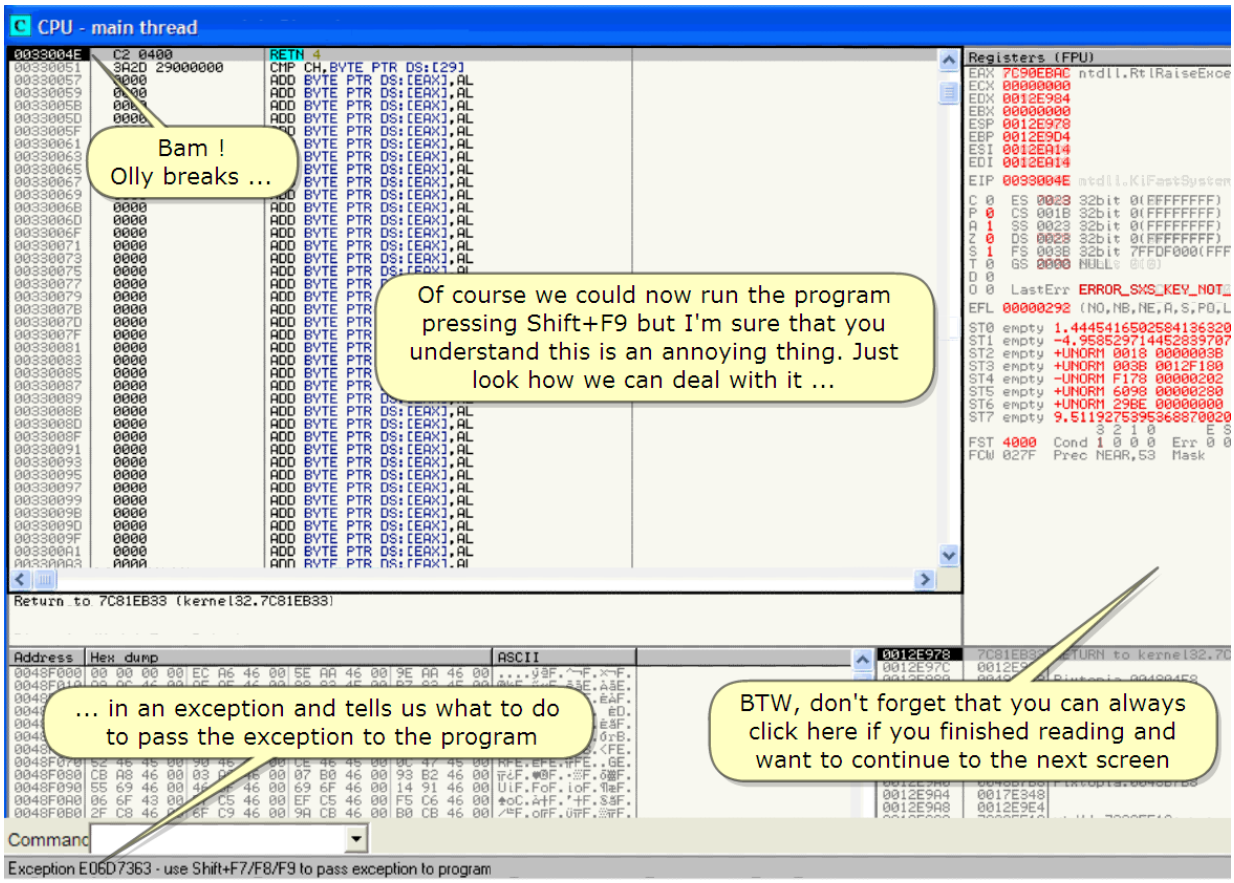
The tools for today are : Ollydebug and... your brain.
The first can be obtained for free at
http://www.ollydbg.de
Like always, the second is your responsibility ;)
Todays target is a program called Pixtopianbook v1.07. It is some kind of phonebook. Probably due to the upcoming broadband internet connections, the program seems to have set back in interest due to its dial-up connection working?
For your convenience and because the downloadsite can only be accessed using a certain browser, I included it in this package.

3. A word about exceptions

Todays application gives me the opportunity to tell you the little you need to know for now about exceptions.
오늘 application 은 나에게 약간의 네가 필요한 것을 알아가기 위해 예외들을 말할 수 있는 기회를 주었다.
The program has the SEH(structured exception handler) to handle wanted or not wanted exceptions.
Program 은 원하는 것과 원하지 않는 예외들을 handle 하기 위해 SEC 가 있다.
Olly has difficulties handling these exceptions but gives us the possibility to let the program deal with these.
Olly 는 어려운 이것들의 예외적인 handling 을 한다. 그러나 우리에게 program 과 협상 할 가능성을 준다.
So, that's what we will do for now. But for completeness, just remember for later that we use the exceptions, thrown at Olly by protectors, to easily get close to OEP.
그래서 우리는 이제 해야 한다. 그러나 완벽하기 위해, 기억해라. Olly 에서 protectors 에 의해 우리가 쓸 예외들을 OEP 를 쉽게 얻기 위해서 던진다.
So, to find OEP in an easy way. That's all for now, I will explain this better when we deal with protectors. Don't worry yet :)
그래서 OEP 를 쉬운 방법으로 찾자. 그것은 현재 모든 것이다. 우리가 protector 와 거래할 때 나는 좀 더 좋게 설명할 것이다. 아직 걱정하지마 :)
SEH are also important when we come to anti-debugging and the likes. Again, see later
우리가 anti-debugging 과 강은 것들을 할 경우 SEH 는 또한 중요하다. 다시 보자.

So, let me show you first what happens if an exception occurs... BTW, I suppose you understand an exception occurs when something is not very good, right? ;))
그래서, 나는 너에게 첫번째로 exception 이 발생할 때 무슨 일이 발생하는지 보여줄 것이다. By the way, 어떤 것은 매우 좋지 않다. 그렇죠? 네가 exception 이 발생하는 것을 이해했을 거라 생각한다.
Like you can see, I have already opened the application in Olly. Let's run and see...

너처럼 보겠다. 나는 이미 application 을 Olly 에 열어 놓았다. 실행하고 보자.



Bam! Olly breaks...

Olly 가 예외에서 멈췄다.

...in an exception and tells us what to do to pass the exception to the program

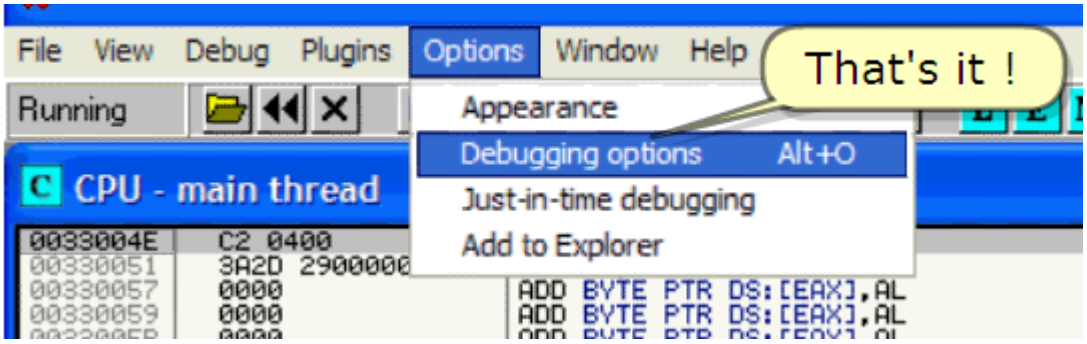
그리고 우리에게 이것이 program 에서 exception 을 pass 하기 위해 어떻게 해야 하는지 말한다.

Of course we could now run the program pressing Shift+F9 but I'm sure that you understand this is an annoying thing. Just look how we can deal with it...

물론 우리는 Shift+F9 를 눌러 program 을 실행한다. 그러나 물론 네가 짜증나는 것을 이해했을 것이라고 생각한다.

BTW, don't forget that you can always click here if you finished reading and want to continue to the next screen

By the way, 잊지 말아라,네가 읽는 것을 다 마치고 next screen 을 진행하고 싶을 때 항상 click 하라.



The exceptions can be handled under options here

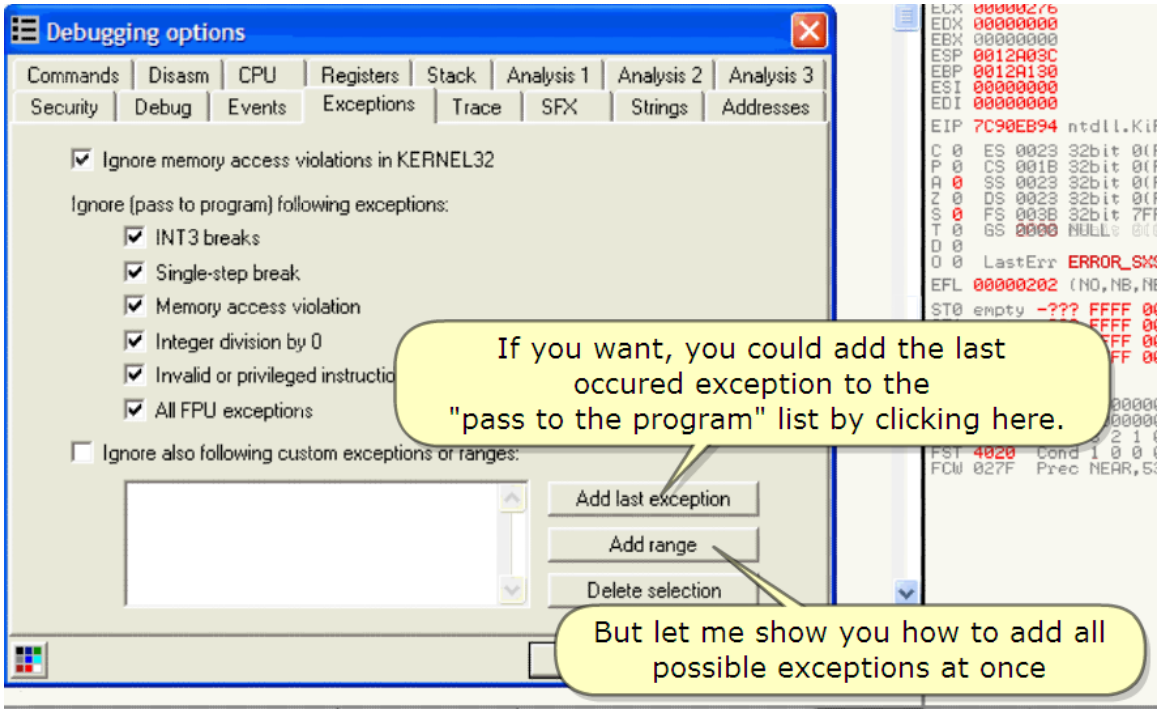
That's it

Exceptions 은 이 아래 option 에서 handle 된다.

이곳이야

Unless when dealing with protectors, leave all these checked

우리가 protector 와 거래하지 않는 한 check 를 해라.

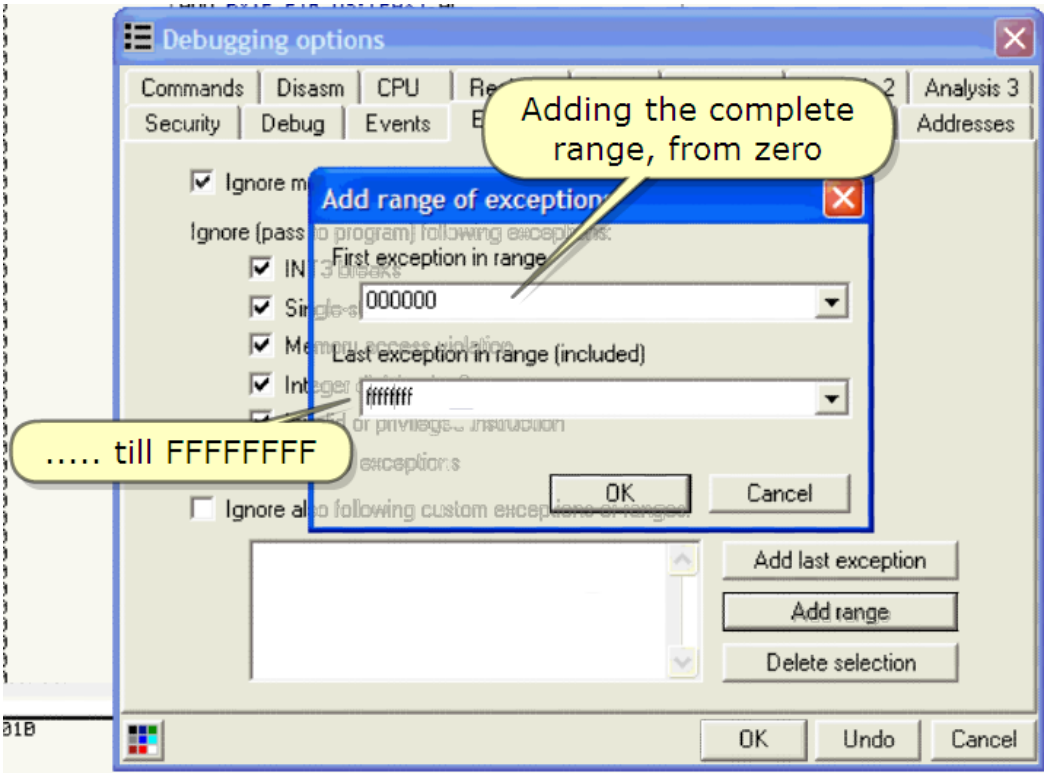


If you want, you could add the last occurred exception to the "pass to the program" list by clicking here.

만약에 네가 원한다면, 너는 마지막 exception 상황을 여기에서 click 하여 "pass to the program" list 에 더할 수 있다.

But let me show you how to add all possible exceptions at once

그러나 나는 너에게 가능한 exception 을 어떻게 추가하는지 보여준다.

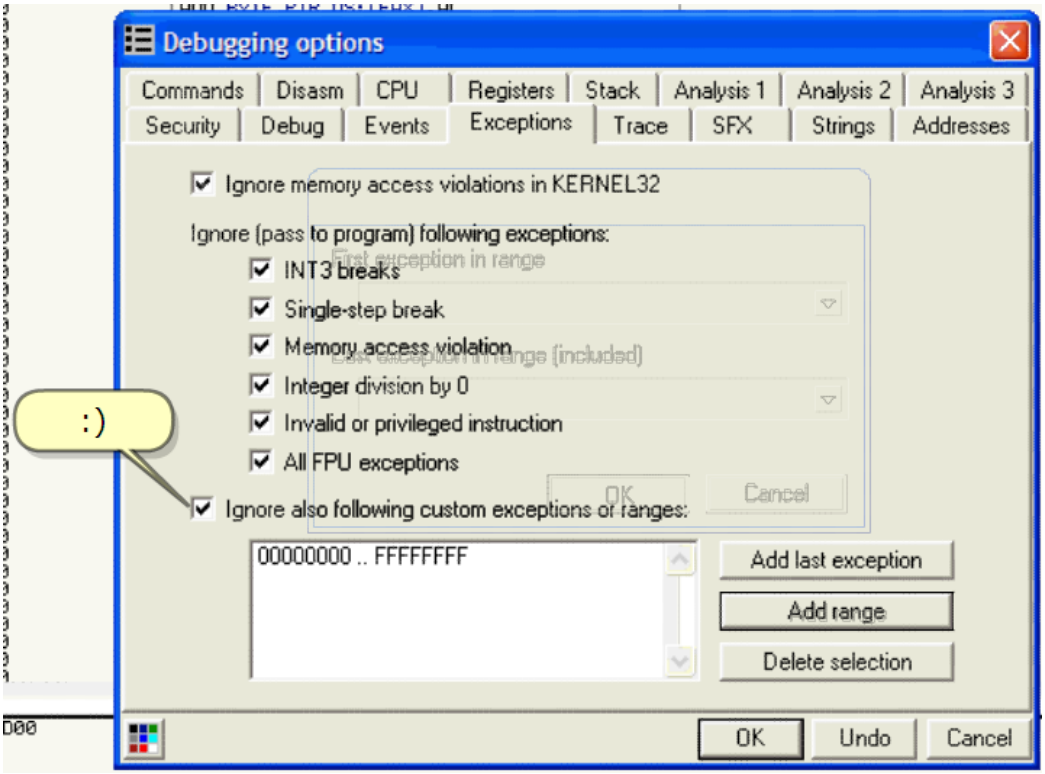


Adding the complete range, from zero

Zero 에서 완벽하게 추가했다.

00000000 ~ ffff ffff

..... Till FFFFFFFF



Ignore also following custom exceptions or ranges:

또한 조정된 exceptions 이나 범위를 무시해라.

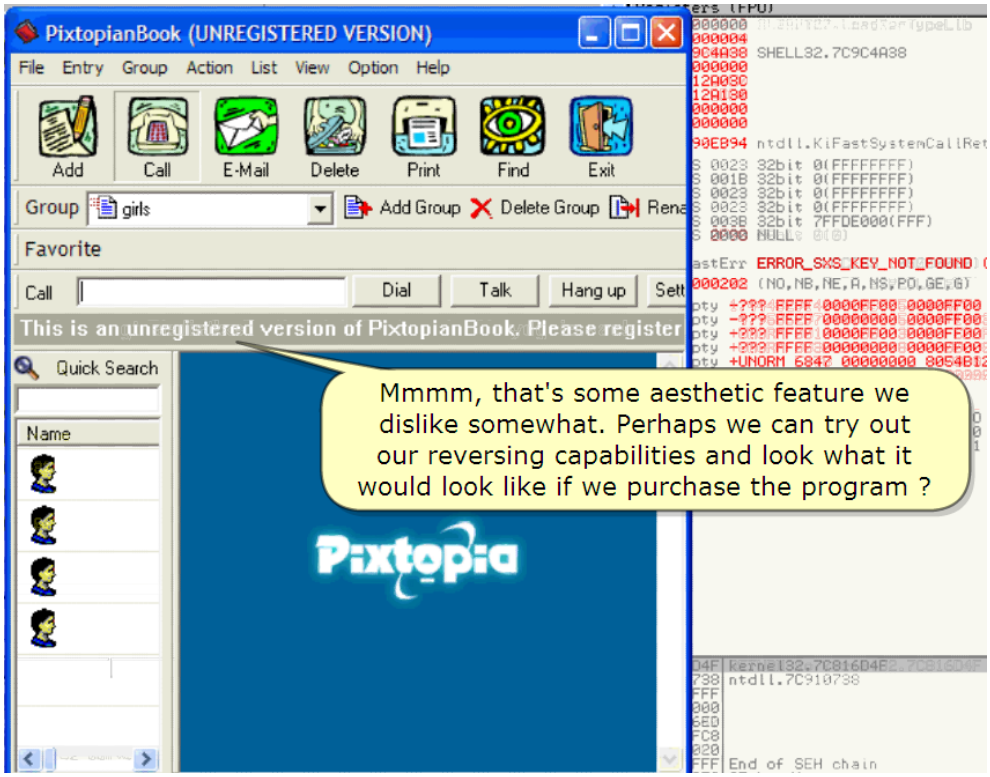
So, that will deal with the exceptions except when stepping, see later.
Now, we can run the program normally and study its behaviour. Restart.
그래서, 우리는 stepping 할 때를 제외하고 예외처리 된다. 나중에 보자.
이제, 일반적으로 program 을 실행할 수 있다. 그리고 행동을 배우자. 재시작.
번역 주) 나도 잘 이해가 안됨

4. Behaviour of the program

I told already that I can't emphasize enough the importance from the study of the target.
내가 이미 말했다. 나는 목표물의 공부에서 충분히 중요한 것을 강조할 수 없다.
It can very often give you detailed info on how to start the attack. Good research of the application can drastically reduce the time you spend on it afterwards.
이것은 매우 자주 너에게 어떻게 공격을 시작해야 할지 자세한 정보를 준다. 좋은 application 연구는 네가 나중에 보내야 할 시간을 많이 줄여준다.
So, let's do that together and run the application in Olly.
같이 해보자. Olly 에서 application 을 실행하자.

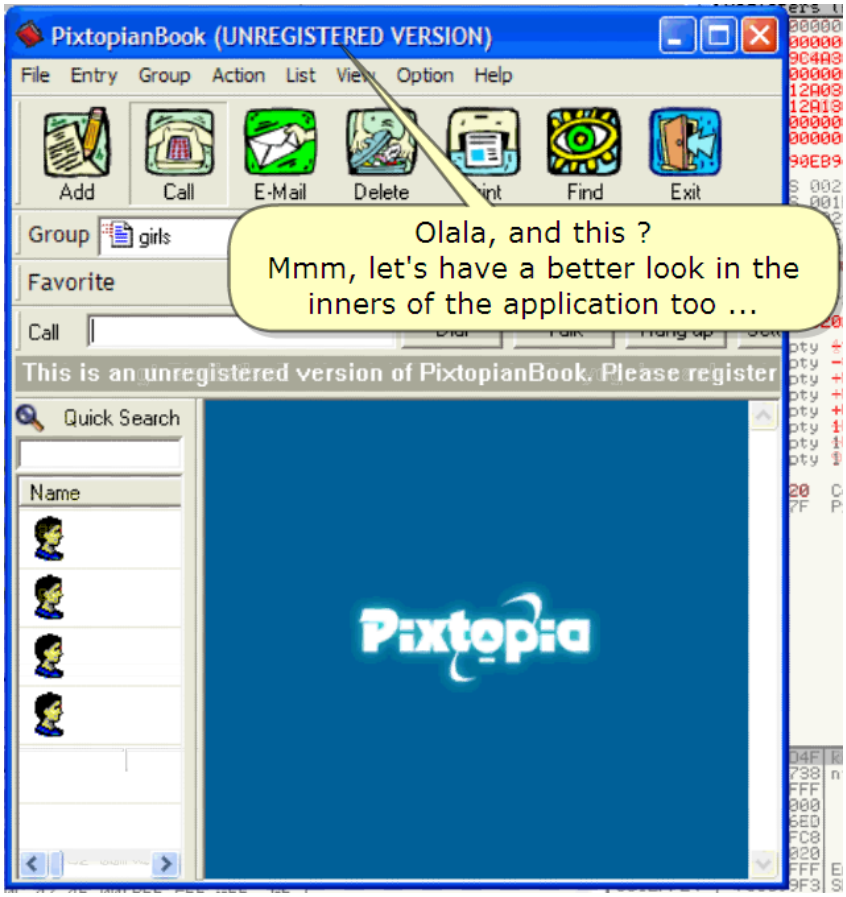
Aha, That's nice!
We get a warm welcome here!
However, after a little while ...
아하. 좋아!
우리는 따뜻한 인사를 여기에서 받았다.
그러나, 잠시 후...

Huh ???
으잉 ???



Mmmm, that's some aesthetic feature we dislike somewhat. Perhaps we can try out our reversing capabilities and look what it would look like if we purchase the program?

음, 약간 심미적인 특성은 우리가 좋아하지 않는다. 아마 우리는 reversing 능력으로 도전했다. 그리고 우리가 구매한 program 은 어떨까 생각해봐.



Olala, and this?

Mmm, let's have a better look in the inners of the application too...

얼라 그리고 이거는?

음, Application 안에서 좀 더 좋게 하자.

:)

We could make some other aesthetic glitches ...

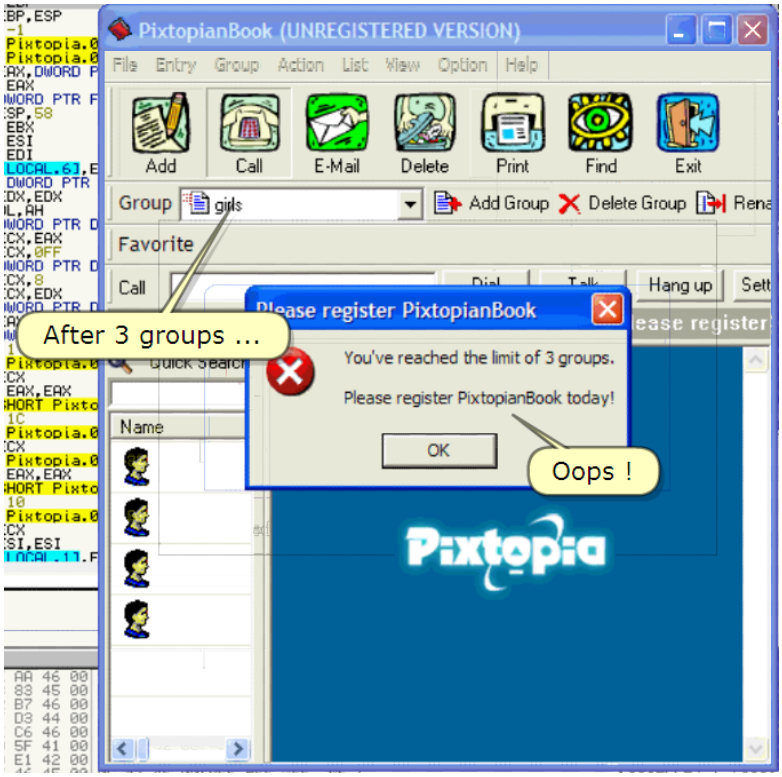
우리는 약간의 심미적인 작은 문제가 있다

Oops!

웁스!

I had already made 4 entries

나는 이미 4 entries 를 만들었다.



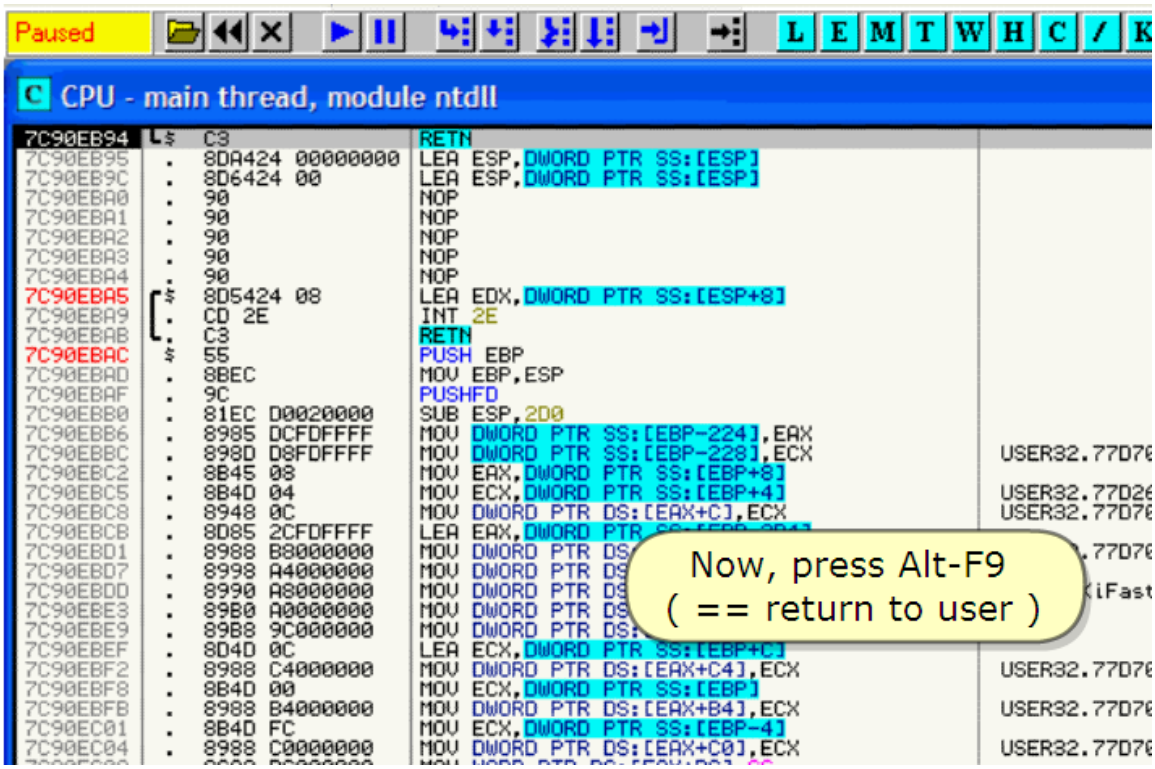
Oops !

After 3 groups
3groups 후에

Ok. So, this is a good example of a "restricted" program. It restricts the trial to 4 entries and 3 groups.
Ok. 그래서, 이것은 좋은 "제한된" 예제 program 이다. 이것은 4 명과 3 그룹에서 제한된 trial 이다.

5. Patching the "Group" restriction

Let's try to reverse the "crippled" application into the full version.
"불구된" application 를 full version 으로 reverse 하기 위해 도전하자.
How to do this? I told in previous parts that we could now grab to the string searches and try to find our way like that.
어떻게 하냐? 나는 이미 전 part 에서 이야기 했다. 우리가 문자열 검색으로 잡자. 그리고 우리들만의 방법을 찾다.
Don't ... or you will get lost as soon as you are confronted with more difficult stuff.
좀 더 어려운 문제에 직면했을 때 너는 곧바로 잃을 것이다.
There are other ways to do this that also work in i.e. encrypted files !!! (See later).
그곳에는 이것을 할 다른 방법이 있다. 또한 암호화된 file 에서 일한다 !!! (나중에 보자).
For example : here we want to know where this messagebox is created and called from.
예를 들어 : 여기 우리는 어디에서 messagebox 가 생성되는지, 어디에서 호출되는지 알기를 원한다.
Use this simple trick: pause Olly first...
간단한 trick 을 사용하자 : 먼저 Olly 를 멈춰라.



Now, press Alt-F9

(==return to user)

이제, Alt-F9 누른다.(==user 로 돌아간다)

And notice that Olly promises he will return to user as soon as you let him ;)

그리고 알린다. Olly 는 약속을 했다. 그가 곧 user 로 갈 때 알려주며 돌아간다.

Remark : if Olly would simply pause again here, then it means this trick fails in that case

주목 : 만약에 Olly 는 다시 간단히 멈추면, 이번 case 에서 그것은 trick 이 실패했다는 것이다.

Hehe, we will let him :)

헤헤, 우리는 알려준다.

Olly kept his promise !!!

He breaks right after ....

(scroll up)

Olly 는 그의 약속을 지켰다.

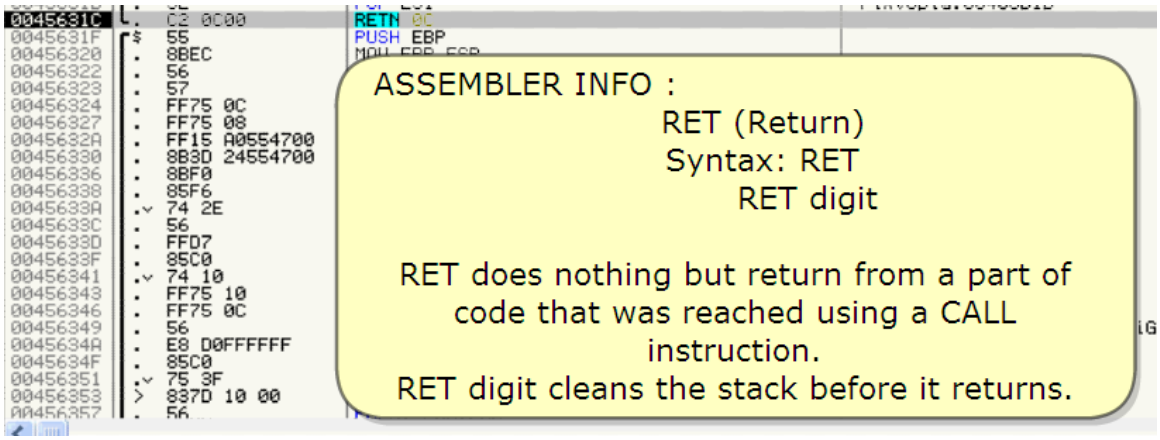
그는 실행 후에 멈췄다.

스크롤 올려봐.

... the messagebox. This little trick works in about 80% of all cases. Of course we could also have used the universal applicable API's (MessageBoxA, etc)

Messagebox. 작은 trick 은 80% 정도로 성공된다. 물론 우리는 일반적으로 해당하는 API 를 사용할 때 쓸 수 있다.(MessageBoxA, etc)

So, all that is needed now is some backtracking to see where this messagebox was called from.
Step F8
그래서, 그것이 필요해졌다. 모든 것은 추적하는 것이다. Messagebox 가 어디에서 불러지는지 보기 위해 F8 을 눌러
Aha. A command we have not discussed yet.
아하! 아직 의논하지 않았다.



ASSEMBLER INFO :

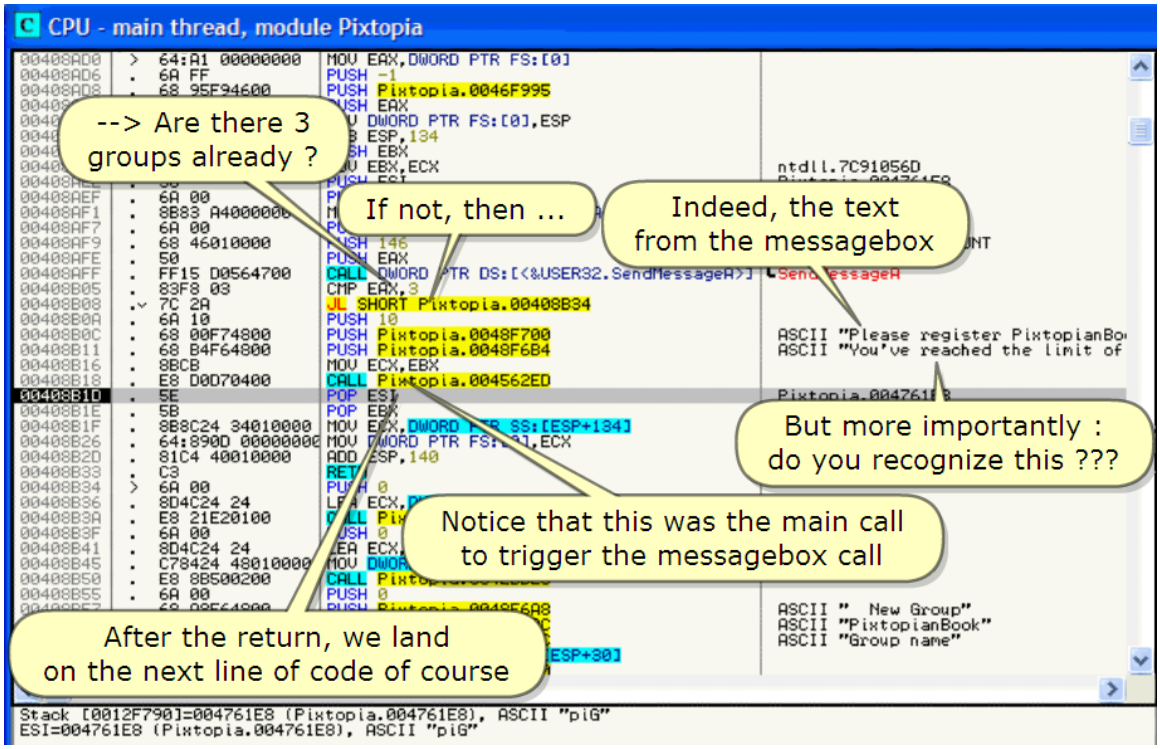
RET (Return)

Syntax: RET

RET	Digit

RET does nothing but return from a part of code that was reached using a CALL instruction. RET digit cleans the stack before it returns.
RET 는 아무것도 실행하지 않는다. 그러나 이 부분의 code 가 call 이 사용됐던 곳으로 돌려준다. 그것을 돌려준 다음에 RET 숫자는 stack 을 깨끗이 한다.

So, also step the return to.
그래서, step 을 return 한다.



Now, scroll up to see what was the caller

이제, 스크롤 올려서 caller 가 무엇인지 봐.

Notice that this was the main call to trigger the messagebox call

After the return, we land on the next line of code of course

But more importantly : do you recognize this ???

알린다. 이것은 messagebox 를 call 하기 위한 mail call 이었다.

Return 후에, 우리는 code 의 다음 line 에 도착했다.

그러나 엄청 중요하다 : 이것을 알아볼 수 있겠어?

Indeed, the text from the messagebox

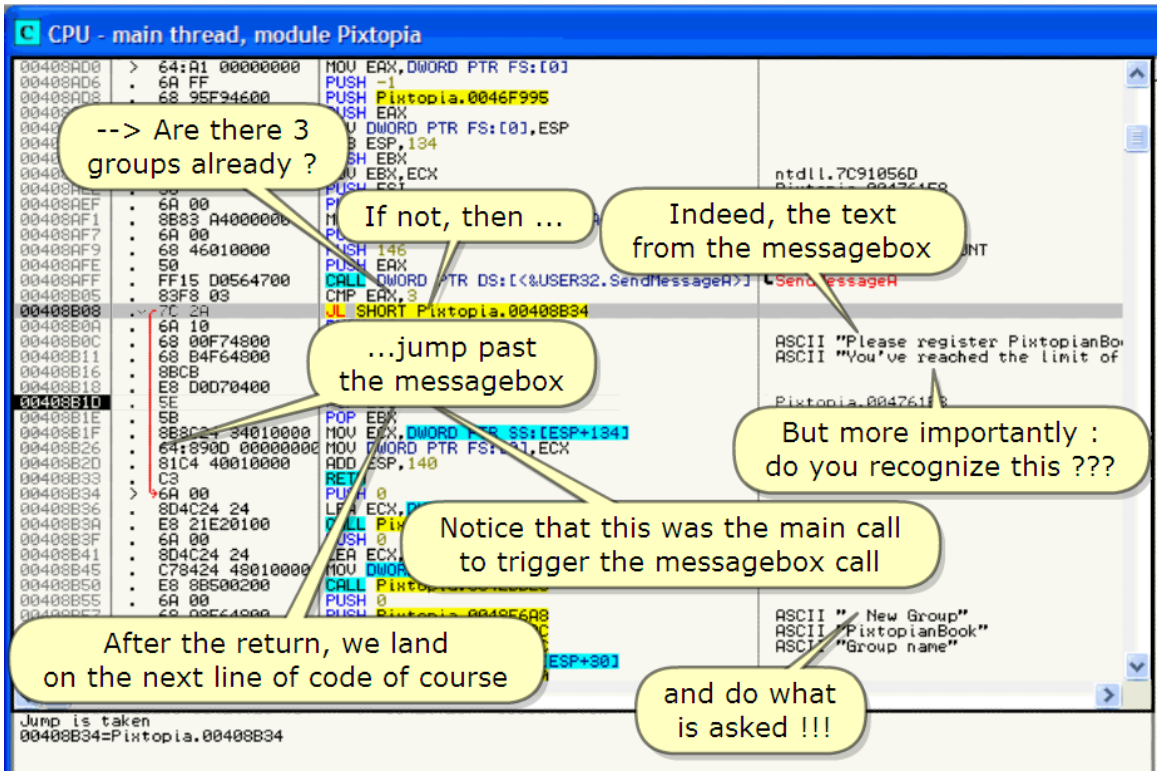
정말, messagebox 에서 나온 text 다.

--> Are there 3 groups already ?

--> 이미 3groups 비교한다.

If not, then ...

만약에 작다면,

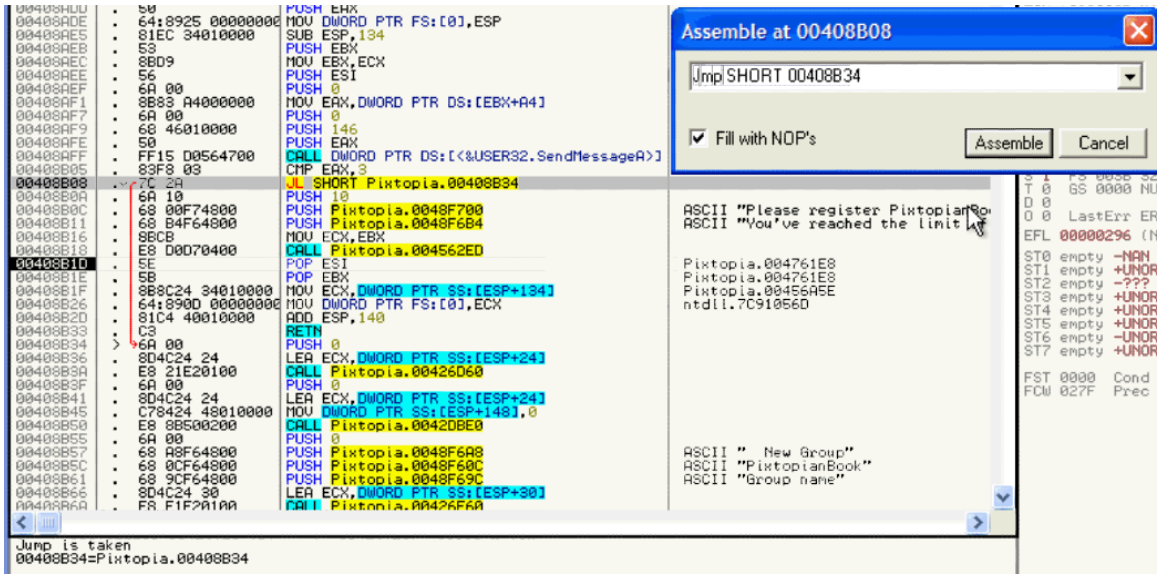


...jump past the messagebox

지나간 Messagebox 를 jump 한다.

And do what is asked !!!

그리고 무엇을 물었는지 봐라.



I trust it's clear for everybody what is needed here. Let's assemble it

나는 이것이 여기에서 필요해진 모두를 위해 명확하다고 믿는다. Assemble 하자.

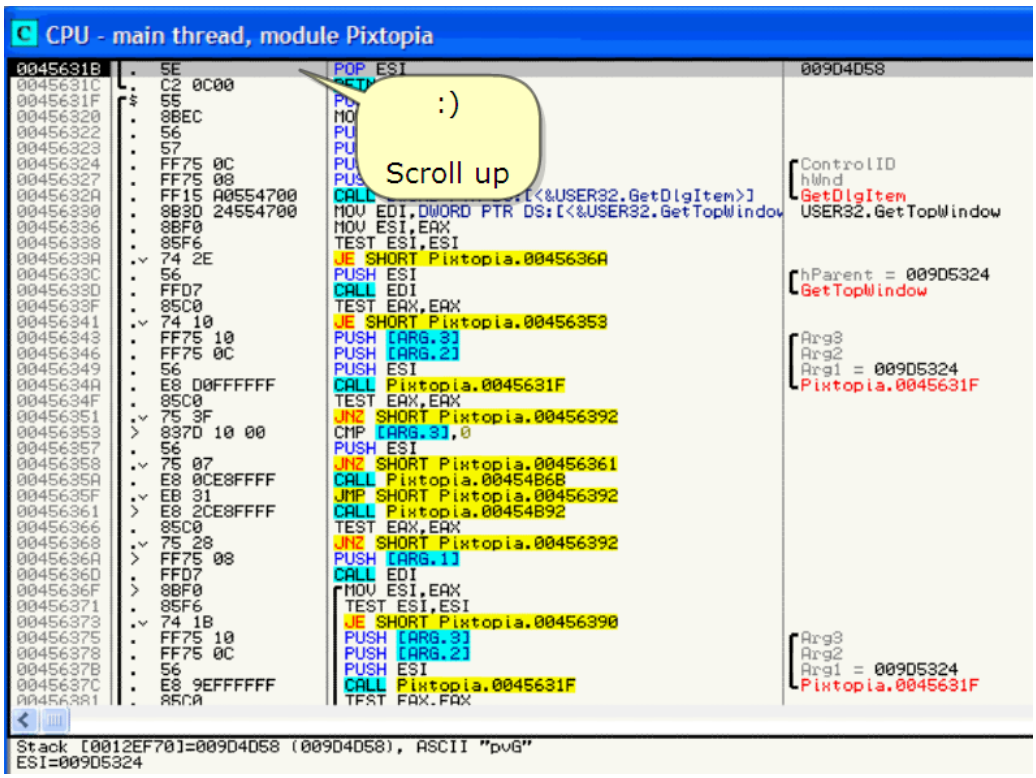
So, the JMP will always jump the messagebox and continue doing what we want!

JMP 는 항상 meesagebox 를 jump 하고 우리가 원하는 것을 계속 한다.
Now, let's see what else is needed...
이제, 무엇이 필요한지 봐라.

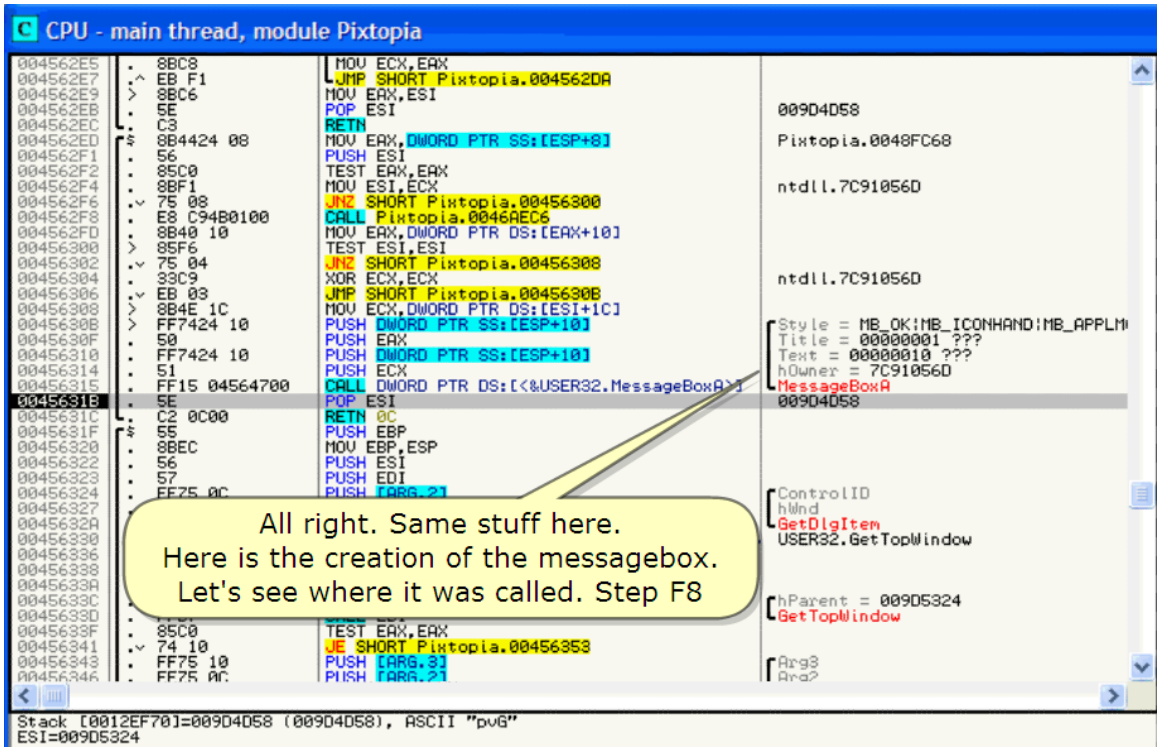
6. Patching the "Entries" restriction

Oh, yeah, we couldn't add a fifth entry
오, 우리는 fifth entry 를 추가하지 않았다.

:(
Let's do exactly the same as before to find the messagebox creation : pause Olly first. Press Alt+F9 and click the OK button ....
정확히 전과 같다. Messagebox 가 만들어진 곳을 찾기 위해 : 먼저 Olly 를 멈추자. Press Alt + F9 와 OK button 을 누르자.



:)
Scroll up
스크롤 올려.



All right. Same stuff here.

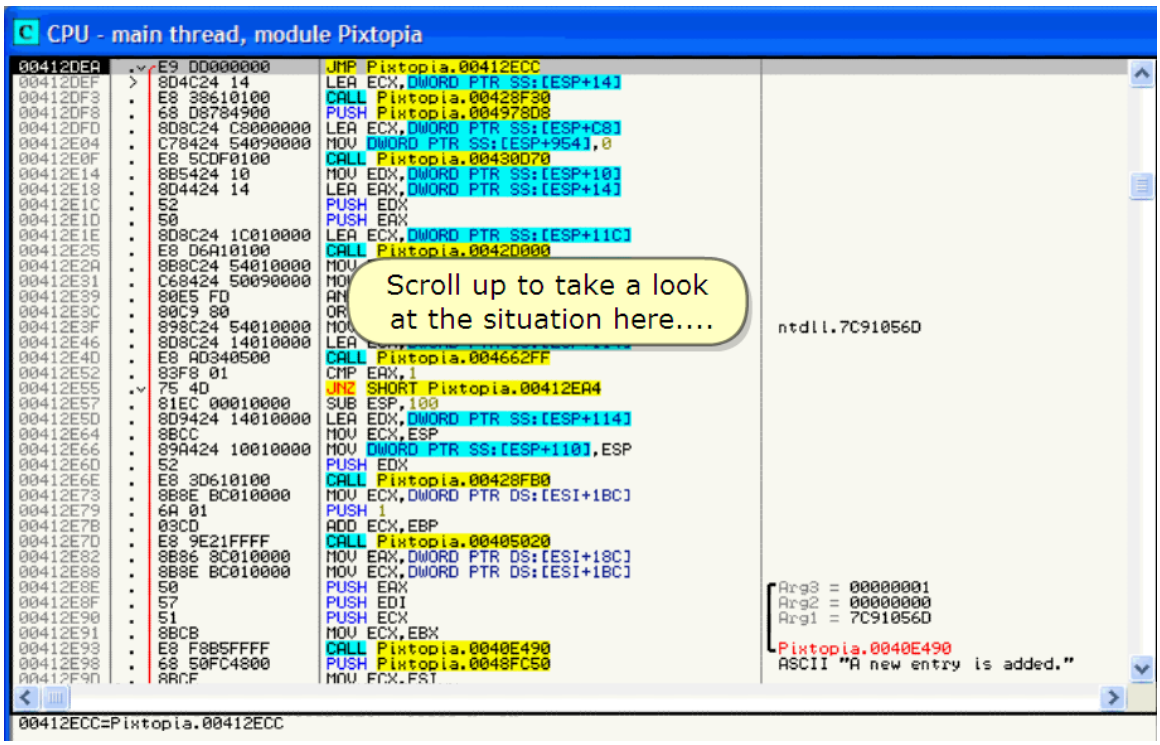
좋아. 이곳은 같은 구성이다.

Here is the creation of the messagebox.

이곳은 messagebox 가 만들어진 곳이다.

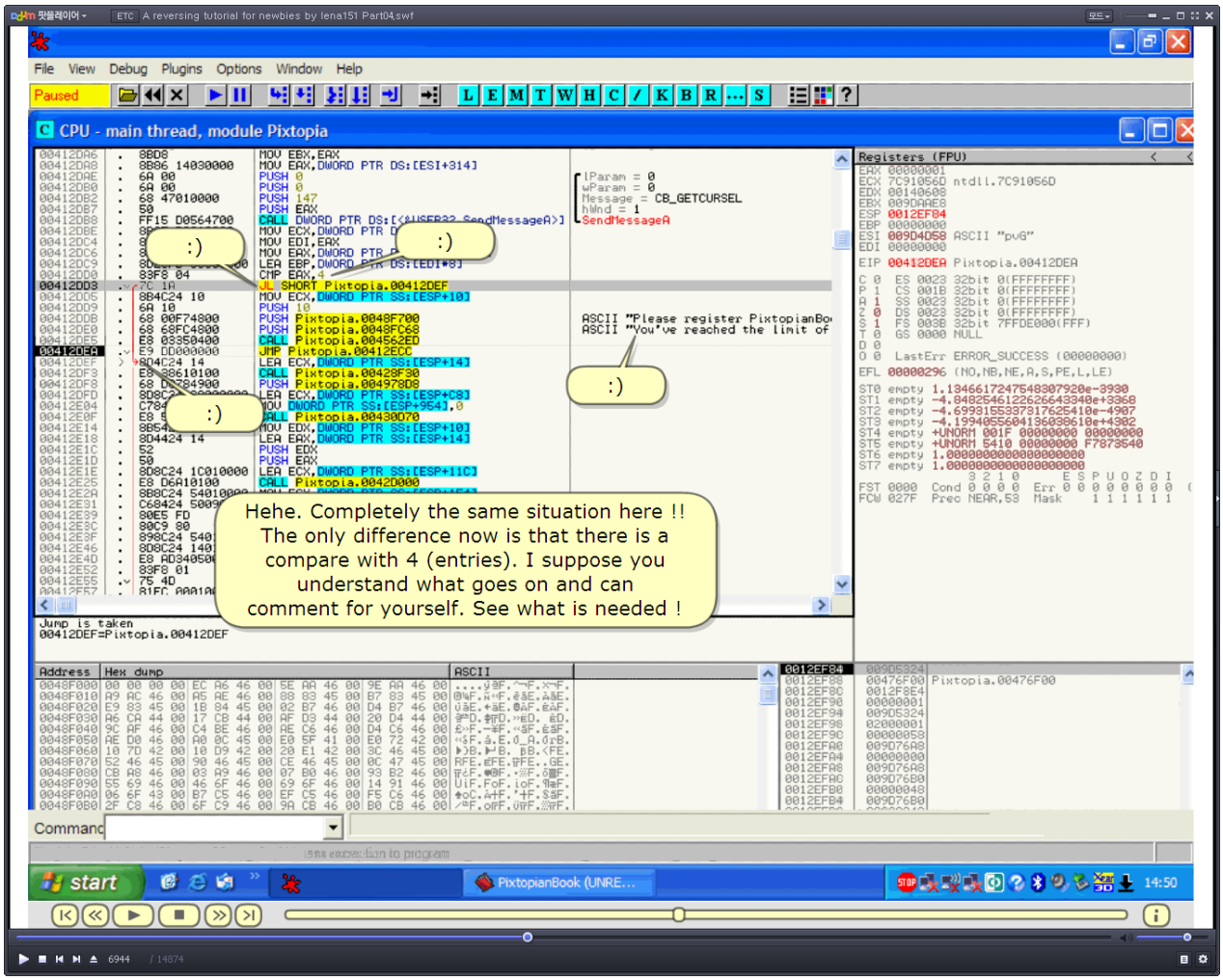
Let's see where it was called. Step F8

어디에서 불러졌는지 F8 눌러보자.



Scroll up to take a look at the situation here...

스크롤 올려서 이 상황을 가져봐.



:)

:)

:)

:)

Hehe. Completely the same situation here !!

헤헤. 완벽히 같은 상황이다. !!

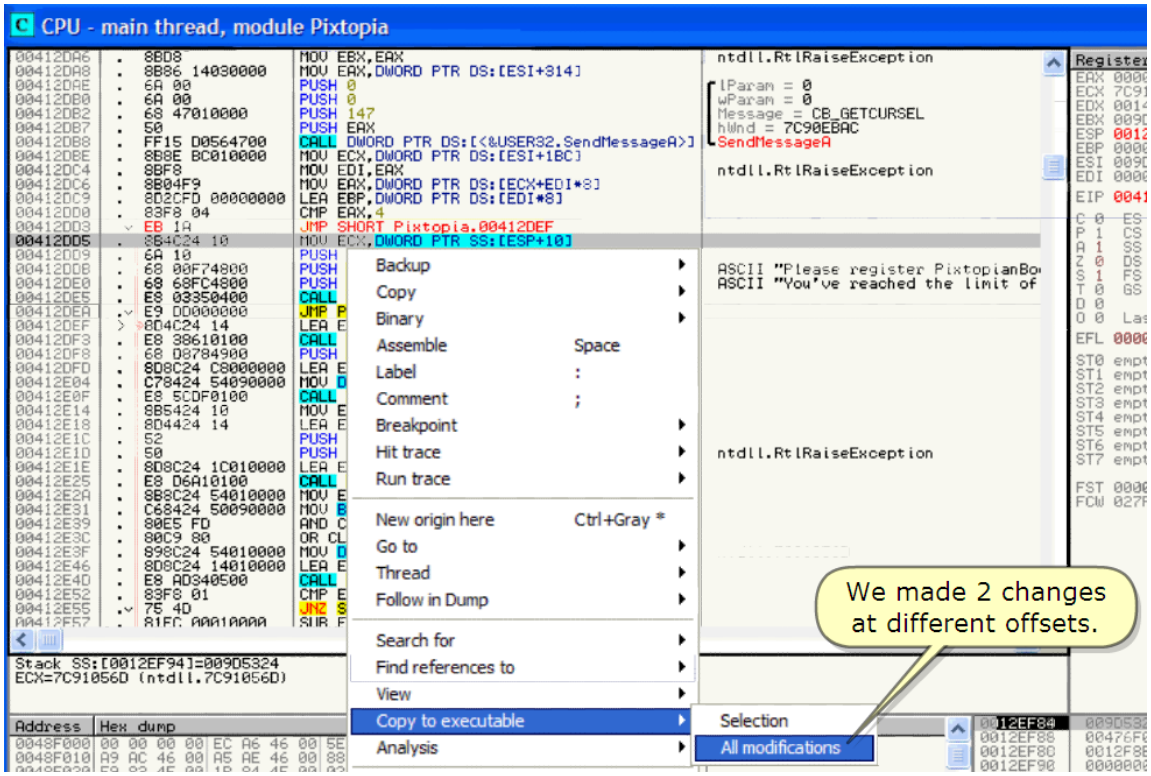
The only difference now is that there is a compare with 4 (entries). I suppose you understand what goes on and can comment for yourself. See what is needed!

오직 다른 것은 4 와 비교하는 것이다. 네가 무엇이 동작하고 너에게 comment 를 남기는지 이해했을 거라 생각한다. 무엇이 필요한지 봐라.

Right. Now, we will always jump past the messagebox to continue the job.

Save all the changes in the usual way.

맞아. 이제, 우리는 항상 과거의 meessagebox 를 jump 하면서 진행 한다.



We made 2 changes at different offsets.

우리는 2 가지 변화된 점을 다른 offsets 에서 만들었다.

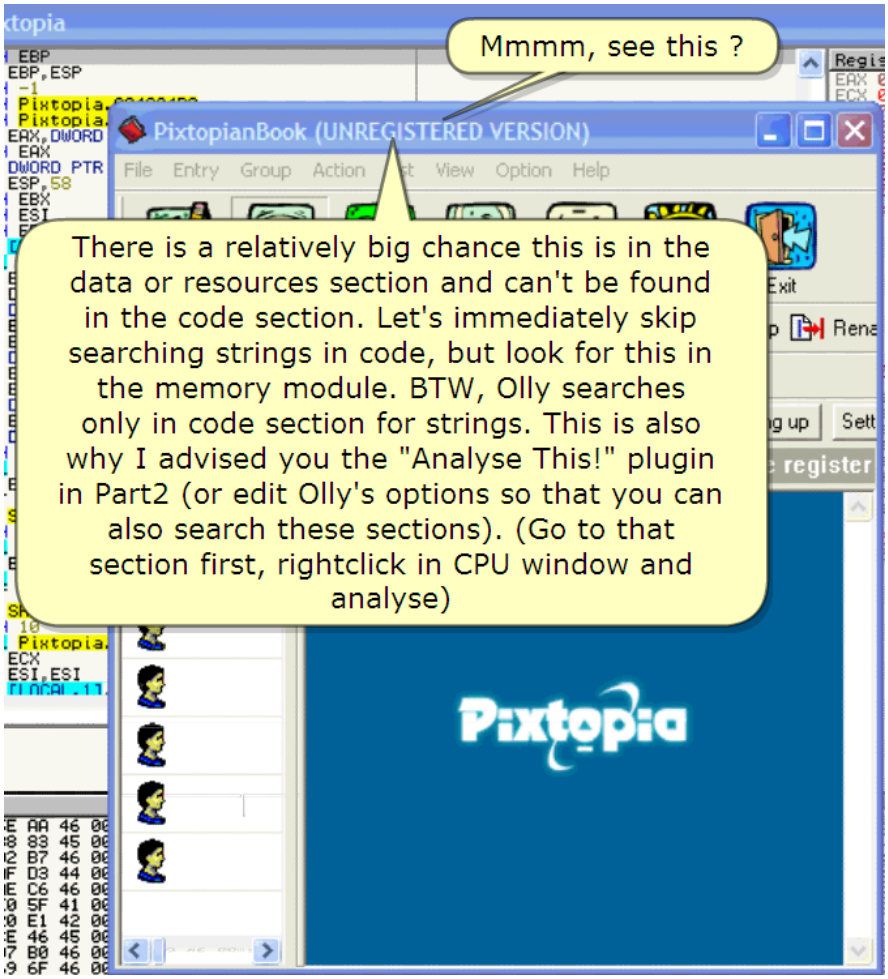
Saving under a different name

다른 이름으로 저장해라.

Load the newly saved application and run it to see what else need fixing :)

새롭게 저장된 application 을 load 하고 무엇을 바꿔야 하는지 보기 위해 실행하라.

7. Some aesthetical patches



Mmmm, see this?

이게 보여?

There is a relatively big chance this is in the data or resources section and can't be found in the code section.

그것은 data 와 resource section 에서 비교적 큰 change 다. 그리고 우리는 code section 에서 찾지 못했다.

Let's immediately skip searching strings in code, but look for this in the memory module.

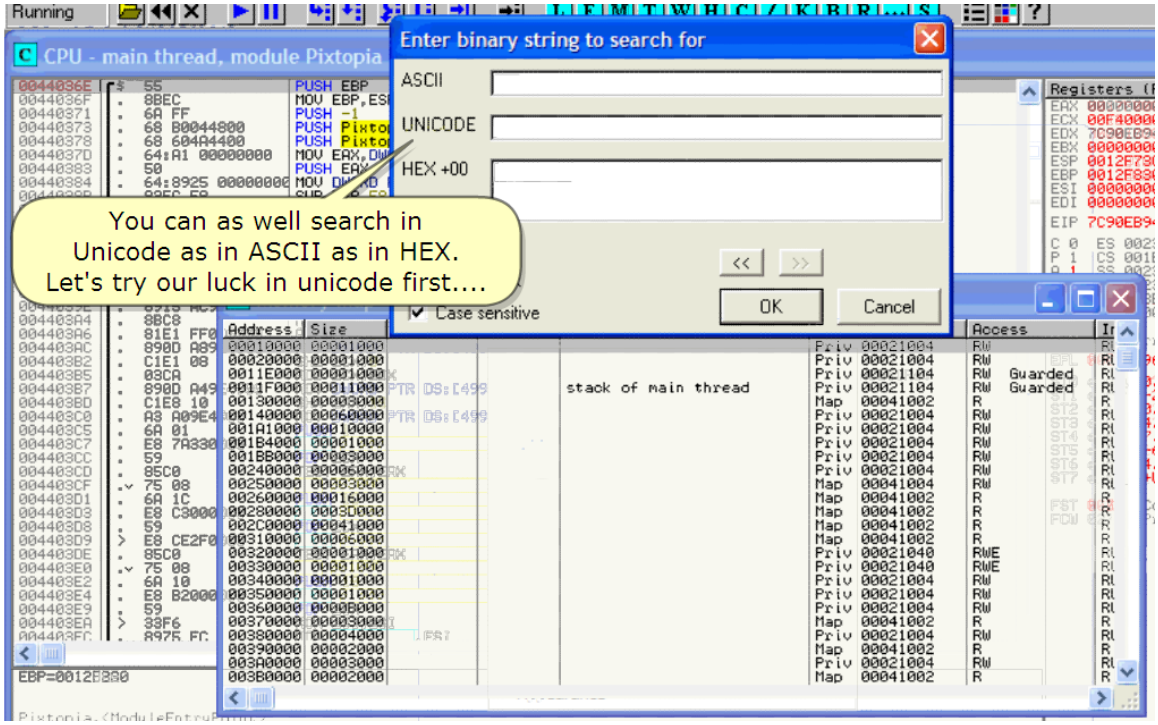
즉시 code 에서 searching strings 을 skip 한다. 그러나 이것을 위한 memory module 을 봐라.

BTW, Olly searches only in code section for strings. This is also why i advised you the "Analyze This" plugin in Part2 (or edit Olly's options so that you can also search these sections).

By the way, Olly 는 strings 을 위해 code section 에서 찾는다. 이것은 내가 Part 2 에서 너에게 충고했던 "Analyze This" plugin 이다.(또는 Olly's option 을 수정해. 그래서 너는 these section 을 찾을 수 있다.)

(Go to that section first, rightclick in CPU window and analyze)

(먼저 section 의 처음으로 가서, CPU window 에서 오른쪽 클릭하고 분석해라)



You can as well search in Unicode as in ASCII as in HEX.

Let's try our luck in unicode first...

너는 ASCII, HEX, Unicode 를 찾을 수 있다.

우리의 운을 믿고 unicode 먼저 찾자.

INFO : ASCII :

Stands for the American Standard Code for Information Interchange, pronounced as "ask-ee".

정보 교환에 사용되기 위해 미국에서 만든 표준 code 다. "아스키" 라고 불린다.

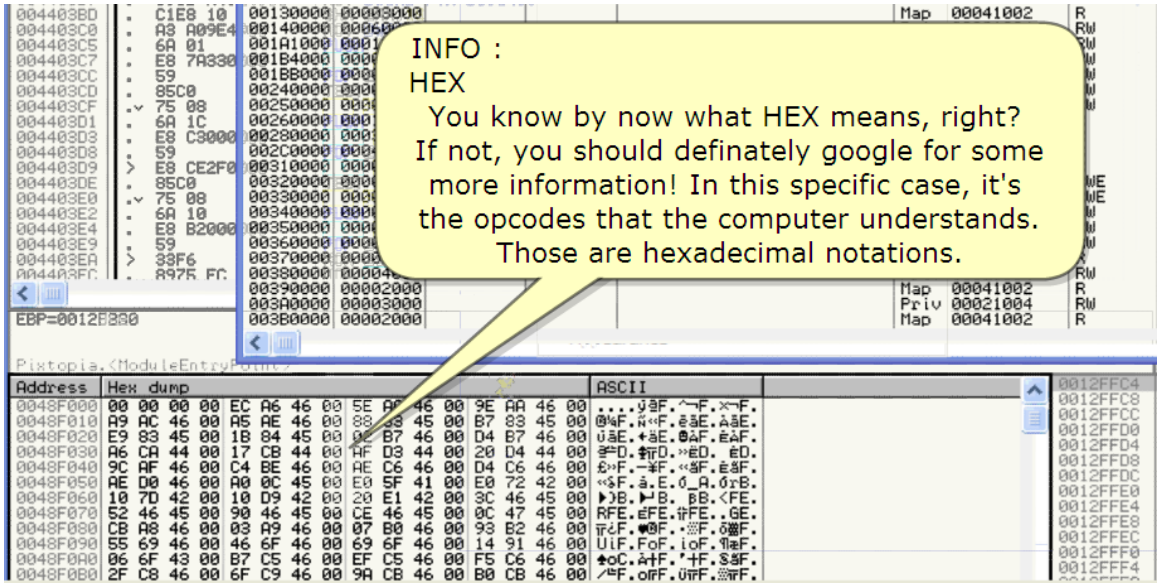
ASCII is a way of defining a set of characters which can be displayed by a computer on a screen, as well as some control characters which have special functions.

ASCII 는 character set 을 정의한 방법. 그것은 컴퓨터의 screen 이나, 제어 character 를 보여줄 때, 특별한 function 을 가지는 것을 보여준다.

INFO : Unicode :

Fundamentally, computers just deal with numbers. They store letters and other characters by assigning a number for each one. The Unicode Standard provides a unique number for every character, no matter what the platform, no matter what the program, no matter what the language.

기본적으로, computer 는 숫자를 처리한다. 그들은 할당된 숫자에 의해 각각 letters 와 다른 character 를 저장한다. Unicode Standard 는 모든 character 를 위해 유일한 number 를 제공한다. 어떤 platform 이든지 문제 없다. 어떤 program 이든지 상관없다, 어떤 language 든지 상관없다.



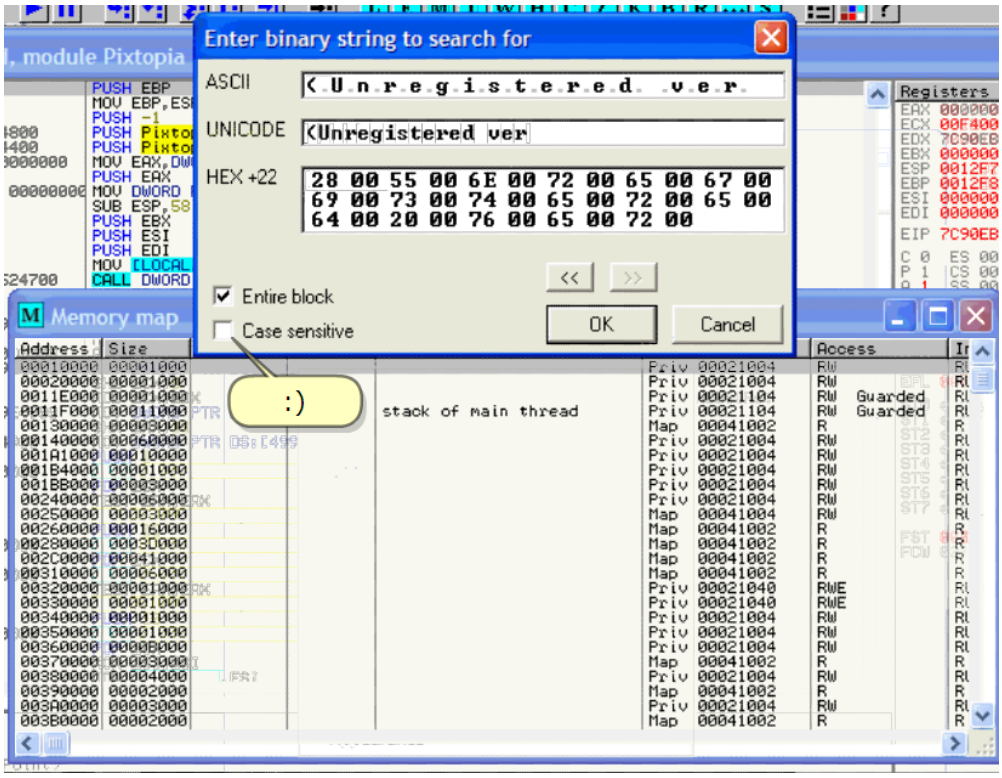
INFO : HEX

You know by now what HEX means, right?

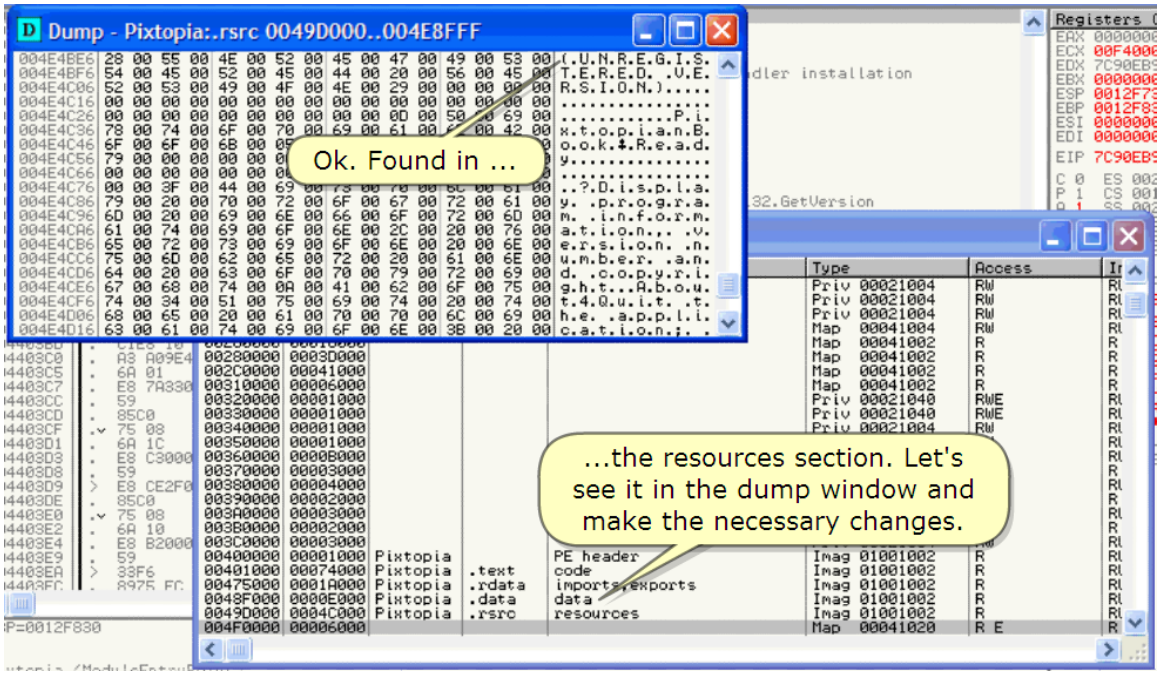
여기에서 말하는 HEX 의 뜻을 알지? 그렇지?

If not, you should definitely google for some more information! In this specific case, it's the opcodes that the computer understands. Those are hexadecimal notations.

만약에 모른다면 분명히 좀더 많은 정보를 google 해라. 이런 특정한 경우에, opcode 는 computer 가 이해한다. 그것들은 16 진수로 표기되었다.



:)



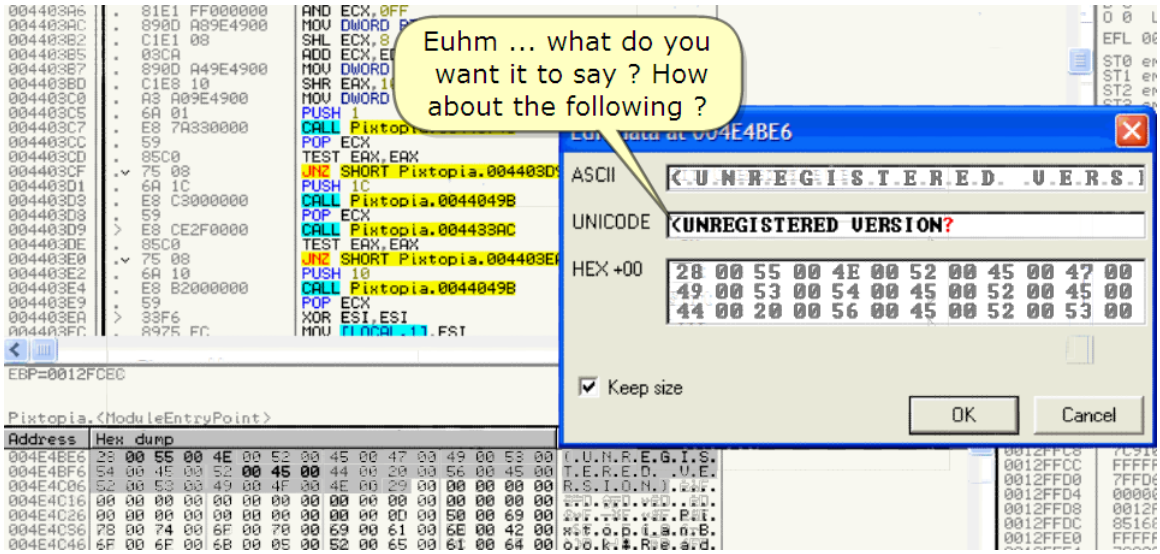
Ok. Found in ...

... the resources section. Let's see it in the dump window and make the necessary changes.

Ok. 찾았다.

Resource section. Dump window 와 변화가 필요한 것을 수정해.

역자 주) resource 범위가 49D000 부터 시작해서 size 가 4C000 다. 즉, 49D000 ~ 4E9000 까지다. 현재 unregistered version 을 찾은 주소가 4E4BE6 이므로 resource 범위에 포함된다.



Hum ... what do you want it to say? How about the following ?

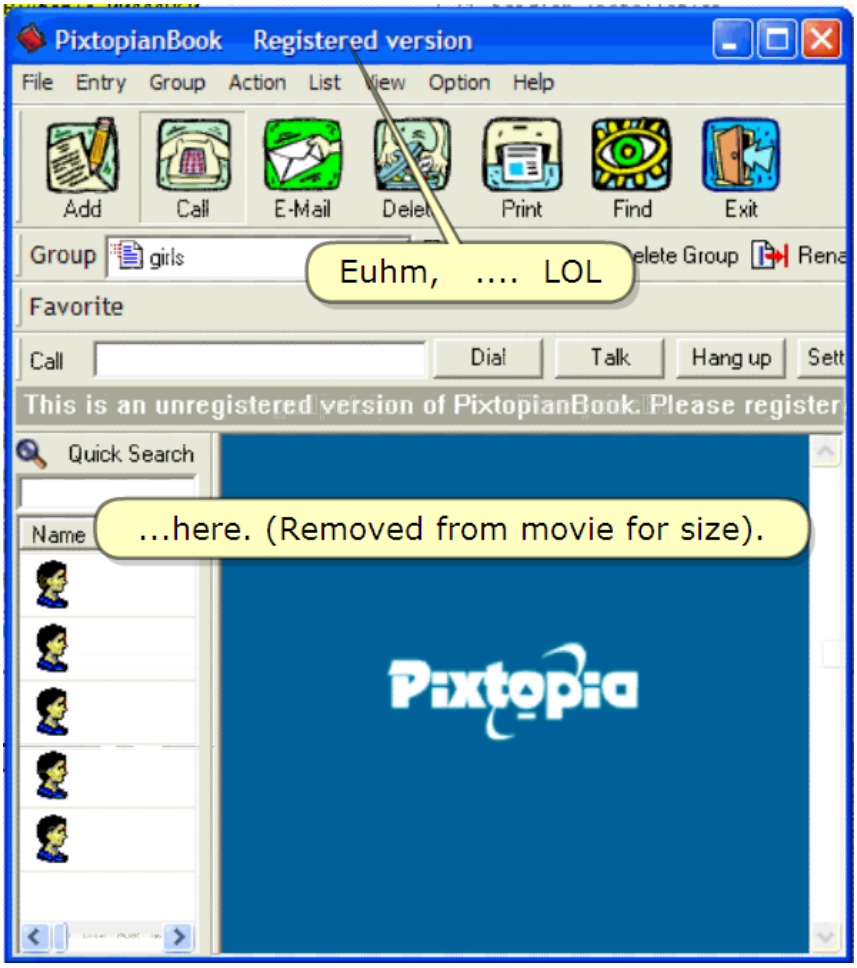
무엇을 원하는지 찾았어? 따라가니 어때?

Understand that these changes were only made in memory. Now, we still have to save these changes to file.

메모리에서 바뀐 점들을 이해하기 바란다. 이제, 우리는 바뀐 점을 file 로 저장해야 한다.

:)

So far, all good I hope :)
But there is more work to be done.
Load newly saved application and run till you drop ....
지금까지 좋았다고 생각한다.
그러나 많은 것을 더 해야 한다.
새로 저장된 application 을 저장하고 drop 할 때까지 실행하라.



...here. (Removed from movie for size).
Euhm, ... LOL
여기 (movie 를 위해 삭제됐다)

Remember ??
Right ! Let's go for this one too ....
기억해?

올바르게! 이것을 위해 하자.

:)
Same recipe !!!
같은 방법 !!!

And once more found in ...
:)
...the resources section.
한 번 더 resources section 에서 찾았다.

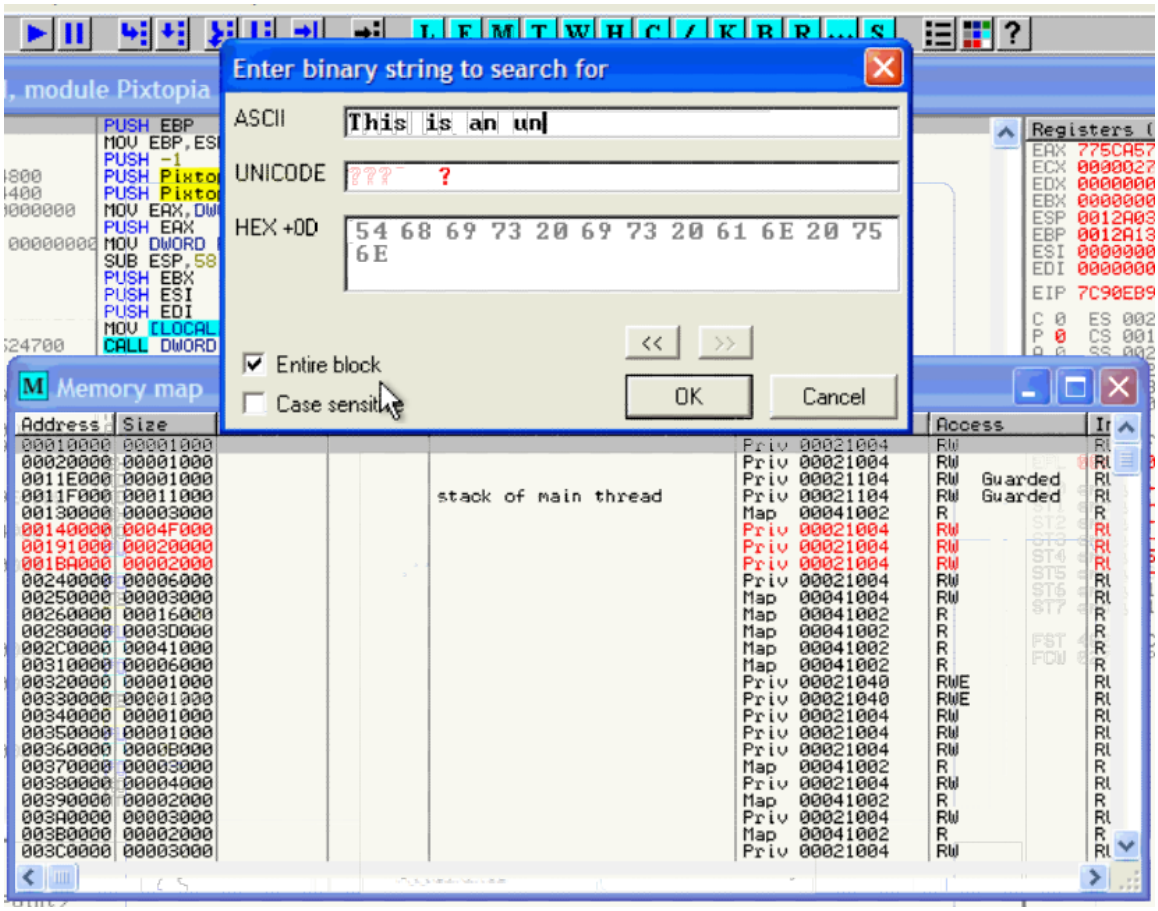
INFO:
There are more advanced ways to achieve the same but we will discuss these in later Parts in this series.
같은 걸 달성하기 위해 진보적인 방법들이 많다. 그러나 우리는 이 series 에서 나중에 의논할 것이다.
Common rule: use strings only as confirmation of the searched on your path, not as the main means of finding stuff!
공통 rule: 경로를 검색에서 확인하고, 찾는 주요 수단이 없을 때, strings 를 사용한다.

!
:)

So, this is what the application could look like when buying. :)
그래서, 이 application 은 우리가 구입했을 때와 같이 보여준다.

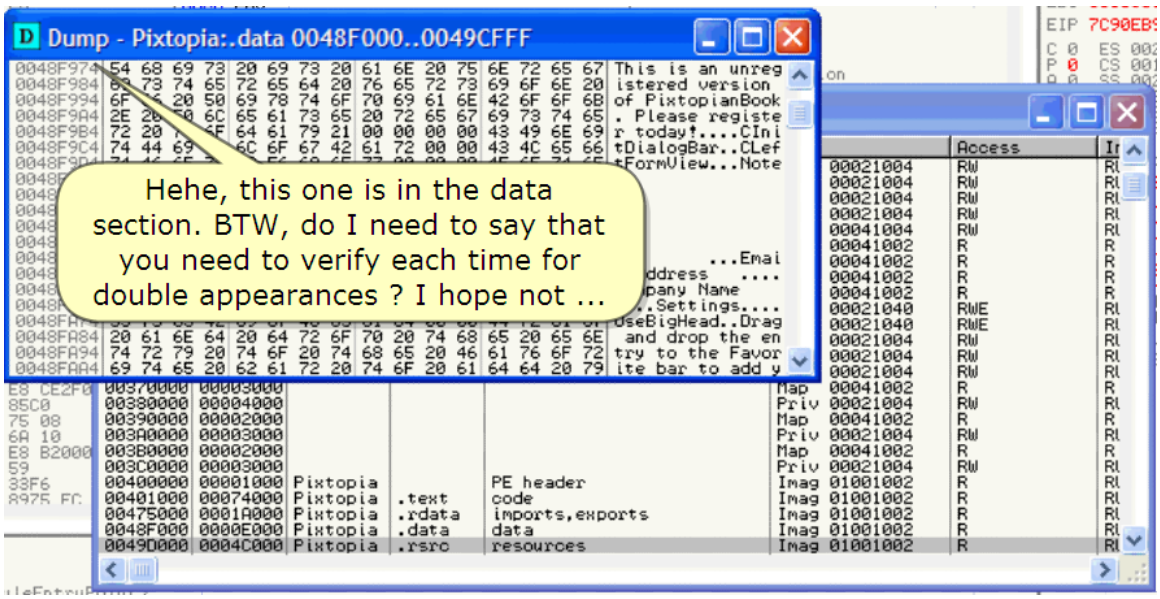
Well, it's again the same, right?
Save the changes, start the new saved application and run it till you land ...
... here. (Again removed from movie to reduce its size)
같은 방법을 다시 해봐, 그렇지?
바뀐 것을 저장하고, 새로 저장된 application 을 시작하고 여기를 도착할 때까지 실행하라.

Now, do you remember this and how it changes after a short while ? Look ...
잠시 동안 어떻게 바뀐 건지 기억해? 봐 ...



Let's go get it. Let's find this one in two ways, use whichever you prefer ...

얻으러 가자. 이것을 찾는 것에는 2 가지 방법이 있다. 먼저 사용했다.



Hehe, this one is in the data section. BTW, do I need to say that you need to verify each time for double appearances ? I hope not ...

헤헤, 이것은 data section 에 있다. By the way, 네가 2 번 appearances 를 위해 매번 검증하는 것이 필요한가? 나는 바라지 않는다 ...

I'm sure you know by now what needs to be done !!!
무론 네가 끝내기 위해 어떤 것이 필요한지 알고 있을 거라고 생각한다.

Saving also these changes to a new file
바뀐 것들을 new file 로 저장해라.

Let's see what this looks like !
어떻게 됐는지 봐라.

Still a nice welcome, and then ...
좋은 환영인사야.
Mmmmmm. I don't like it. Let's always keep the nice welcome and skip this from appearing ...
음. 나는 좋아하지 않는다. 항상 환영 인사를 유지하고 보여주는 것에서 이것을 skip 하자.

Where was that again ? Hehe, I forgot the exact address ...
어디였지? 헤헤, 정확한 주소를 잃어 버렸다.
Aha, right, there it is. Now, let's see if Olly knows where this address is pushed
아하, 맞다. 저기에 있다. 이제 봐라. 만약에 Olly 가 address 를 어디에 넣었는지 안다면.

Olly "knows" where this address is pushed. So, if you doubleclick this address, you will land in this string
Olly 는 어디에 주소를 넣었는지 안다. 그래서, 우리는 이 address 를 doubleclick 한다면 이 string 이 있는 곳에 도착한다.

Ok. Here, the string is pushed on the stack == preparing to write it on window
But also notice the compare just above == am I registered
Ok. 이곳이야. String 은 stack 에 넣어졌다. == 그것을 window 에 쓰기 위해 준비했다.
그러나 위에서 비교되는 것을 알린다 == 나는 등록됐다.

And if registered ... jump this routine == don't write it on the window !!!
그리고 만약에 등록됐다면...routine 을 jump 한다 == window 에 쓸 수 없다.

Mmmm, so that means in this application : unregistered
if EBP == 907
음, 그래서 등록되지 않았다는 뜻이야.
만약에 EBP == 907 이라면

Let's see when we are registered

우리가 등록됐을 때 봐라.

Scroll up to follow the jump leading here
스크롤 올려봐. Jump 가 이끄는 곳이 이곳이야.

Are we registered?
906 == registered
등록됐어?
906 == 등록

If not, jump to the place where we were before to put the string on the window
만약에 우리가 전에 왔던 곳으로 jump 를 하지 않는다면 그러고나서 string 을 window 에 넣었다.

Do understand that this is a splendid indication to verify where the registration is set.
Find the place where is decided about
이해했어? 이것은 훌륭한 말이야. registration 이 set 되는 곳을 검증하기 위한 곳.
결정되는 곳을 찾아라
EBP == 906 (registered)
Or
EBP == 907 (unregistered)
Changing this to always EBP == 906
---> always registered
EBP == 906 (등록)
Or
EBP == 907(미등록)
항상 EBP 를 바꿔라 == 906
---> 항상 등록

However, we know enough for this application.
Let's do this kind of stuff later in another example program.
Scroll down
그러나, 우리는 이 application 에 대해서 충분하다는 것을 안다.
여러종류를 다른 예제 프로그램에서 보자.
스크롤 내려봐.

Yep, assemble this JNZ to jump always.
It will keep the lovely welcome message on the window until we buy the application
예, assemble 해. JNZ 는 항상 jump 한다.
우리가 application 을 살 때까지 항상 window 에서 따뜻한 환영의 message 를 유지해.

Because you know how to do it, I have removed from this tutorial the saving of the changes to file, the restarting and running of the new saved application till we land ....
... here. The only thing that rests now is
그래서 우리는 어떻게 해야 되는지 알고 있다. 나는 바뀐점을 file 로 저장하는 movie 를 tutorial 에서 삭제했다. 재시작하고 새로 저장된 application 을 실행하면 우리는 여기에 도착한다.
여기. 오직 있다. 그것은 쉽다.

8. Testing the program

OK
OK
OK
Ok. It keeps the welcome message
Ok. 이것은 환영 메시지를 유지한다.

Ok. We can make more then 4 entries !
Let's enter an empty one
Ok. 우리는 4 entries 보다 더 많이 만들 수 있다.

Right !
All right, we can enter more then 3 groups.
Let's make one ...
맞아!
모두 맞아, 우리는 3groups 보다 더 많이 만들 수 있어.
하나 만들어 봐.
Ok.
Ok
And so we end with a full working program.
우리는 가득 찬 full program 을 끝냈다.

9. Conclusion

In this part 4, we learned to kill some restrictions in a "crippled" application. There are more possibilities and other solutions to do this of course.
이번 Part4 에서는, 우리는 "불구" program 에서 약간의 제한된 것을 kill 하기 위해 배웠다. 그것은 많은 가능성이 있고 다른 해결책들도 있다.
We also studied some other basic stuff. I hope you understood everything fine and I also hope someone somewhere learned something from this.

우리는 또한 다른 기본적인 자료들에 대해서 공부했다. 네가 모든 것을 이해했기를 바라고 누구나 누구든지 여기에서 무언가를 배웠으면 좋겠다.
See me back in part 05 ;)
Part 05 에서 돌아올께.

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.