Unpacking Eziriz .Net Reactor 3.9.8.0
by CodeRipper / SND

31-03-2009

Tools used: Olly, CFF Explorer
You should use .NET Framework 3.5

If you don't use .NET Framework 3.5 the dump will have some errors because StandAloneSig tables is fucked - these are in .text section (first section) before string 'BSJB', usually the file offset of first crap to fix is 1050, and the error message will be:
Exception occurred: "System.MethodAccessException: Signature for the entry point has too many arguments." or "Bogus local variable signature (0xACBD0000)" in IDASM.
At run-time, the signature is checked to confirm whether it has only one parameter or not. Since there are two parameters in the entry point function, the run time exception has been generated. If there had been a single int parameter, no exception would have occurred at run-time.
As a conclusion this is because the type of method (number of parameters, type of them and the type of returned value) don't have the right value – not matches the IL code which runs!

This tutorial will teach you how to unpack even .NET REACTOR 3.9.8.0

Intro:

.NET Reactor provides complete protection for your sensitive intellectual property by converting your .NET assemblies into unmanaged processes which cannot be understood as CIL, and which no existing tool can decompile. Hackers have no access to any intelligible form of your source.
The native code wall created by .NET Reactor between the hacker and your source includes industry leading NecroBit technology, which is exclusive to .NET Reactor. NecroBit protection has never been broken since the first release in 2004.
(This is from the help of program)

:-)
Don't be a lamer - this protection is easy to crack (this is my first tutorial on .NET Unpacking).

I. Dumping part:

- load the file in Olly, start the program
- go to memory map, select first memory address, search the UNICODE string "Assembly Version", the block of memory where you find this string should start whit "MZ".

In my case was like this:

```
...
00F20000   00055000          Map    R       R
\Device\HarddiskVolume1\WINDOWS\Microsoft.NET\Framework\v2.0.50727\mscorrc.
dll – after this one
00F80000   00001000          Priv   RW      RW
00FA0000   00002000          Priv   RW      RW
00FB0000   00003000          Priv   RWE
00FC0000   0000E000          Priv   RW      RW
00FD0000   00005000          Map    R       R
00FE0000   00041000          Map    RW      RW
01030000   00041000          Map    R       R
0117B000   00001000          Priv   RW  Guar  RW
0117C000   00004000          Priv   RW  Guar  RW
01180000   00003000          Priv   RWE
01190000   00014000          Priv   RW      RW
01290000   00035000          Map    RW      RW ; this one;
                                                ; has always Map, RW properties
012D0000   00030000          Priv   RW      RW
01300000   00001000          Priv   RW      RW
01310000   00001000          Priv   RW      RW
01390000   00002000          Priv   RW      RW
013A0000   00007000          Map    RW      RW
01480000   00001000          Priv   RW      RW
014A0000   00002000          Map    R       R
014B0000   00182000          Priv   RW      RW
024B0000   000DF000          Priv   RW      RW
0FFD0000   00001000  rsaenh  Imag   R       RWE ; before system dlls
0FFD1000   00021000  rsaenh  Imag   R       RWE
0FFF2000   00003000  rsaenh  Imag   R       RWE
0FFF5000   00001000  rsaenh  Imag   R       RWE
0FFF6000   00002000  rsaenh  Imag   R       RWE
```

Just save the memory area to a file whit exe extension and you should see the icon of program, if not seach again for the UNICODE string "Assembly Version"

II. Fixing after the war

1. In common cases we must fix these:

- The characteristic of file is set as “File is a DLL”
  Go to Nt Headers->File Header->Characteristics and unmark “File is a DLL”
- Inside .NET Directory members Metadata RVA and Metadata Size are wrong
  For Metadata RVA we search inside file the string ‘BSJB’, convert file offset of him to RVA and enter this as Metadata RVA.
  We calculate Metadata Size after formula
  Metadata Size = Import Directory RVA – MetaData RVA

You can find Import Directory RVA in Nt Header->Optional Header-Data Directories-> Import Directory RVA and you should notice that we don't need the exact value for Metadata Size (can be bigger)

- Inside .NET Directory cb should be 048, also:
  MajorRuntimeVersion = 2 and MinorRuntimeVersion = 5 for .Net Framework 2.0
  MajorRuntimeVersion = 2 and MinorRuntimeVersion = 0 for .Net Framework 1.1

2. Fixing „Invalid number of data directories in NT header." error

.NET Reactor makes some Fields in the Optional header invalid to prevent opening the exe in Reflector.
Fix it whit CFF Explorer: go to Optional Header and set NumberOfRvaAndSizes to 010

3. Fixing „Module '…' contains zero or multiple module definitions." error

This will also prevent loading in Reflector. This happens because inside Tables we have 2 modules while we should have only one (the second one is bad)!

Module Table it's a one row table representing the current assembly.
Columns:
• Generation (2-byte value, reserved, shall be zero)
• Name (index into String heap), can be a word or a dword
• Mvid (index into Guid heap; simply a Guid used to distinguish between two versions of the same module)
• EncId (index into Guid heap, reserved, shall be zero)
• EncBaseId (index into Guid heap, reserved, shall be zero)

What must be done:
A. We load the file inside CFF Explorer, and we go at Metadata Stream->Tables->Module. We go at second module (the one which must be cleaned) and we note the offset of first member **(**I've note this **value1)**
We go at Nested Classes and at last member from this, we add 2 (the size of this member) to offset of him and we keep this value **(**I've note this **value2).** This is the raw offset where Tables ends.

The module table can have the size: (I've note this size_of_module_table)
- 0C if Name is an dword (in most cases)
- 0A if Name is an word

We have two options for removing bad module table:
a. We load the file inside WinHex and we copy the hole block between value1 and value2 at the raw offset (value1-size_of_module_table)
or b. We load the file inside WinHex and we add zero bytes equals whit the size of module table (0C) at raw offset value2 (at the end of Tables).
We delete the table of second module (the table of course has the size 0C).

B. We search for the ASCII string '#Blob'
At start of string '#Blob' +020h is the number of modules (the old value is 2) - set number of modules to 1

Now you can load the file in Reflector!

This is all.
I hope you enjoy reading this.

Greeting:
Kurapica [Black Storm], Ufo-Pu55y [SnD], rongchaua, LibX [Ret], ColdFever, MaDMAn__H3rCuL3s [ARTeam], tKC, Newbie_Cracker [UnREal RCE], Kwazy Webbit [RETeam], ArTeam