Lena 20강

2012년 2월 7일 화요일

오전 12:01

* 1. **Abstract**

In this Part 20, we will reverse (unpack) some unpackme's to learn something about packers/protectors/compressing/unpacking.

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

In my search not to harm authors, I chose these unpackme's because all are freeware packed, open source or deliberately chosen older versions of the packer/protector.

The goal of this tutorial is to teach you something about a program's behaviour. Here, these unpackme's 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 Part01.

**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. **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>

PE Tools Xmas ed can be downloaded from

<http://www.uinc.ru/scripts/load.cgi?files/neox/PE_Tools.zip>

LordPE can be downloaded here :

<Http://www.sistemo.com/LordPE/LPE-RTS.zip>

Unfortunately, no download for the brain ;)

All tools are freeware.

Todays targets are 8 unpackme's

I included them in this package for research.

I included the Ollydump plugin too. Also included are some plugins to hide Olly from detection. Only use those for the 2 last unpackme's !!!

* 1. **Some info on packers/protectors**

INFO :

What is a packer/compressor ???

Executable packing is the process of compressing an executable file and prepending a decompression stub, which is responsible for decompressing the executable and initiating its execution.

The decompression stub is a standalone executable, making packed and unpacked executables indistinguishable to the casual user as they are not required to perform any additional steps to start execution.

Executable compression is any means of compressing an executable file and combining the compressed data with the decompression code it needs into a single executable.

When executed, this will unpack the original executable code and transfer control to it. The effect is the same as if the original uncompressed executable had been run. A compressed executable requires :

* + Less storage space in the file system
  + Less time transfer data from the file system into memory
  + More time to decompress the data before execution begins than the uncompressed original.

A compressed executable is a variation of an executable archive, where the compressed data happens to be an executable file itself.

There exist executable compressors for DOS and Microsoft Windows and other OS, these being as well command line as GUI based.

INFO :

Allow me to put all this in human language: packers do simply as their name suggests.

They 'pack' or 'compress' a program, then attach their own decrypting/loading stub which 'unpacks' the program before resuming normal execution from the programs original entry point.

I suppose it is clear that sooner or later, the unpacking stub must "give command" back to the -at that moment- unpacked main program. Packed files have several advantages and disadvantages.

First : the physical file size is usually smaller which is some benefit if you are marketing a product via internet download.

The second benefit is that packed files are resistant to the casual or newbie reverser, since a target must be unpacked or rebuilt before it can be patched.

Naturally these advantages come at a price, both run-time and commercially.

Main disadvantage is antivirus based. Many problems have arisen in this domain lately.

INFO :

An executable protector is in fact a derive of the simple packer, meaning that the protector emphasizes more or even completely on protection against RE(reverse engineering) than the simple packer.

This however brings some important disadvantages to the distributed program: first of all the size. Where packers aim to reduce the packed executable's size as much as possible, the protector often adds so much code trying to protect against RE, that the size of the total can considerably increase!

I have seen examples where the size increased to about 600% (especially for small programs) where the simple packer reduced the size to about 30% of the original!

Anyway, if you ask me: packers are very commendable while protectors are pure stupidity!

I've heard that in the "cracking" world, the top shots hope the next version of the protector they are specialized in will soon be released so that they can have another challenge and prove their skills.

Indeed, what is packed can be unpacked, what is protected can be unprotected!

There is not a single packer or protector that is unbeaten yet, at least if it has some name or fame worth bothering for.

My opinion is that programmers would better protect their own program with a custom made protection instead of all putting their trust in a couple protectors.

But then again, this is only my personal opinion of course.

INFO :

Another very important matter for the reverse engineer is that more and more malicious software (virii, worms, …) "use" the protected executable to hide themselves from detection.

It is the reason too why it is very important for the reverser to know and understand what protectors do and what they are.

This is also the reason why RE is so occupied with packers/protectors. It is also due to protectors that more and more false detections of virii come up.

And I suppose all this doesn't make it easier on the AV software companies to win the battle !

INFO :

I already explained about the entry point of a program in previous parts in this series. With packers/protectors coming in play, it is clear that the normal EP for the program will no longer be the EP for the combination of "main program/packer stub".

Indeed, the EP will become the EP from the stub, while we will call the former EP from the program the "original entry point", hereafter called OEP.

INFO :

Normally, I always first open a program in PEiD or another similar program(RDG packer Detector, etc) to verify for packers and/or compiler.

This time however, I have NOT done that. Firstly because I know what packer/protector has been used(Teddy, who packed these unpackme's also names the packer/protector), but secondly also to reduce the size of this movie.

I have shown in previous parts in this series how to do it, so go ahead and verify if you want.

INFO :

To resume what it's all about: sooner or later, the packer/protector has unpacked the program in memory.

At the moment the packer "jumps" to OEP to give command to the program, we can dump the program and we end up with a perfectly normal program.

(I'm rather oversimplifying this, but it's the main idea). Now, the next important question is : "But how can we do this ?"

In fact, when manually unpacking (MUP), there are three main ways we can follow to reach OEP to dump a program :

* + Trace the code (== step the code)
  + The "general" way by using ESP register
  1. Use exceptions generated by compressor

I'll handle each of these in this or the next Parts in this series in due time.

BTW, I'm talking about manually unpacking here. So, I have not included "unpackers" (programs able to unpack certain packers) nor Olly's SFX function which I'll explain in a later Part in this series, nor for example the most recent trend : unpacking using tracers.

REMARK :

A wrapper "wraps" itself "around" the executable, "protecting" the exe this way.

Let's finally get to work.

Let's first see an easy example of tracing to OEP

* 1. The CrypKeySDK5.7 UnpackMe

INFO :

Crypkey is in fact more protector then packer : it also adds quite a lot of code to the program to protect it from RE.

Perhaps you wonder now "Huh? Immediately starting with the tough stuff ???" Well, I'll let you decide about the protecting capabilities of this Crypkey version (5.7).

It is in fact one of the simplest I ever came across :)

Olly has noticed something is not right. Well, we'll see.

Just accept it.

We land at the EP from the packer/protector.

I told already that it's a good habit of taking an overview in the code first.

However, this time, there is not much overview to take: a very short routine followed by a return!

Step the code F8 to see what's up …

Hold on! Study the code a little better here before continuing !

And notice that this jump brings us to a complete other place (section).

Let's jump and see where we arrive :)

...we land at OEP !!!

How do I know that? Well, what we see here already gives me a good indication: this is "new" unpacked code that Olly has not had a chance to analyze yet.

That's why it is still "displayed" like data. So, analyze the code first. I'll give you some more info why I know this is OEP a little further on …

:)

That looks rather different huh ?

INFO :

OEPs from different compilers also look different. It is a good practice to remember how different OEPs look like.  
It can be very helpful in finding OEP !

This here for example is MS Visual C++

Right. So, if this is OEP, then we can dump this program to disk. Let's do that.

INFO :

Olly can not dump. However, there are some very good plugins for Olly which let us dump from Olly!

For your convenience, I have included Ollydump.dll in this package. As always: thanks to the author!

There are also many other ways to dump a program: LordPE, PE Tools, etc. Even with ImpRec you can dump a program! See in a later part in this series.

Dumping a program with this plugin is easy: the plugin fills everything in for you.

Make sure this is checked (and leave it always checked)

See that the plugin has filled in the right image base and the size

Notice that the plugin automatically modifies the EP to the OEP!

And notice also all the sections that will be dumped!

BTW, you can easily see here what all sections the protector has added to the program!!

See also further in this Part.

INFO :

Sections are named in a "conventional" way by compilers, although names are completely free

… not even required!

Teddy who has packed this even wishes us a nice day! Thanks Teddy :)

And this "protector" is so much of protector that we don't even need to rebuild imports! So, just uncheck!

BTW, see more on this in a next Part in this series.

:)

Saving as "dump.exe"

INFO :

So much for the "deprotecting" from Crypkey 5.7 :=)

If you verify dump.exe, you will see that it runs fine. Also: take a look at the "packed" unpackme: it is 627 kb while the dump.exe is ….. 444 kb in size !!! And we still can make it smaller: see further!

But I promised you to explain why I also knew I landed at OEP here. I'll show you …

Before starting this tutorial Part, I had take a quick look in the memory map(linke now).

And that's the reason why I knew I had landed on OEP!

See that we started at EP 0046B6DE in the protector's section

While OEP is in the program's code section of course ;)

:)

Also notice that the base of code is 1000 (section starting address minus ImageBase) and ….

… that the base of data is 4B000.

We will need these a little further. (0044B000 - ImageBase)

INFO :

If you don't understand this, then see Part03 in this series again. However, I'm sure all will become clearer as we proceed with unpacking.

I said we can make this dump still smaller.

Let's do that. Remember that I told you before i would come back to LordPE and show you some more when unpacking?

That time has come …

;)

Correct!

Correct!

Correct!

Correct!

Let's see the sections now!

;)

Now think with me. The protector's sections are still in the dump's image, but … they will never be run! Then indeed, why leave them here??? Let's cut them away!

:)

This will look quite better, right?

INFO :

So far, I have only removed the sections' "info" from the header. Now, I need to rebuild the executable with the new header's info to really cut the sections from the image.

:)

All succeeded fine.

Verify it and you will see that size is reduced from 444kb to 404 kb ;)

INFO :

Cutting the packer's/protector's section(s) is not always possible.

Later, we will see examples where it is not possible.

Try the dumped file : you will see that it runs like a Swiss sewing machine's engine.

I tested that too but I removed it from this movie to reduce its size because I prefer to show you another unpacking and so we suddenly land ….

…. Here

* 1. The EZIP1.0 UnpackMe

I opened the EZIP UnpackMe in Olly already. We are at EP here. I showed you to trace to OEP in the Crypkey UnpackMe, so, I'll show you the "general" way in this EZIP UnpackMe.

Just follow along and step F8

INFO :

In the "general" way, we use the fact that sooner or later, the packer needs to jump to OEP to give command to the program.

So, the OEP's address needs to be memorized by the packer for later. And this is exactly what happens at the first change of the ESP register.

REMARK :

This should be a general way, however, many packers/protectors have found a way around this !!! Although, it's still always worth to try it out in other packers!

Notice that the ESP register has changed after the above step (PUSH EBP). The method now consists in hardware breakpointing the value for ESP in dump so that we break when this value is accessed to effectively jump to OEP. Just follow along and you will understand.

:)

Doesn't matter on WORD or DWORD :)

The HW BP on access(word or dword) has been set. We can run the program to hopefully break in the jump to OEP.

INFO :

The "jump to OEP" can of course also be a call or a return :=)

Bam! We break in a JMP EAX …

… which is indeed a break in the HW BP ;)

It seems to be the same executable that was packed with EZIP as with Crypkey ;)

(same OEP)

Now if you want, scroll up till EP again and look at the passing code: you will clearly see the complete stub's code. I've done that to but removed it from this movie for size.

We could of course have used the tracing method too (you can always use that) but this way is much easier in this UnpackMe.

Now, press F8 (or F7) to jump to OEP and let's quickly dump this UnpackMe.

INFO :

See more info on this in later parts

The executable is dumped. It is not really necessary to do something extra: the dump will run fine. However, I want to show you how to make a clean dump, this time with PE Tools(Again, any similar tool will do)

;)

:)

Let's cut the unnecessary sections from the executable too. You can see two but there are three of which the last is hidden.

You have the choice in fact (header or file): either way will have the same result after rebuilding :)

INFO :

I want to verify the base of code and base of data too. You can easily see here in the sections what they are :=)

Let's see the optional header and verify !!!

Aha! Both are Wrong !!!

INFO :

This is not a real problem though. The executable would run fine. Olly would only complain at startup that it ahs found "an entry point outside the code" (as specified in the header).

However, I like to show what it should be…. So, let me also show you where to find this in Olly!

;)

:)

INFO :

For better comprehension, I'll say this in human language: the base of code is the address where the code section starts(is loaded). The base of data is the address where the data starts(is loaded).

INFO :

Remember all the stuff I talked about in Part 03 in this series…. You can see it all here(or in any similar tool of course)

All fine !!!

Run the executable in (or outside) Olly and it will run like a Swiss sewing machine engine.

The packed file was 227 kb and after unpacking 407 kb.

That's what one could call a useful packer !!! Again, I prefer to show you another unpacking(you should test it all further please) and so we land ….

번역 주)Swiss sewing이 뭔 뜻이냐?

… here.

* 1. **The eXPressor1.3 UnpackMe**

I suppose that you will start to get the point of unpacking if you have understood the previous.

So, I'll unpack this one faster but I'll explain better where necessary

If something is not clear : just review the previous unpacking.

Let's go!

Step F8 till ESP register changes (indicated by red color)

:)

Ah! Ok. I suppose you understand this is only a messagebox in the packer's code?

So, we need to bypass this to get to OEP. Right?

Bam!

You know what is going on?

Just follow along …

INFO :

Scroll up till EP : you will clearly see the code from unpacking stub. It's always nice to verify and to learn by studying what exactly has been going on …

OEP!

INFO :

This packer is a little different : you need to leave the "Rebuild Import checked. Why and how I know this?

See me about it in the next Part in this series, but it means that Olly/plugin will try to rebuild the IAT and "attach" the IAT in an extra section.

This will be explained better in a next part in this series.

INFO :

Notice that there are 3 sections that will be dumped. After dumping with "Rebuild Import" checked, Olly will add an extra section usually called ".idata2" containing the IAT.

Do not worry too much about all this yet.

:)

That's it! Job done although you can always "repair" some things in this dump if you want.

But I'll leave tat for you. Just for your info, but I hope you will try it : you can wipe the ex\_code section which contains the unpacker's code. Don't forget to rebuild the exe!

You'ld better also change the base of code to 1000 or Olly will complain :)

Meanwhile, take a look at the idata2 section that has been added by Ollydump plugin.

REMARK : I'm not showing this because there is difference with what we've seen before, and so we land ….

… here.

* 1. **The MEM1.1 UnpackMe**

MEW1.1 is freeware : it has been my favorite packer for a very long time. MEW offers very good compressing ratio's and it packs almost anything while it is easy to unpack (if necessary to take another look in your code).

By now, I suppose you know what to do to unpack "the general way". However, for MEW, you can also just scroll down and find the end of the packer's stub : the return is the "jump" to OEP. Follow along.

INFO :

Try always to look, try, test and learn. For example: take a look at this jump: where is it leading to? Indeed, an address in the header!!

Conclusion : the unpacking stub will be situated in the header and there will be NO section to wipe after unpacking!

This is also one of the reasons why MEW can keep sizes so small !!!

I hope you understand that wiping the header from the executable might have some disastrous effects ;)

INFO :

Remember this for later : in the header is almost always "empty" space (code cave).

We will use that later when inline patching and writing the patches in … the header.

Scroll down to find the return at the end of the stub.

REMARK : you can also unpack MEW in the "general way". Just give it a try!

Hehe. You wonder what this is? Never mind. Olly just warns us that I'm wanting to set a BP in the header.

Let's comfort Olly and calm him down :))

It seems Olly trusts me and sets the BP all right :)

:)

Ok. Now press F8 and we land …

… in the OEP.

See that there are only 2 sections !

But there will soon be a third section because you'll need to keep the "Rebuild Import" checked when dumping.

Ok. I'm assuming all the rest is understood. Now dump and study but I won't show it because … I want to show you another unpacking, and so … we land …

… here

* 1. **The NsPack3.5 UnpackMe**

INFO :

NsPack is a relatively good packer with about the same packing ratio as MEW.

Let's quickly unpack this UnpackMe.

The OEP is "hardcoded" in NsPack 3.5

:)

Done ! The dump will run fine. Let's go to the next UnpackMe!

Hence, we land …

… here.

* 1. **The Noname UnpackMe**

Well, I have no idea what this one is.

Let's go slow and first take a look ..

Huh? Aha. Does this ring a bell?

I closed Olly and ran the UnpackMe outside Olly ---> completely different behavior. Have you seen part#19 in this series?

Debugger detected indeed, no doubt about that. Let's find where …

… and so we land here on EP again.

BTW, removed testing from movie for size.

Start exploring and take an overview. Start stepping F8

INFO :

Have you noticed I have scrolled down a lot, passing all the calls and calls without (cond) jumps? I will now BP here and try to find out if the problem is before or after the BP.

Ah! It seems good so far!

INFO :

This is not sure though: remember that the detection may have happened long before the action is taken !

Continue stepping F8

Mmmm. You certainly think what I think. Right? (See part 19 in this series)

:(

Do you remember this is not good?

Let's help the UnpackMe in the right direction ;)

:)

:)

Mmmm. A loop. I tried it but the code keeps coming back here, running this small loop in a bigger loop. (I don't want to go too deep in what exactly is going on).

Just think with me now : sooner or later, the command must be given back to the program at OEP. So, what if setting a memory BP on code section ?

We will break as soon as the code section is accessed !!!

BTW, of course this only works if the code is already unpacked, else one keeps breaking while unpacking(and thus accessing) the code. Let's try it.

BTW, I could also have set a normal BP (F2)

;)

And notice that I want to break in the code section!

Try it!

OEP!

All right : that's it then.

Dump like this and you ….

… can cut the .Prt section afterwards.

Rebuild the dump and all will be fine.

* 1. **The Exe32pack1.42 UnpackMe**

In the previous unpacking, we have seen that packers can also use anti-debugging techniques to hinder unpacking.

This Exe32 UnpackMe does that too. IMPORTANT : I want to show you some unpacking without the need for anti-anti-debugging all the time first.

For the reason, I have included some anti-anti plugins. For this particular unpacking, only a "cure" with Re-pair.exe (version 0.60) on disk suffices already.

Remember : I will not look into this anti-anti anymore : let's concentrate on the important things in further parts and make sure you have protected Olly in the best way.

This UnpackMe is again an easy example of the "general" unpacking method. I think you know the way meanwhile. Just follow along but Olly's analysis is not so clear.

Let's remove analysis first (and manually analysis when necessary)

ESP has changed!

JMP EAX to OEP

Ah ! OEP will be 40A000

OEP!

INFO :

Perhaps you wonder why I'm unchecking "Rebuild Import". And what it is? Allow me to keep this for the next Part in this series.

Let's not worry about this yet….

The unpacked executable runs fine. Of course you can also rebuild with a PE editing tool. (See before).

* 1. **The Fusion 4.0 UnpackMe**

We land here on the EP of the fusion UnpackMe for the last easy unpacking in this Part20. Also this UnpackMe can be done in the general way if you don't wan to trace it.

Just follow …

:)

OEP !

Again, I will uncheck "Rebuild Imports" because it is not necessary to rebuild imports(in this case)! Like said before, I will explain more on this in the next part in this series.

The unpacked executable runs fine. Again, you can rebuild if you want. (See before).

INFO :

Don't think that every unpacking is as easy and fast as shown here… On the contrary, unpacking is an art, one can amuse himself(herself) for hours with it but it is really appealing/tempting.

I understand that in the beginning, it may all seem rather confusing, but after a while, unpacking is big, big fun!

* 1. **Conclusion**

In this part 20, the primary goal was to learn something and practice about packers/protectors/compressing/unpacking while unpacking 8 UnpackMe's.

I hope you understood everything fine and I also hope someone somewhere learned something from this. See me back in part 21 ;)

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)

2012-02-08 오후 10:55