Lena 16강

2012년 2월 4일 토요일

오후 10:01

* 1. **Abstract**

In this Part 16, we will reverse two "real" applications to study using a program's resources for reversing.

This is because indeed, the best practice is found in real applications. I will also work in a reversed Olly, exactly intending to show you what RE(Reverse Engineering) is capable of.

For better comprehension and if you are a newbie, I advise you to first see the previous parts in this series before watching this movie. The goal of this tutorial is to teach you something about a program's behaviour.

In my search not to harm authors, I found an old version of URLegal (not updated since 2004) and an old Gif Movie Gear(version 3.02).

To oppose possible misuse, I only included both main executable (not the install exes!) for your research and to study the shown techniques.

This makes the programs useless for other purposes then studying material. Taking a look in the specialized media, I also found these software's to be "crack already. Here, both programs are only chosen because they are ideal for this tutorial in reversing and they are 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

* 1. **Foreword**

Taking a look around, you'll notice that Olly looks quite different then usual. Well, there is nothing special about that: just right-click in any window and you can change its looks by choosing "appearance".

And also by changing the looks for code, jumps, etc etc

But you can also change "anything" you want!!! I'm sure there is some freeware on your HD that you'ld like to give another touch? Or some shareware that you purchased that you'ld like to personalize somewhat?

See an example of what one can do with a debugger too! And think of all the other possibilities like protecting against debugger detection, adding functions, etc ;)))

Another example : see all these buttons that are different from the usual ones from Olly !!!

INFO : The buttons were not changed by me, I departed from an edited version, although it's very easy to do it in a resource editor !!!

And see the extra buttons giving immediate access to our favorite or most used tools!

INFO :

Coding is required for the enhanced menu and the extra buttons. However, my friend arjuns' plugin for Olly (see the included file Tbar.manager 0.3 Gold) also adds all this functionality to Olly.

As usual for plugins : just unrar in Olly's plugin directory and access it through the plugin menu in Olly.

In part 10 in this series, I included a "translated" VeoVeo. Let's see how easy it is to achieve its translation.

(with a good translator software, I don't understand a word of Spanish, sorry …)

:)

;)

:)

Just see all the translated menu items here

And see them here again in the data section, just plain "simply" translated!

INFO :

Keep each begin byte the same and also in the same size if you are inexperienced.

Notice that the empty spaces were occupied before …. Less can, but more can't (if you don't know how)

So, go on! Explore! Try out and learn !

This is what reversing is about !

But first, let's take a look in some programs to see how resources can be even more helpful

* 1. **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>

eXeScope is a shareware resource editor Google for and download the latest version 6.41 (at the time of writing this tutorial)

Reshacker (Resource Hacker) is a freeware resources utility and downloadable at

<http://delphi.icm.edu.pl/ftp/tools/ResHack.zip>

I also included the before mentioned plugin Tbar.manager 0.3 Gold.Final

Again, I can't help you with the brain ;)

Todays targets are URLegal v.3.3 and Gif Movie Gear v.3.02 of which I have only included both main executable in this package for your research.

* 1. **Behaviour of the program**

Because meanwhile, you understand the importance of decent study of your target and because you've seen how to do it in my previous Parts in this series, I will only show you the things we will need to take care of while searching for the patches.

Please do some preliminary study of the target yourself.

;)

Well, we could use this feature, but I've shown how to deal with this kind before. But I remarked something quite different from usual in this application: a closing nag. Just see!

The only way to leave the application, is to promise the author that you will soon register.

Clicking all the rest results in something you don't want at first …

Let's reverse that.

First : take a good look at this nag and study what you see on it, it will be helpful

* 1. **Finding the patches**

INFO :

Another proof of how practical reversing can be. The menu looks quite different from yours I suppose ?

BTW, there's no need to ask me for this Olly, after this Part 16 …. You will be able to make your own personalized Olly, although this one has some extra features :=)

;)

INFO :

URLegal is opened "read-only" because I opened the program in eXeScope while it is loaded in Olly.

That's no problem however because I don't plan to edit the resources in eXeScope : I only want to study resources.  
Should you want to make changes to URLegal in eXeScope : however, then you must first close URLegal in Olly.

There is a lot to see in this resource utility but there is no need to explain eXeScope : it is self explaining. Just play around a little in the program.

We are interested in the resources!

… and especially in the Dialogs :)

Let's study these and see if we can find anything useful !

Mmmm, no, this is the About Box.

No need to dig deeper.

;)

Ok. Let's see Dialog 103 then

Aha!!! This is the one I'm looking after already !!! If you studied the looks of the nag well, then you will immediately recognize the title !!!

Let's see what it should look like

INFO :

Because URLegal is opened 'Read-only" (see before), this button is greyed out.

If not, you can edit almost anything you want and really change behaviour and looks.

If this is all new for you, try it out and learn !!!

BTW, don't forget to save your changes, but make a backup first!

Well, I suppose it's clear what this is.

No need to verify it in the application.

REMARK :

Notice that this is in Dialog 103

eXeScope helped us fine. We know enough by now. We can close eXeScope.

Ok. We are back in Olly.

Next, I need to give you some ….

INFO :

The DialogBoxParam function creates a modal dialog box from a dialog box template resource. I already told you in previous Parts in this series that the arguments to create a Dialogbox are first pushed on the stack. One of these arguments for the Dialogbox is the TemplateName.

lpTemplateName :

Identifies the dialog box template. This parameter is either the address of a null-terminated character string that specifies the name of the dialog box template or an integer value that specifies the resource identifier of the dialog box template. If the parameter specifies a resource identifier, its high-order word must be zero and its low-order word must contain the identifier.

INFO :

And it is exactly this TemplateName that we have found in eXeScope !!! Now, because we know that this TemplateName must first be pushed on the stack …. It is very easy to locate where the Dialogbox is created.

INFO :

If all this is rather difficult to understand, never mind. That's why I have chosen two programs and I will try to explain in the second part of this tutorial showing you all this with explanation from Win32.hlp :)

I am sure it will become clearer in a while.

:)

INFO :

eXeScope displays the resource identifier values in DECIMAL !!!

Hence, we need to convert its value first back into hex.

:)

103d == 67h

:)

Press <enter>

;)

Only one occurrence :)

Let's go there and see it in the code by doubleclicking this line

;)

Scroll up see all better

Let me resume. So far, I found a closing nag in the application. Searching the resources in a resources editing utility, I found the "identifier" (=name) for the nag. Knowing that the identifier (here == 67h) is pushed on the stack right before the creation of the dialogbox, it is really easy to find the creation of the dialogbox. And that's exactly where we have landed at this moment.

See how the resource identifier (and the other arguments) are first pushed on the stack before

The Dialogbox is created in the API

So, this small routine is the creation of the nag. Let's now find where this nag is called.

There are different ways to find that call. I have already explained the stack method before. I'll explain shortly once more.

INFO :

A call needs to remember where to return to when the call's code is executed. That return value is pushed on the stack.

So, by simple breakpointing in the beginning of the routine, we can see on the stack where the code will return to after the call is finished. Just follow.

Ok. Because this code is executed at this moment (the nag is displayed), we need to restart the application to be able to break in the BP. Just follow …

:)

:)

We break in the beginning of the nag routine, hence, the first value on the stack is the return address from the call.

;)

Scroll up to see more code

This is the return address from the call

Hence, this is the call that calls the dialogbox.

Are we registered?

Jump past the nag screen if we are

Scroll up

… and it is always the same …

In the proceeding call is decided on jumping or not == registered or not

Indeed : we need to investigate in the call.

:)

We land in the BP.

Study the code here and notice that EAX may not be zero to jump the dialogbox.

Let's step in the call

Notice that we start with EAX == 1

Scroll down to see better

EAX may not be zero when returning…

That's easy here, with a lot of possibilities

Assemble MOV EAX, 1 here (5 bytes)

But first, always set a BP in the beginning of such a routine to verify if the program uses the same routine at startup … in that case, the program will be immediately registered too …

Or like this

Will the program be registered with these changes ?

Let's find out …

Right. The nag has gone …

Now, let's restart to see if we break in the same BP …

:)

Bam! We land in the same BP indeed.

So, this means that the program uses the same verification scheme "Am I registered ?" for the nag as when starting up.

Hence, patching this scheme will immediately register the program too !!! Let's try it.

Save these change to file as URLegal1.exe (You know how to do it, I've done it too but removed it from this movie to reduce its size) until …

...we drop here at EP again.

I loaded the saved file already.

* 1. **Testing the patches**

Greyed out!

The nag's gone too :)

All seems to work fine.

Ok. Let's look at this in another program.

And so we land in the new opened soft in Olly : Gif Movie Gear v3.02

* 1. **Behaviour of the program**

Again, this is not the goal but …

...this is the goal !

… also a closing nag… but the approach is the same for all nags if dependent from registration

See that we have 30 trial days

And the only thing we can do is press the order button until

A timer triggers the OK button

Ok. We know enough.

* 1. **Finding the patches**

INFO :

I will use Reshacker for this target.

It is clear - I hope - that you may use the resource editing utility of your choice.

:)

Oops! Quite a lot of dialogs !

Ok. Start with the first :)

???

This is immediately what we are looking for !!!

Let's verify

Sure, that's it!

Remember! The TemplateName is 100

We know enough !

;)

;)

:)

Quite a lot!

Let's find the right one.

:)

:)

:)

Mmm, not the right one!

We need to break after closing the nag!

This is not what we're looking for

Bam! This must be the right one

:)

Scroll up

Remember, you can find info on APIs in Win32.hlp.

BTW, see the TemplateName I told you about that is pushed on the stack (with the other arguments of course)

INFO :

It is advisable to look up every API you don't know yet until you understand what they mean. Especially for non-programmers, it is a helpful guide to reversing success

Study the code here.

Notice that EAX needs to be 1 to jump past the dialogbox when returning here from this call

Hence, we need to take a look in this call

;)

Fine. We break in the call. Let's remove the unnecessary BPs first.

And press <Del> on your keyboard

Right. We stepped in the call that decides on jumping or not and … it seems we land in the verification of the registration!

That's why I know that

That's why I know that

Scroll up a little. It's a habit of mine to look what code is just above …

Nothing interesting :)

Another habit of mine is to first take an overview, look at the code here, then scroll down till the end of the routine to see what we can expect

Does this ring a bell ?

Remember, we need to return with EAX > 0

Scroll up again.

Ok. Start stepping F8 to see what comes up

Win32.hlp !!

It's not difficult to understand. Just look ….

Now, scroll down for the results

;)

Or here of course

I don't think this needs more explaining.

Continue stepping.

Success !

:)

I don't think this needs further explaining.

But in short:

The API RegQueryValueExA verifies for a value of the key ….

Opened by the API RegOpenKeyA.

That value is the registration name.

In this case : failure garantied :(

번역 주)guarantee가 아닐런지?

And in case of success, next is the registration key

INFO :

In case this confuses you : this is not a verification for the right name and key !!! It is only a verification "Is there a registration name (and next is registration key) in the registry?"

Scroll down and step to see it happen!

Failure!

Hence, we jump.

To land here at the end of the routine already

I am sure this doesn't need explaining :)

And then here is decided about EAX.

You probably have a solution in mind for this.

See the value for EAX first.

EAX is zero.

But remember that we must return EAX == 1

Well, there is no big problem in that because we have two bytes to play with.

Different possible solutions but see this one

In this case, I'm relatively sure that this scheme is also the verification scheme at startup.

Hence, I will immediately save the patches to the file, but still continue verification after that.

:)

You know how to do it, so I removed it from this movie to land here again after saving.

… and so let's continue stepping. I have saved the patched file but the nag should not appear any longer !!!

:)

:)

We jump the nag !!!

Let's run !

Right.

* 1. **Testing the patches**

:)

Registering is not longer possible …

… because we are already !

And the nag is gone too!

* 1. **Closing remarks**

Of course it isn't always possible to use the resources for reversing an application. For example in VB and Delphi, it can give quite a lot of problems for the resource editor to find anything useful (see VB concept)….

But in general, it's often worth trying ;)

INFO :

After unpacking (especially with protectors), an application's resources have often been divided over different sections. In that case, no resource editor is capable of resolving that.

However, there exist resource rebuilding utilities that can handle this perfectly after which the program is accessible for resource editing app again.

IMHO, the best free utility for that is FixRes from dReAmThEaTRe. But I'll see into this when dealing with unpacking/unprotecting

* 1. **Conclusion**

In this part 16, the primary goal was to study how to use the resources to reverse a program. Meanwhile, we also learned something about the power and the positive possibilities in reversing.

I hope you understood everything fine and I also hope someone somewhere learned something from this. See me back in part 17 ;)

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)