Unpacking With Tracers I

+NCR/CRC! [ReVeRsEr] of ARTeam

Version 1.0 – January 2006

Keywords

Tracers.

1. Abstract

Hi you all!!!. This is my first tutorial for a non spanish group. So forgive me if something's a little difficult to understand. Besides, this is my first tutorial for such an important and well known cracking group as **ARTeam**.
Well, sorry but I haven't introduced myself. My name's **+NCR/CRC! [ReVeRsEr]** and I'm a member of CracksLatinoS! 2005. Many of you surely know that cracking list, by one of its founders, the Master Ricardo Narvaja. He, besides CracksLatinoS, has also been a member of lists like **[aRC]**, **[RVLCN]** and **PDAToolBoxHispano**.

Well, I hope my nerves won't let me down and that the work will be interesting enough for you.

In this tutorial we'll do some unpacking using Tracers, a new technique developed by **AkirA** (a friend of mine and author of tutorials about Xprotector and Themida, You surely know him ;) and now implemented by yours truly.

It'll be an easy unpacking, but what's important are the tracers.

Before commmencing I warn you that my way of working will be the same as in CracksLatinoS. The ones that already know me are surely used to it. And for the new ones, I hope you find it interesting.

Now, to the tutorial...

As usual I will provide sample code with this tutorial, and non-commercial sample victims. All the sources have been tested with Win2XP 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

As I said before, we're going to perform an easy unpacking of a packer that completely destroys the IAT as well as the table of jumps. We'll automatize things very little 'cause our objective is to show the technique and besides, all these things are still in a testing phase since I asked **kaos_xlro** to make some changes in the source programs. So it's still a long way to go ;)

This is my second tutorial on tracers, the first one was written for CracksLatinoS and if anyone wants to download it, you can go to Ricardo Narvaja's FTP or just drop me a line.

Let's start by introducing the program, this is my Guinea pig:

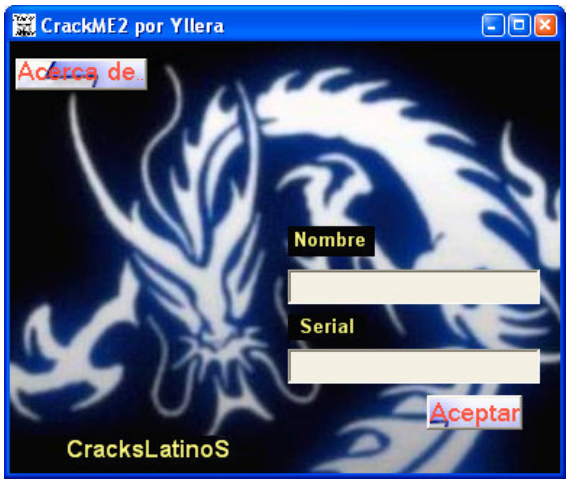

Figure 1 – My guinea pig =).

First of all, let's use the **RDG Packer Detector** to identify the packer:

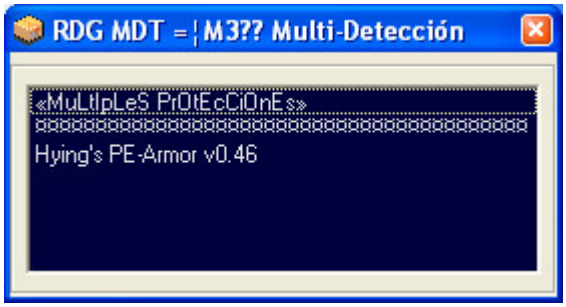

Figure 2 – Hying's PE-Armor v0.46.

It tells us that it's Hying's PE Armor v0.46. Now, let's try with PEiD to see if we can find the OEP:

Figure 3 – The real OEP?.

Well, it seems that this packed program's OEP is at 401000. Let's check it out with Olly.

I open my OllyDbg, with the patch for VProtector, Anti-CAPTION and CLASS WINDOW and the HideDebugger plugin activated with all its options, and I load the program:

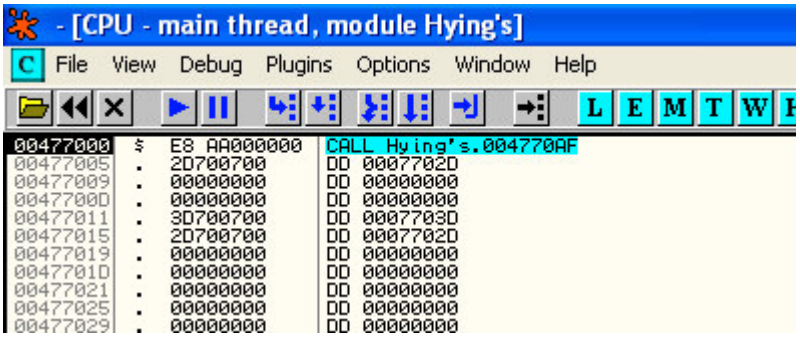

Figure 4 – EntryPoint.

There you can see the packer's EntryPoint (EP). Now, let's try to get to the OEP. But first, let's configure Olly in the following way: We go to Options – Debugging Options – Exceptions and we only check the 'Ignore Memory Access Violations in KERNEL32.dll' box:

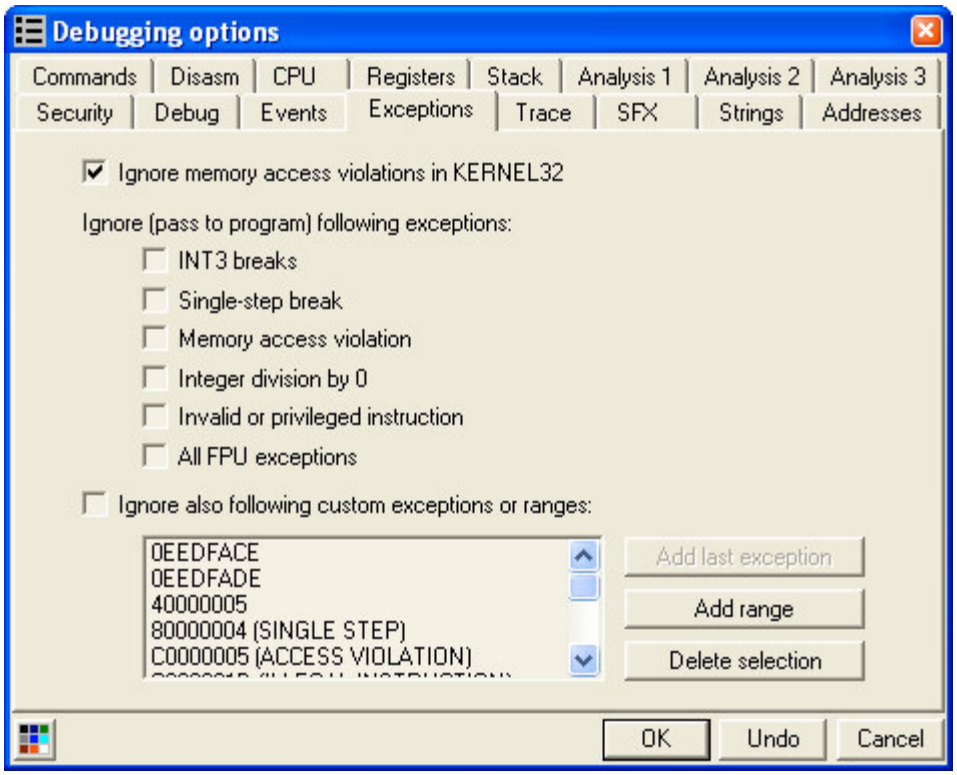

Figure 5 – Configuring OllyDbg.

Then, I press F9 (Run) and an exception occurs:

Figure 6 – Exception.

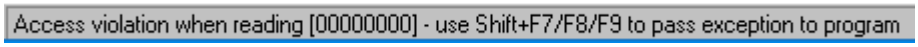

Figure 7 – An Access Violation.

In order to pass the exception we only need to press <Shift+F7> and then F9 again. Another exception occurs and I again pass it by pressing <Shift+F7> followed by F9. If we do this several times, we can see that many more exceptions occur. So we repeat the above process until the program is running (it hasn't any anti-debug tricks). In that very moment, I press the minus sign key ('—') to go back in Olly and I see what the last exception was (befote the program run) so that I can write a script later on with OllyScript that will leave us in that last exception without working too much:

```
start:
eob break
run

break:
cmp eip,581fe8
je final
esti
jmp start

final:
ret
```

So we run the script and the program stops just in that last exception:

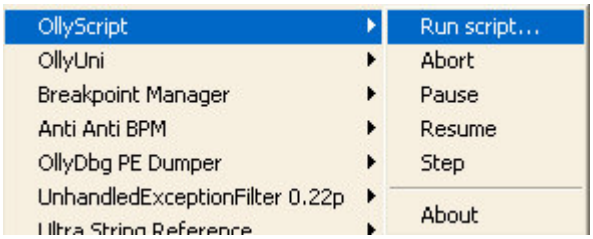

Figure 8 – Running a script.

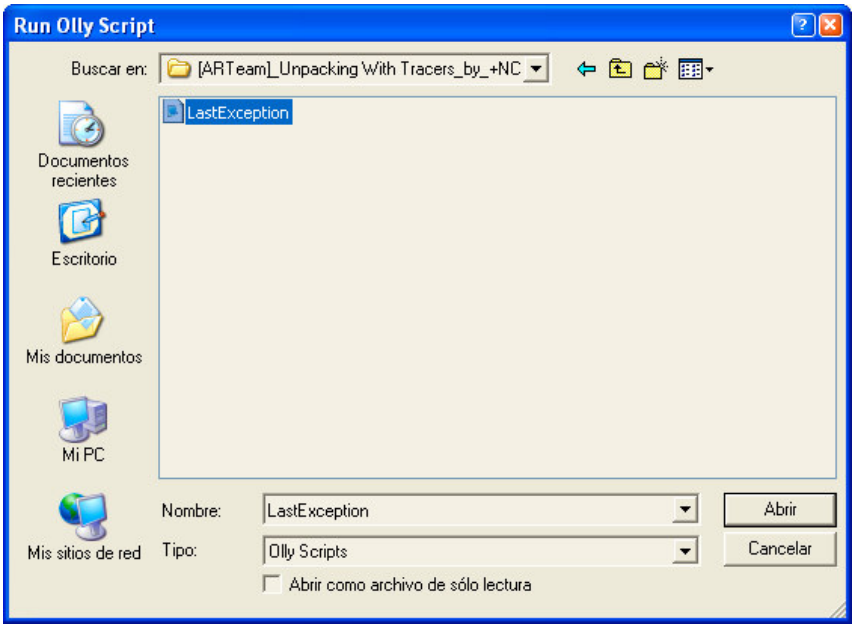

Figure 9 – Selecting the script.

And soon after, we get the warning telling us that the script finished its work:

Figure 10 – Script finished.

And we are at the last exception, that in my case is:

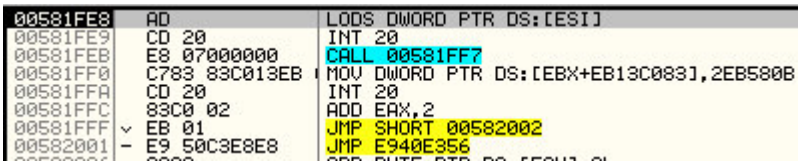

Figure 11 – My last exception.

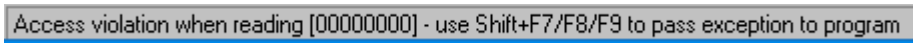

Figure 12 –An access violation.

I pass this last exception with <Shift+F7> and then I go to the Memory Map by pressing <Alt+M> in order to put a BPM on Access in the code section of the program, the one just below the PE-HEADER: