13.The use of API's in software, avoiding doublechecking tricks

2012년 2월 4일 토요일

오전 9:30

Hello everybody.

모두들 안녕.

Welcome to this Part 13 in my series about reversing for newbies/beginners.

나의 초보자 reversing series Part 13에 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 skip할 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.

1. **Abstract**

In Part12, we found a case where simply patching the reg scheme sufficed to also effectively register the program.

이번 Part12에서, 우리는 program을 효과적으로 등록하기 위하여 간단히 reg scheme을 patching하는 사건을 찾았다.

However in this Part 13, we will reverse a "real" application to learn something about defenses to trick the reverser, in the case doublechecks.

이번 Part13에서, 우리는 doublecheck case에서 reverser를 속이는 것을 방어하고 배우기 위해 "real" application을 reverse 할 것이다.

For better comprehension and if you are a newbie, I advise you to first see the previous parts in s this series before seeing this movie.

좀 더 이해력을 높이고 네가 초보자라면, 나는 이 movie를 보기 전에 이 series의 이전 parts를 먼저 보라고 조언을 한다.

The goal of this tutorial is to teach you something about a program's behaviour.

이 tutorial의 목표는 너에게 program's behaviour에 대하여 가르치기 위한 것이다.

In my search not to harm somebody, I found an old version of Xoftspy(version 4.13.88) which is no longer available for download.

나의 연구는 누구에게도 해가 되지 않는다, 나는 Xoftspy(version 4.13.88)의 old version을 찾았다. 그것은 더 이상 download가 가능하지 않다.

Taking a look in the specialized media, I also found this application to be "cracked" already.

특화된 media를 봐, 나는 이미 "cracked"된 application을 찾았다.

Here, this applications is only chosen because it is ideal for this tutorial in reversing and it is targeted for educational purposes only.

여기, 이 applications은 오직 선택됐다. 왜냐하면 이것은 reversing tutorial에 이상적이다. 그리고 이 target된 것은 오직 교육적인 목적이다.

I hope you will exploit your newly acquired knowledge in a positive way.

네가 얻은 새로운 지식을 긍정적인 방향으로 이용하길 바란다.

In this matter, I also want to refer to Part 1.

문제가 있으면, Part 1를 참조하기 바란다.

1. **Tools and Target**

**이것도 똑같음**

The tools for today are : Ollydebug and… your brain.

오늘의 tools은 : Ollydebug와 너의 두뇌다.

The first can be obtained for free at

첫번째로 무료로 얻을 수 있다.

<http://www.ollydbg.de>

Todays target is Xoftspy 4.13.88 Because it can no longer be downloaded,

오늘 target은 Xoftspy 4.13.88이다. 왜냐하면 이것은 더 이상 download가 안된다.

I have included it in this package for research.

연구를 위해 이 package에 이것을 첨부했다.

1. **Behaviour of the program**

Because you know meanwhile about the importance of decent study of your target and because you've seen how to do it in the previous Parts in this series,

왜냐하면 너도 그 동안 너의 target 적절한 공부는 중요하다는 것을 안다. 왜냐하면 너는 이 series의 이전 part에서 어떻게 하는지 봤다.

I will only show you the things we will need to take care of whilst searching for the patches.

Patche를 찾는 동안 그것을 돌보는 게 필요할 때 보여줄 것이다.

Please do some preliminary study of the target on your own.

Target의 예비 공부를 하자.

INFO :

I already told you about packers/protectors, throwing exceptions at Olly though Olly isn't good with those.

이미 너에게 Olly에서 Packers/protectors에 대하여, 예외를 던지는 것을 말했다. Olly는 좋은 것과 함께 있지 않다.

Olly has a means of passing the exceptions to the program though which can deal with them using the Structured Exception Handler(SEH).

Not only packers/protectors can do this of course.

And here with Xoftspy, we have such a program. Remember that we usually add a range of exceptions 00000000-FFFFFFFF for Olly to pass to the program so that we have no problems with this. So, let me show you how to pass one distinct exception.

But first …

The program was opened in PEiD

And shows us this

And this

So, give it a go(or F9 of course)

Now, pay attention : the program runs and then …

Hop, there it happens already

See this exception here ???

Well, if you leave this too long unattended, Olly crashes each time

Set that Olly has paused the program but will crash anyway after short time

(Well at least, that's what happened on my system)

So, hurry and hit the Shift+F9 keys

… to pass the exception to the program

All right, the soft is running

Because I pressed the Shift-F9 keys meanwhile !

Now to avoid this for the future

Just add the exception to the list like this …

:)

:)

Olly automatically find the last occurred exception

:)

And adds it to the list of exceptions to ignore for the future (== exceptions that will be passed to the program)

INFO :

This is not necessary if you have added a range of exceptions that includes this one of course.

See also the previous Parts in this series about it. (Adding a range).

:)

If you want to return to the code of the program, you can do it like this

Doubleclick the main exe

We are not yet in the main exe, so … once more

INFO :

We land here in the code from the main program. We are not at the EP of course but at the first line of code from the main program.(First line after PE Header)

Now, let's study the program

;)

Mmmmmm

;)

Don't ok yet

All right. We know enough to find what we need to patch.

But return to Olly

1. **Finding the patches**

INFO :

Windows programs use API's to interact with the kernel. Hence, also finding input from users in a registration window for example is done with API's

A technique to find info on and to land in the registration scheme, is to use the API's to our advantage. The API's that are important for this are :

INFO :

DialogBoxes:

DialogBoxParamA

GetDlgItem

GetDlgItemInt

GetDlgItemTextA

GetWindowTextA

GetWindowWord

INFO :

MessageBoxes:

MessageBeep

MessageBoxA

MessageBoxEXA

SendMessageA

SendDlgItemMessageA

INFO :

Registry Access:

RegCreateKeyA

RegDeleteKeyA

RegQueryValueA

RegQueryValueExA

RegCloseKeyA

RegOpenKeyA

INFO :

Reading/Writing files:

ReadFile  
WriteFile

CreateFileA

INFO :

Reading data from INI file:

GetPrivateProfileStringA

GetPrivateProfileIntA

WritePrivateProfileStringA

INFO :

Reading data(other)

LoadStringA

lstcmpA

MultiByteToWideChar

WideCharToMultiByte

wsprintfA

INFO :

Time And Date:

GetFileTime

GetLocalTime

GetSystemTime

GetSystemTimeAsFileTime

SetTimer

SystemTimeToFileTime

INFO :

Createing a NAG-window:

CreateWindowExA

ShowWindow

UpdateWindow

INFO :

Find messageboxtext

SendDlgItemMessageA

SendMessageA

SetDlgItemTextA

SetWindowTextA

For a registration scheme, I usually place a breakpoint in one(or all) of the next API calls :

GetdlgItemTextA

GetWindowTextA

lstrcmpA

GetPrivateProfileStringA

GetPrivateProfileIntA

RegQueryValueExA

WritePrivateProfileStringA

WritePrivateProfileIntA

INFO :

This list is far from, complete!

There are a lot more, but I'm only giving you a guide here.

INFO :

Do you really have no idea what API to break on or have you tried "every" API but you don't succeed?

Solution: ---> rightclick ---> search for ---> all intermodular calls ---> rightclick ---> set breakpoint on every command ---> click "ok" button from registration or whatever you need to do ---> you will definitely break somewhere !!!

REMARK :

If you break too soon, even before you can click the "ok" button, then this(these) breakpoint(s) is(are) useless and may be removed till the program lets you click the required button ;)

So, for a registration scheme, I usually start breakpointing in one (or all) of these API calls

GetdlgItemTextA

GetWindowTextA

lstrcmpA

GetPrivateProfileStringA

GetPrivateProfileIntA

RegQueryValueExA

WritePrivateProfileStringA

WritePrivateProfileIntA

For your convenience, I included a file with all before noted down for future reference.

Question is how now how to find where and which of these API's is used. Well, it's often trial and error but decent study of your target can significantly improve your skills in this.

You can set BP's on API's in different ways.

One way is to use the command bar. Just press BP -space- followed by the API's name and press enter.

Example :

BP GetWindowTextA

BTW, Olly comes with the commandline plugin which is basically the same. Or find the commandbar plugin at <http://www.tuts4you.com>

Or you can also set BP's on API's like this ….

Just follow along.

Push Ctrl+N to bring up the Names

And find GetWindowTextA

INFO :

One of the other ways is to type the API's name in the Goto box and breakpoint where you land in the dll

See previous tuts

:)

Let's start with this one.

:)

3 BPs set

And let's continue registration

Mmmm, we break here

… already before displaying the window

So this is probably a useless API

Or better said, a useless BP :)

Because we didn't click the OK button yet

So let's remove it

Doubleclick opcode or F2

And continue

And register

Right

Indeed, let's take a look in Win32.hlp what exactly this API does and why I suspected this API to participate in the registration process

The API name explains a lot already, and I think the rest is clear by now too. So, let's see what the API exactly does in our case …

GetWindowTextA reads 13h(19d) bytes in the text box

And puts them in a buffer at 00EE4C30

INFO :

Reading the code like this reveals that the name is only read for 19d chars, more input is useless !

INFO :

There are three textboxes for our data on the registration window. Hence, we need to click another 2 times to retrieve all the data. I did that but removed it from this movie to reduce its size till we land ….

… here

Now, I always take a fast overview

By stepping over the code(F8) till I see something interesting

Now I skipped some of the film to reduce its size(not much though)

Just by stepping F8(some returns too !)

Until we land here

So step F8 till here

Mmmm

Length of the name

Aha

:)

Mmmmm,

"Is something entered" in both boxes

First the length of our name

And continue stepping F8

Then the length of the key

INFO :

All this is not so important of course. But it really helps to be able to read the code this way. It also helps in "When do I need to expect the good things to arrive ? :)

Now, pay attention to conditional jumps

We need to find where we jump to the BadBoy

RESUME :

But let me resume so far first. Using the API GetWindowTextA, I managed to land in the registration scheme. First, the program verifies if we effectively made all necessary data input.

After this, we can expect the verification of the input(serial).

So, continue stepping F8 …

Scroll down to see what this is

Huh???

BadBoy …

And GoodBoy

Scroll a little up again to see all the conditional jumps

And study the code better here.

See this, twice the same routine with the same calls and jumps!!!

So this is definitely doublechecking registering

Both jumps go past the BadBoy

But eventually, let's make the first JNZ jump always to avoid doublechecking

So it's clear that the important stuff

Is in this call !!!

Or perhaps here ???

Let's first try the last call(most probable the good one)

That means that we can skip all the …

Previous lines by placing a BP here so we can run till here next time

Important : note that we DON't want AL to be zero when retuning from this call Remember that.

Because, if AL == 0 --> Badboy !

And here too, just in case ….

Now step over till call (F8)

And step in call(F7) now

Once more : when returning here, AL must be different from zero!

And so we land here

Now always take an overview first and scroll down

Mmmmm, quite a big routine ….

Indeed, here is a return

Aha, and here too

Yep, I hope you understand this leads to the BadBoy

Scroll up again

Aha, and this leads to the GoodBoy

INFO :

In a clear scheme like this one, we can immediately choose the right path to follow and choose to end (with the right cond jumps) up in the Goodboy scheme, already from the start.

So, this means not to jump this place ;)

Scroll up again

Mmmm, the jumps that will decide it all

And all to the same place :)

… the place we don't want to jump too

Let's step over and see what goes on

Look at jump taken or not each time you land on a jump

As we do not want to jump past the GoodBoy

Ok

Ok

Loop, no problem

INFO :

It is in loops like this that a program reads the buffer with data and copies it in a register(here EDX and ESI respectively) so it can "work" with the data.

If you look closer, you will notice that the program reads the double serial.

:)

:)

Not far, so no problem

Place a BP after the loops and press F9 to avoid turning in circles like a squirrel LOL

Run till BP

Remove BP(F2)

And step over(F8)

Mmmm, here are the jumps that need our attention

Scroll down

BadBoy

Let's make sure we go to the GoodBoy to see what goodies that brings us

Look here

And see where the jump would take us by scrolling down

Mmmm, past the good return …..

Remember this is not where we want to arrive

Remember that we want to return here

After AL has been set to 1 !

All right.

Scroll back up

Yep, change the flag so we do NOT jump

I trust you know this well by now?

Doubleclick to change the flag and

See change

Ok. Step over jump

See here

And again

Step over jump

:(

:)

Step over jump

Remember though that we will have to give these conditional jumps a treatment afterwards

INFO :

I will already tell you we won't need to change them after all ;)

Just continue to see what surprises the program has for us

:(

Jumps also to the badboy,

Hence ….

But scroll down first

:)

:)

Scroll down and continue stepping

So, this way, I forced the program to think I entered the right serial

:)

We will return AL == 1

Scroll up to remember where we are

AL = 1

And that's what we want !!!

Now jump (F8)

This is where we wanted to land, past the Badboy

And execute the GoodBoy F8

And BAM !

The program pops up, telling us that we successfully registered!

So, did you understands?

We have to NOP all the JNZ

We changed the flag for

However, the program ahs still a surprise for us …

;)

Olly pauses the soft when coming back

So, press F9

Mmmm, I forgot to remove the BPs

And so, Olly breaks in them when pressing F9

Remove the BP now F2

And run

INFO :

To avoid all confusion : I have not restarted the program, only pressed F9 after running the registration scheme successfully !

Huh ?????

Let's think ….

So the program either doublechecks or runs another routine to verify

If it was effectively the right serial that was entered

Let's see this in the code and …

Go back to Olly

:)

I planned to look for some strings but … look what is here !!!!

This is exactly what I was going to search for !!!

That is the string we need !!!

Let's see, doubleclick to go there

And we land here

RESUME :

I successfully registered and then ….

The program says it is not registered !!!

Searching for these last strings however, we land here in another routine that also seems to deal with registering. Will this be another doublechecking ???

Let's see better here and scroll up

And study the code here

Mmmmm

I suppose you begin to get the hang of it ???

Meaning that AL will decide about goodboy or BadBoy here

And it's all decided in (one of) this (these) last call(s)

Doubleclick opcode

To set a breakpoint

Scroll up to also set a BP in the beginning of this routine.

And let's restart to see what happens …

And run

;)

BAM !

We break in the breakpoint we just set !!

Now, think with me. When clicking the About box button … we land here in the verification of the registration.

This is not abnormal of course : in the about box is displayed if we are registered or not.

However, when registering by the registration routine, then here is a doublechecking done.

That also means that this routine is the main "registered or not".

Hence, if we patch here, then there is a relatively big chance registration will be permanent.

Let's try it…

First, let's study the code here.

Scroll down.

Mmmm, ok. There are no jumps till here, so press run (F9) to land on the BP

And we land here on the call

I suspect it's all happening in

Step in the call (F7)

But first remark that we need to return AL >< 0 to be registered !!!!

We land in the call

And let's see what's all happening here

INFO :

It's never a bad idea when going through registration schemes to set a BP and restart and run.

If you do that here, you will see that this routine is ran at startup(breaking in BP here), meaning that indeed here is decided about registered or not at startup.

Scroll down

Run till here

Remove BP

And step F8 from here

Ok

Mmmm, ok, no harm done

BTW, note the data we want the program to register us with

No harm done

INFO :

I have deliberately removed some of the last steps to eb able to reconstruct with you what has happened.

But it is rather simple in fact : we must return

AL > 0

And we see that this depends on BL

We are not registered at this time, so, we will jump here passed the good news :

MOV BL, 1

And because AL is zero

… we have not jumped here

Thus setting BL to zero ---> unregistered

And we will jump the setting of BL to 1 ( ---> registered)

And so, we jump the good news landing here now

Do you understand that this is the place to patch ????

Now, do you understand that IN ADDITION and for extra surety, I could also have patched …..

This JNZ to JMP ?????

But of course, this needs always to be patched !!!

For complete certainty

Now, let's make changes permanent

1. **Patching, Saving and testing**

;)

Save under a different name

;)

:)

All fine !

1. **Conclusion**

In this part 13, the primary goal was to make you learn to "think reverser" and not give right up if a program acts otherwise then expected. Give it another try, and finally, you will succeed.

So, in this Part13, we studied what to do if not simply patching the reg scheme effectively registers the program.

I hope you understood everything fine and I also hope someone somewhere learned something from this.

See me back in part 14 ;)

The other parts in this series are available at

<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)