Unpacking With Tracers II

+NCR/CRC! [ReVeRsEr] of ARTeam

Version 1.0 – January 2006

Keywords

Tracers.

1. Abstract

Hi folks!! How are you doing?. I'm back again writing a tutorial but this time with a goal that I hope to be able to achieve sometime in the near future. I pretend to follow (if he allows me to do so) the steps of the Master **AkirA**. This guy, a good friend of mine, cracker, programmer and researcher has opened to us a wide range of possibilities we must take advantage of. Everybody in CracksLatinoS! 2005 know who I am talking about, but some people know him specially for his excellent work with Xprotector and Themida.

AkirA and his tracers have given us many new possibilities when it comes to face our beloved packers, either new ones or not so new ones. Up to now, the tracers are the best tool to do 70%, if not more, of the work. In this "short" tutorial we'll see how to blow up a packer and how to play with it. I can't remember if it was RLP, Hygyn's, SCP or PE Mutilator, but it's a packer nonetheless with an easy redirection of the import table. But the important thing is that we'll see how to use the tracers and how to apply them to diferent cases and we'll realize how they make our life easier, specially with the import table (IAT).

Before starting, as AkirA once said, the tracers are useful with most of packers nowadays, even the new ones that come out, 'cause few programmers now them yet. But if we start using them and make them known, we'll see that things will change. It's a fact that packers are more and more complicated, with more anti-debug and anti-dump tricks, redirections of the import table almost impossible to figure out, and many more difficulties. But with this tool, things are much easier.

Well, enough said, let's start working and see what happens...

As usual I will provide sample code with this tutorial, and non-commercial sample victims. All the sources have been tested with WinXP and Visual Studio 6.0 (Visual C++).
The techniques described here are general and not specific to any commercial applications. The whole document must be intended as a document on programming advanced techniques, how you will use these information will be totally up to your responsibility.

2. Working With Tracers

First of all, we'll define the basic idea of a tracer, which is the name that AkirA gave to them.

A tracer consists basically of an inyector (exe) and a library (dll).
The inyector creates a suspended process, which in our case will be the packed program and then it inyects a remote thread in that process. That remote thread is nothing but our DLL which has a hook. That hook can intercepts functions which can be the usual ones or even Zw functions or a ntcall. We can even dump sections, log any instructions we want, etc. All in all, we can do whatever we want with to the program once we have it under control, thanks to the hook.

I don't pretend to explain all the theory here, you can study AkirA's tutorials for that matter. I've only used them and tried the ideas shown there to see if they worked. And they did! They're pretty powerful.

Those who want the whole theory about tracers should read the tutorials here: www.iespana.es/OllyDbg under AkirA's section, all of them are there, the one about Xprotector, Themida, ApiWrapper, and the rest of them.

In this tutorial we'll use the type 4.c tracer, because the type 4.b had a bug when working with kernel32.dll; which was solved by **kaos_xlro**.

We'll also use a couple of crackmes, one packed and the original without packing so that we can compare to see if things are going well. Our main goal isn't unpacking but to show how well the tracers work and what a fantastic tool they are to fight packers. They can even be used with programs that aren't packed, but that's another story.

Now, let's go. First of all let's see how we would unpack a program without using tracers. We load the program in our OllyGhost modified with the patches for VP, plugins for anti-debug,... Well, what we always use in unpacking tutorials. I've said this many times, so I hope you know what I'm talking about.

Figure 1 – Error on PE-HEADER.

Soon after the start, we see this warning message in Olly that prevents us from loading the program. As you can see, it stops at System StartUp Breakpoint:

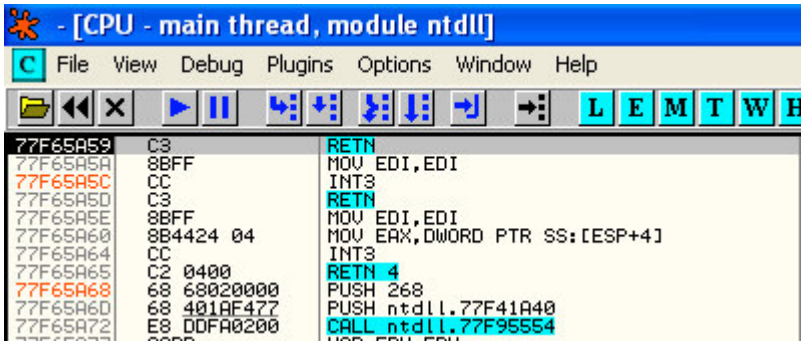

Figure 2 – System Start Up Breakpoint.

This happens because the value in NumOfRvaAndSizes has been modified on purpose by the packer itself. In general, in many packers this value is used to decrypt the last portion of code before jumping to the OEP and perform a CRC32. In this case, the value has been changed to 40h:

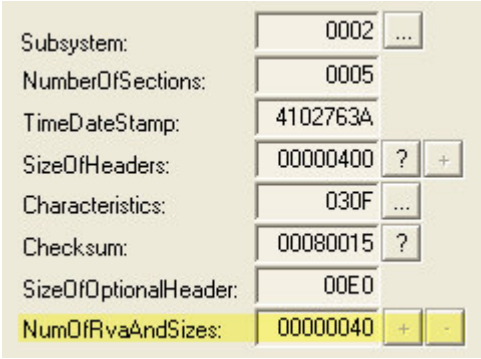

Figure 3 – NumOfRvaAndSizes.

The default value in Windows executables is 10h and every program, suposedly, should have that value. But as I said, sometimes this value is changed with evil purposes in mind ;)

If we also look at the MemoryMap, we'll see that many of the sections of the program haven't been loaded. Only the PE-HEADER and another section:

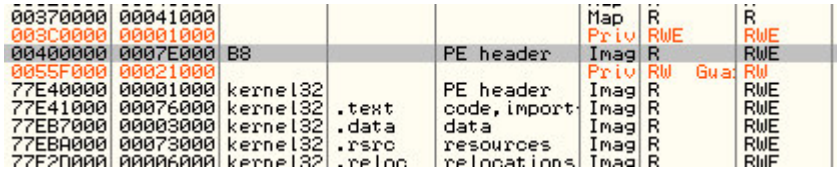

Figure 4 – Sections & PE-HEADER.

And if we open the original crackme with a PE Editor, we see that there are a few more sections that should be loaded:

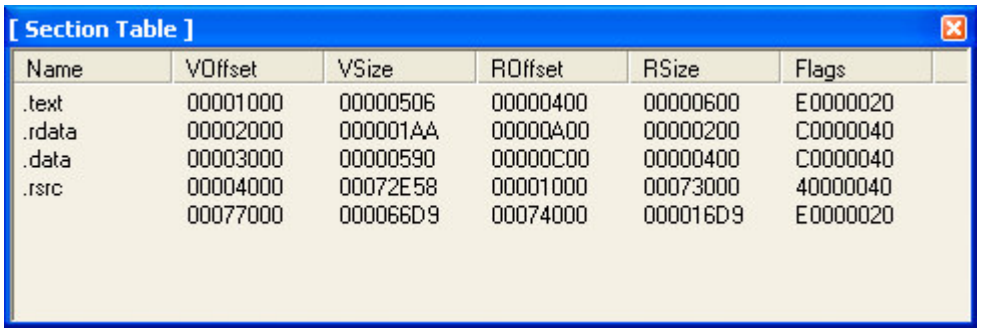

Figure 5 – Sections.

The view is too fuzzy by now. What can we do? Well, I, personally, would start trying things and perhaps do some unorthodox thing to unpack the crackme. Many of us have done this sometime, but today we will resist the temptation and we will try this new tool we've heard about some time ago that maybe we don't fully understand but we know that it works and that's enough. At the end of the tutorial you will be able to judge if it was worthwhile or not.

Before using our tracers, let's try something. Let's change the value in NumOfRvaAndSizes to 10h and load the Olly again.

First we change the value using any PE Editor like LordPE:

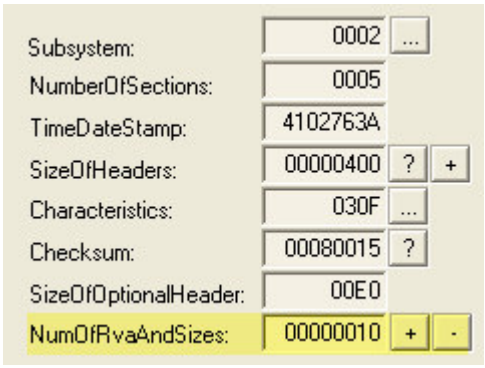

Figure 6 – Correct NumOfRvaAndSizes.

Now we load the program in Olly and we see the usual warning we like so much:

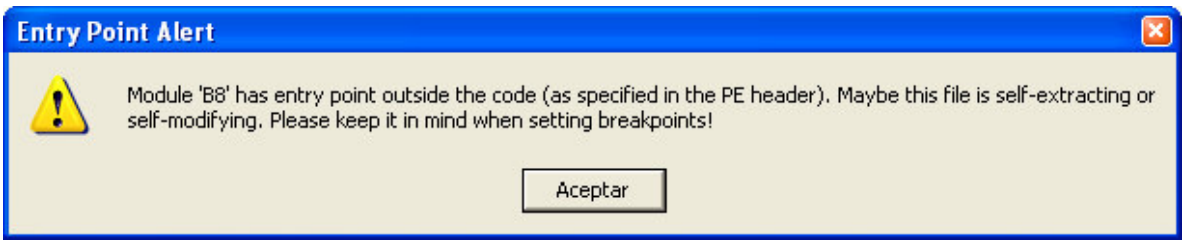

Figure 7 – EntryPoint Outside the code

.

You missed it, didn't you? XD.

And then this annoying message pops up:

Figure 8 – Empy code section

We press <Aceptar> and now it seems we're at the EP:

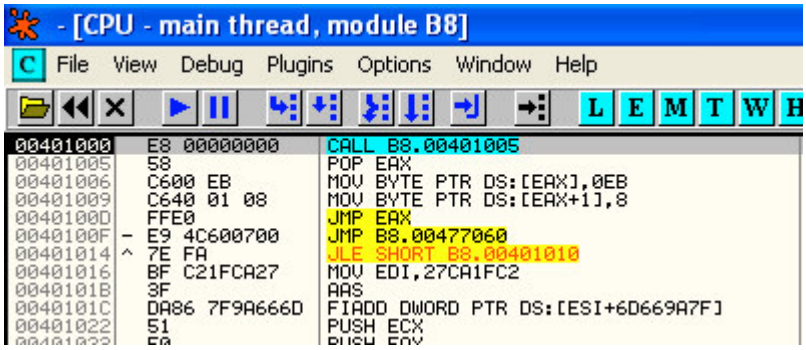

Figure 9 – EntryPoint.

I configure my Olly so that it doesn't stop at the exceptions:

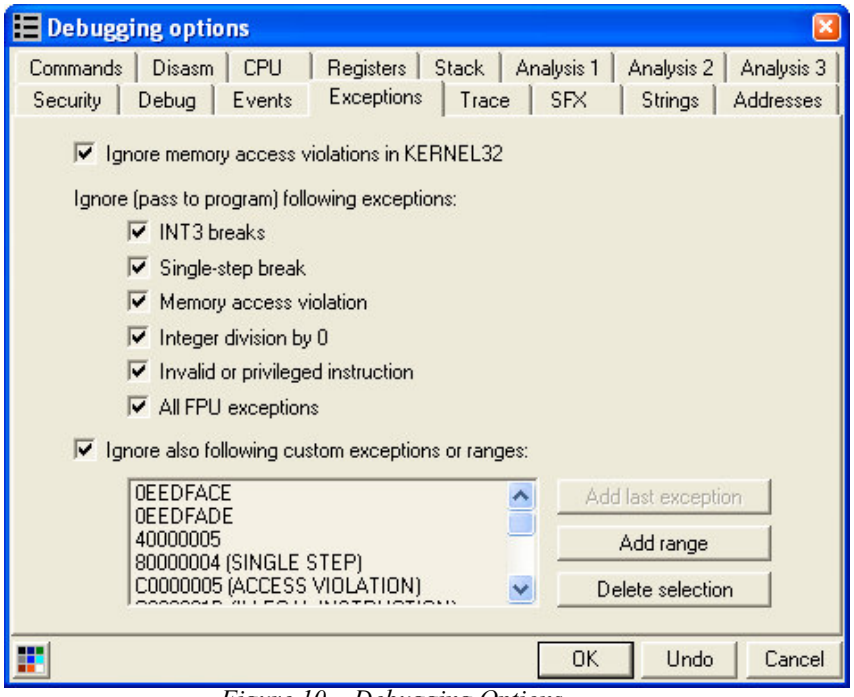

Figure 10 – Debugging Options.