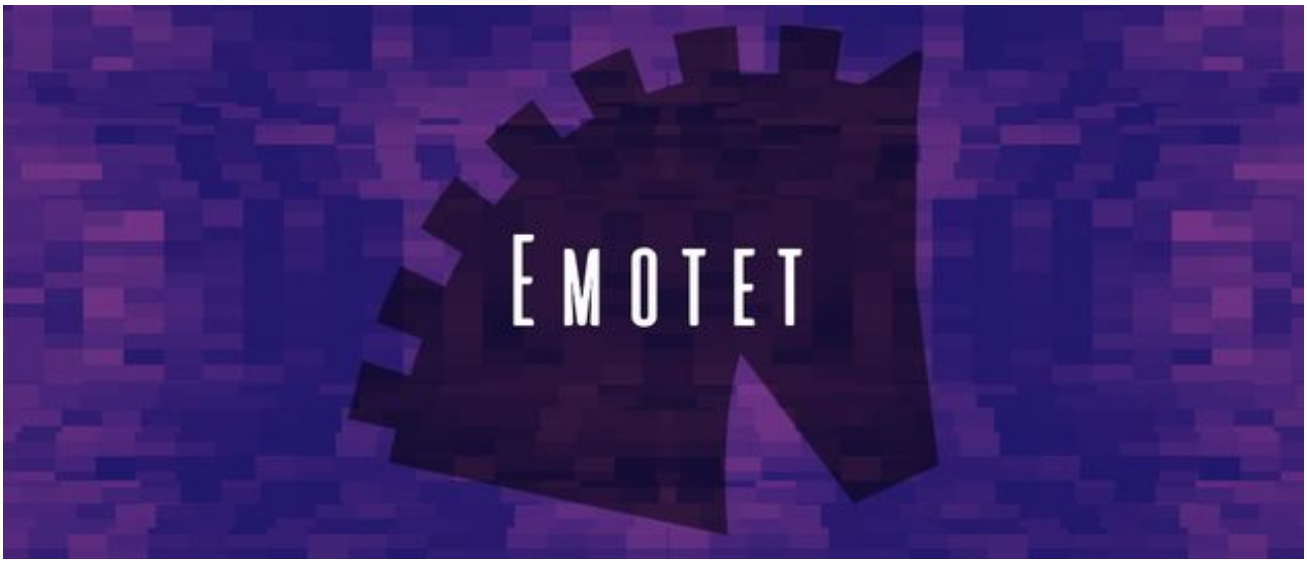
# Emotet Analysis Part 1: Unpacking

pl-v.github.io/plv/posts/Emotet-unpacking/

Player-V April 2, 2022

Player-V on Apr 2*2022-04-02T15:26:00+08:00*

Updated Apr 10*2022-04-10T22:41:15+08:00* 3 min read




## Introduction

That's will be my first post in the blog, i will make a series of posts about Emotet.

Emotet is a Trojan that is primarily spread through spam emails (malspam), we're going to digg deep in the anlysis of this Trojan, the first part is about unpacking the malware then we will try to analyse the different modules and techniques used by the malware to compromise a machine, so fire up your virtual machine and let's start.

## Triage

The first thing i always do before opening a sample in `IDA` or `Xdbg` is opening the binary first in a hex editor, in my case i will use CFF Explorer, so opening the sample in CFF explorer shows that we're dealing with 32 bit binary.

Emotet.dll

| Member | Offset | Size | Value | Meaning |
|---|---|---|---|---|
| Machine | 0000010C | Word | 014C | Intel 386 |
| NumberOfSections | 0000010E | Word | 0004 | |
| TimeDateStamp | 00000110 | Dword | 6194732D | |
| PointerToSymbolT... | 00000114 | Dword | 00000000 | |
| NumberOfSymbols | 00000118 | Dword | 00000000 | |
| SizeOfOptionalHea... | 0000011C | Word | 00E0 | |
| Characteristics | 0000011E | Word | 2102 | Click here |

Let's check import section, the malware use only one library which is `Kernel32` that's the first sign which indicate that we're dealing with packed binary.

Emotet.dll

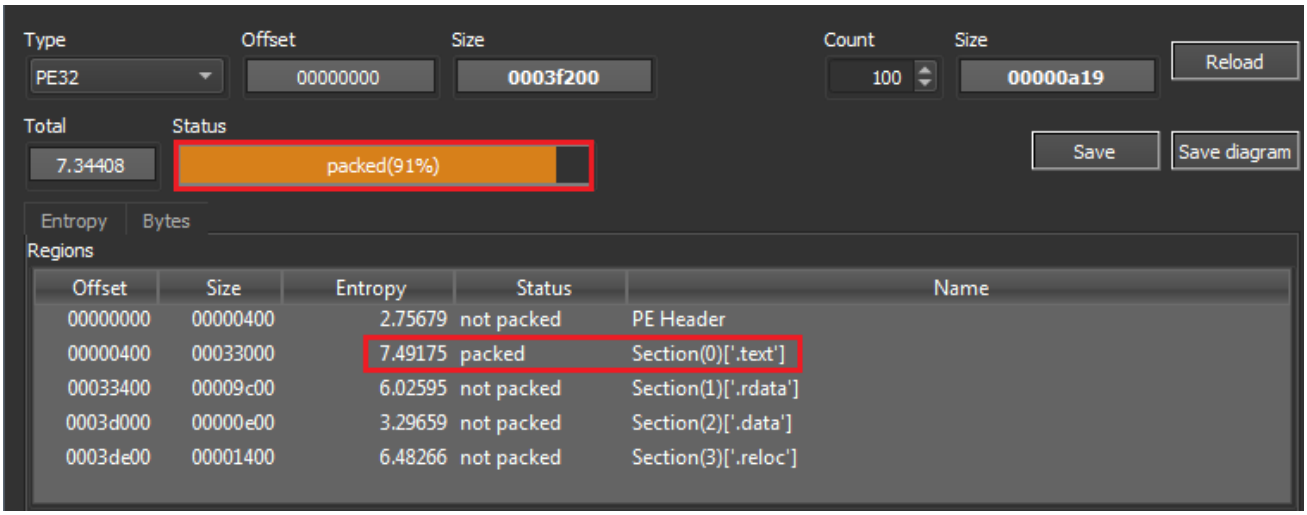
| Module Name | Imports | OFTs | TimeDateStamp | ForwarderCh |
|---|---|---|---|---|
| 0003CB7E | N/A | 0003C8CC | 0003C8D0 | 0003C8D4 |
| szAnsi | (nFunctions) | Dword | Dword | Dword |
| KERNEL32.dll | 69 | 0003D4F4 | 00000000 | 00000000 |

| OFTs | FTs (IAT) | Hint | Name |
|---|---|---|---|
| | | | |
| Dword | Dword | Word | szAnsi |
| 0003D60C | 0003D60C | 05E0 | VirtualAlloc |
| 0003D61C | 0003D61C | 05E6 | VirtualProtect |
| 0003D62E | 0003D62E | 02BD | GetProcAddress |
| 0003D640 | 0003D640 | 03D7 | LoadLibraryA |
| 0003D650 | 0003D650 | 0461 | QueryPerformanceCounter |

Two intersting API functions are used:

1. VirtualAlloc
2. VirtualProtect

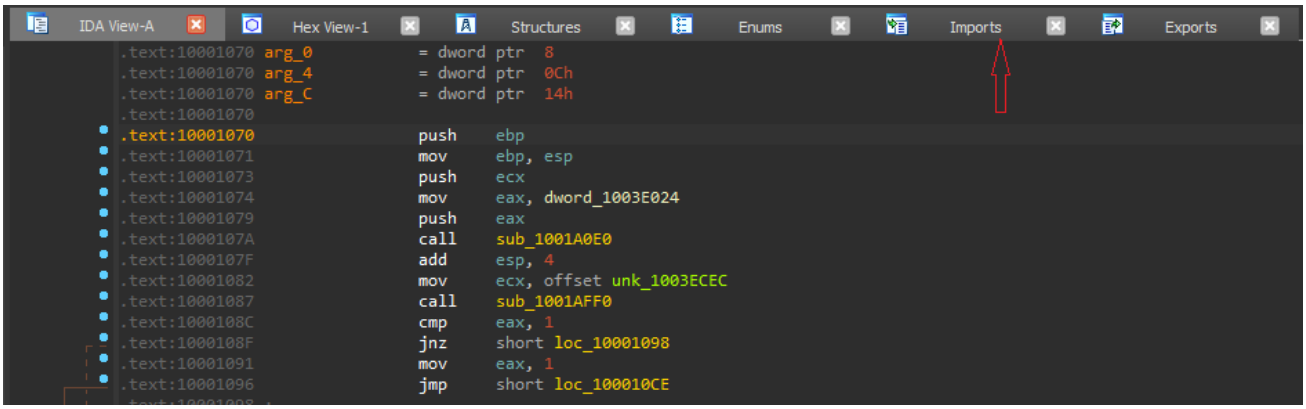
The VirtualAlloc function allocate memory while the VirtualProtect function changes the protection on a region of committed pages in the virtual address space, most of time those two functions are used by malware during the unpacking process. To make sure that our sample is packed Let's open the binary on Die(Detect-It-Easy).



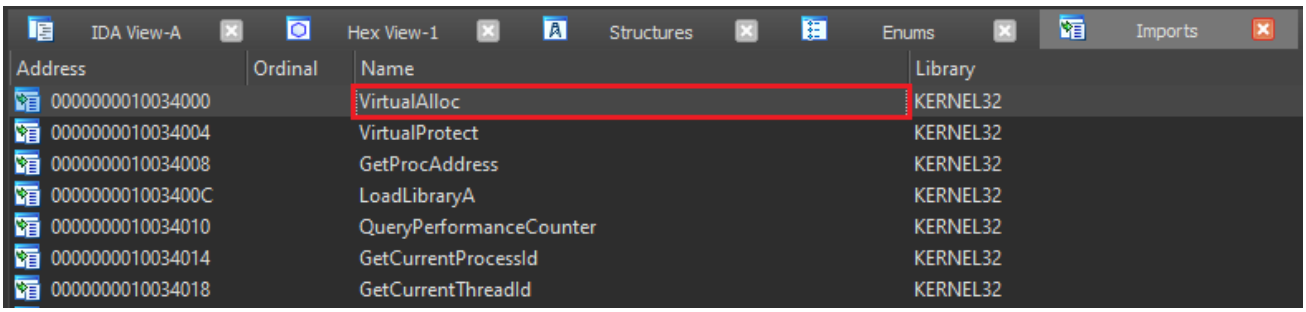

The status bar says that it's 91% packed and `.text` section has a high entropy, that's a strong indication that the malware is packed and we should unpack it for further analysis.

## IDA

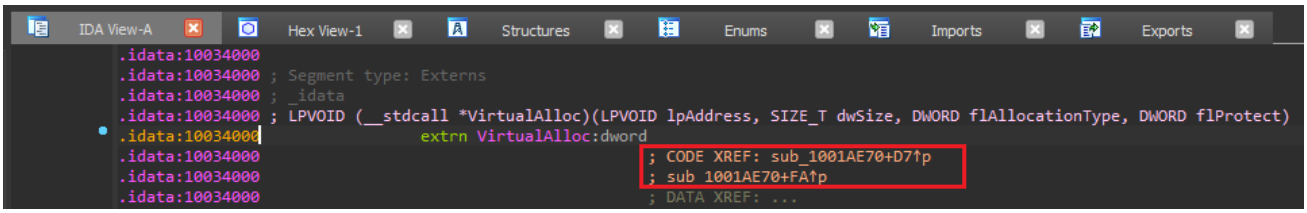
Now that we're sure that our sample is packed, let's open it in `IDA` and try to find the function which is responsible for unpacking.




Click on `Imports` to reveal all the functions used by the binary.



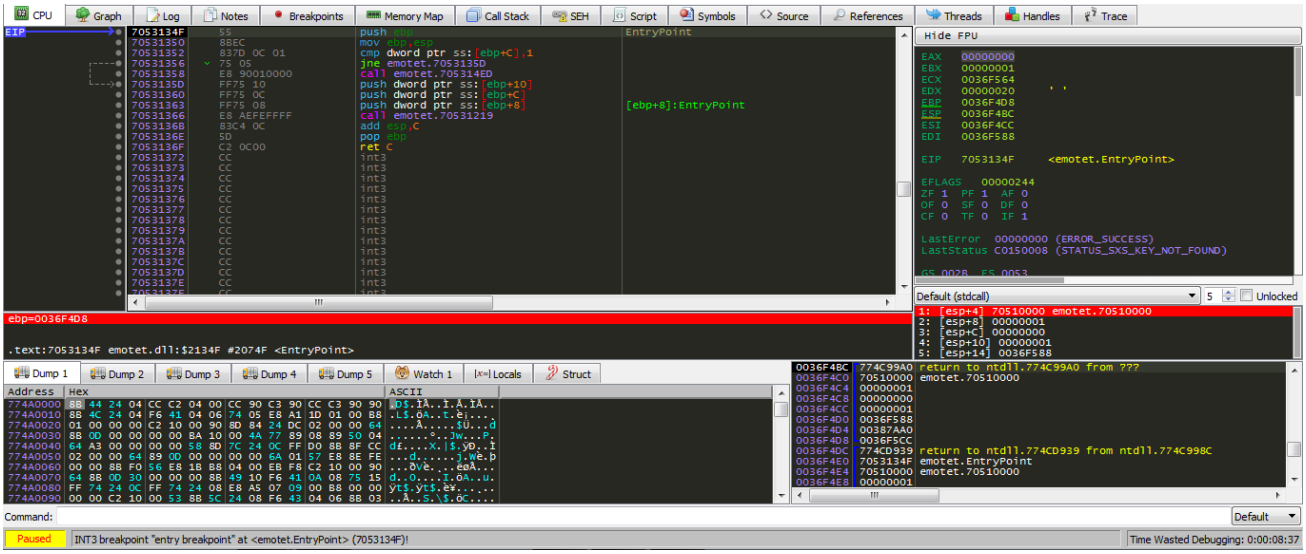

Search for VirtualAlloc and double click on it.



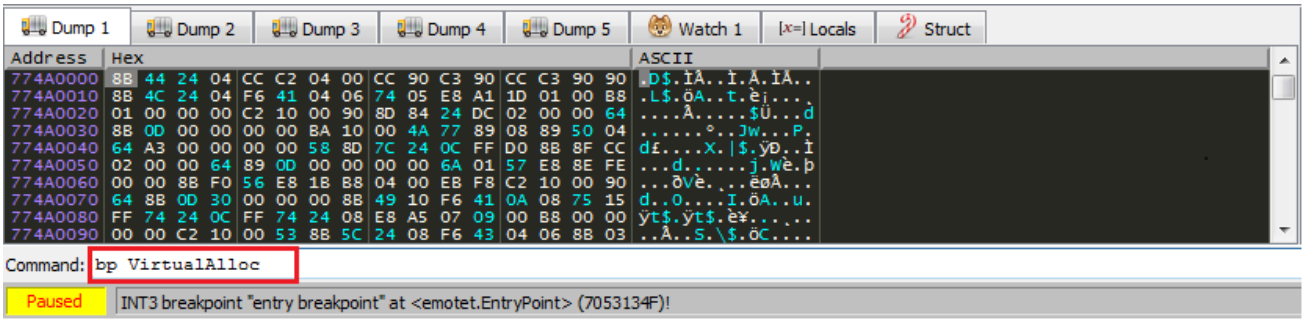

VirtualAlloc function is used two times by the same function `sub_1001AFF0`, double click on `sub_1001AFF0` and scroll down we notice that the first function called after VirtualAlloc is `sub_10022C40`, so maybe we've found our unpacking function. to make sure let's open it on `Xdbg` and figure out.

## Unpacking

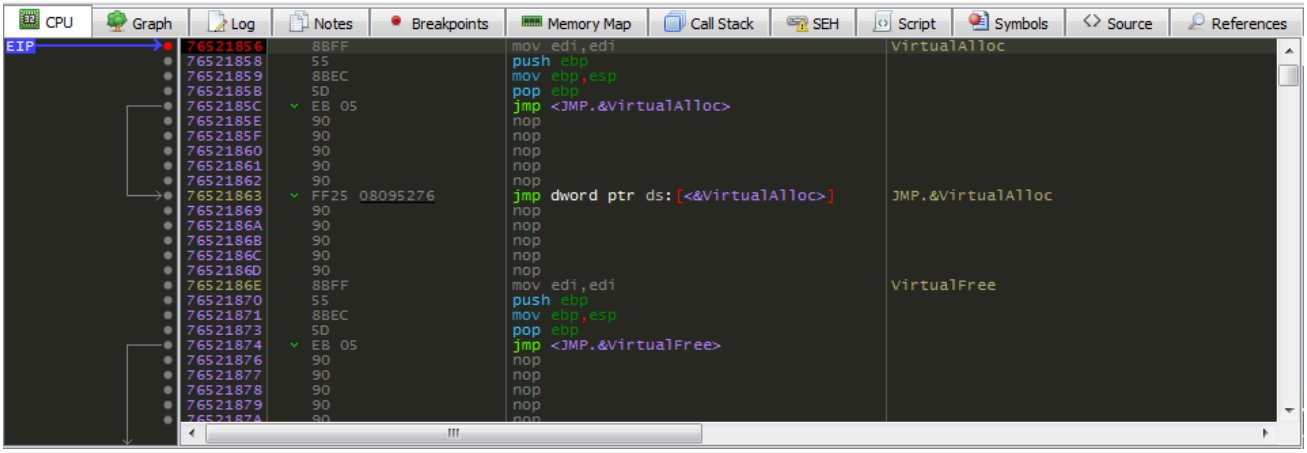
Open your `X32dbg` and paste and paste your sample to it.



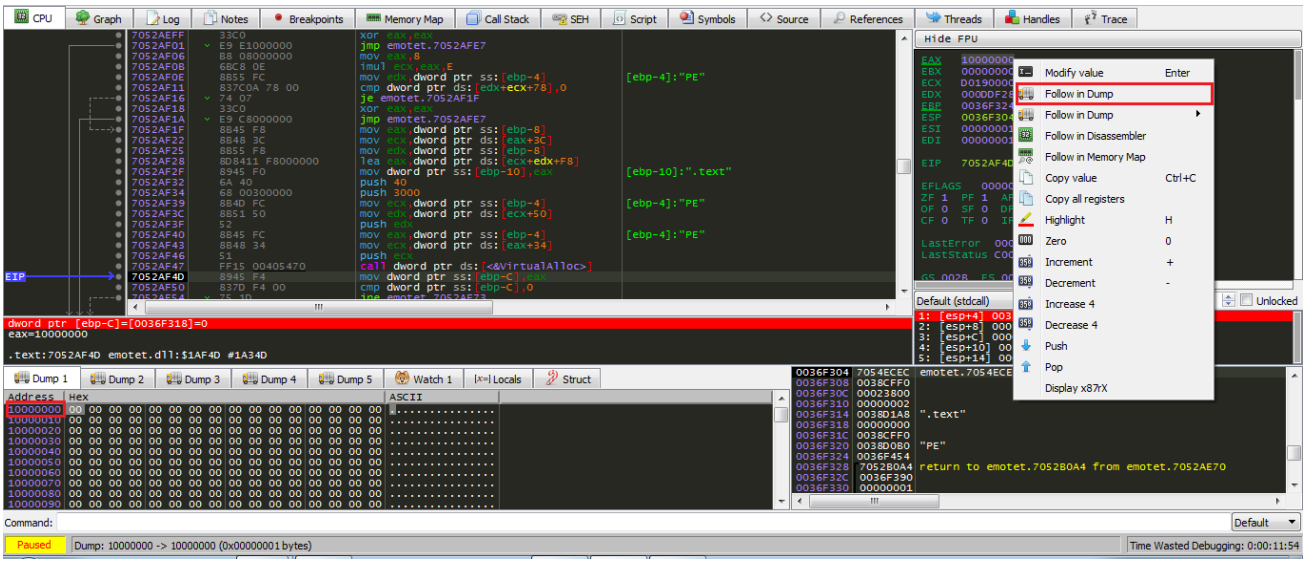

Place a breakpoint on VirtualAlloc and hit run.



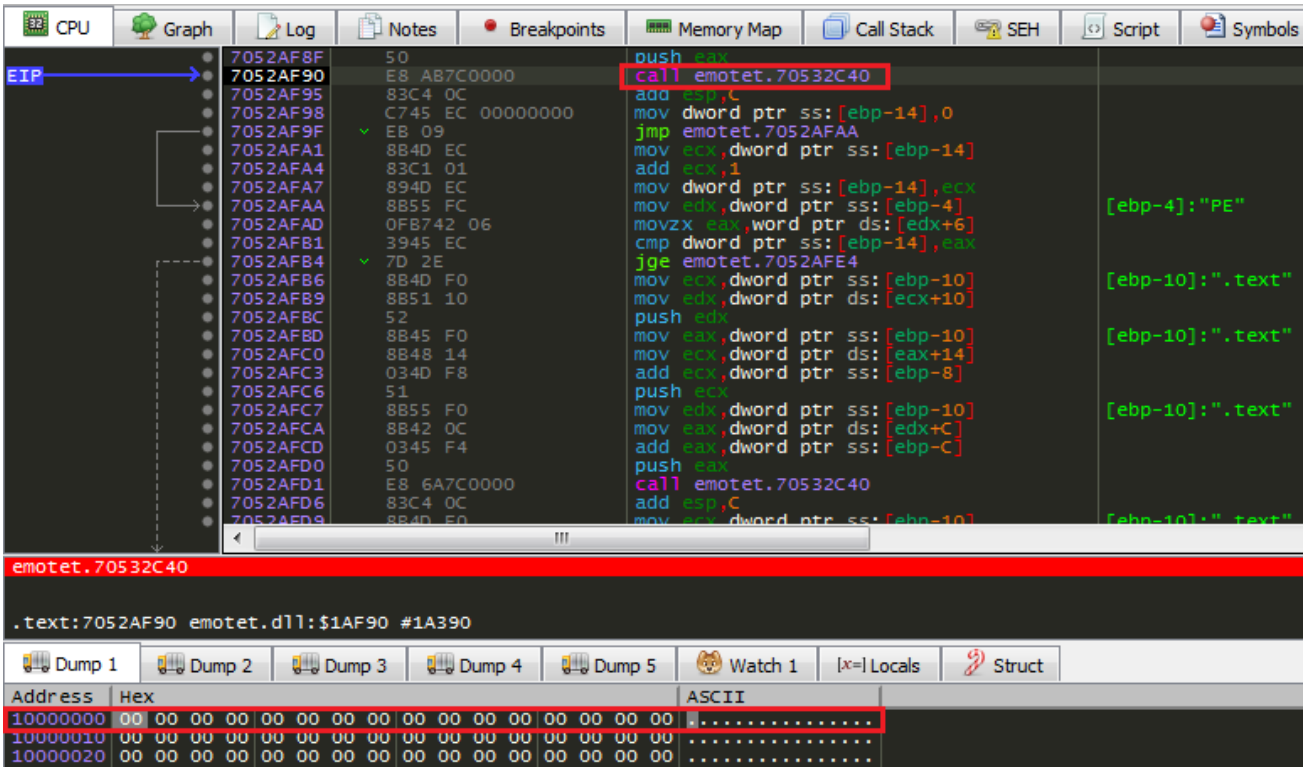

`Xdbg` will keep running untill it hit the breakpoint, after click two times on `Execute till run`.



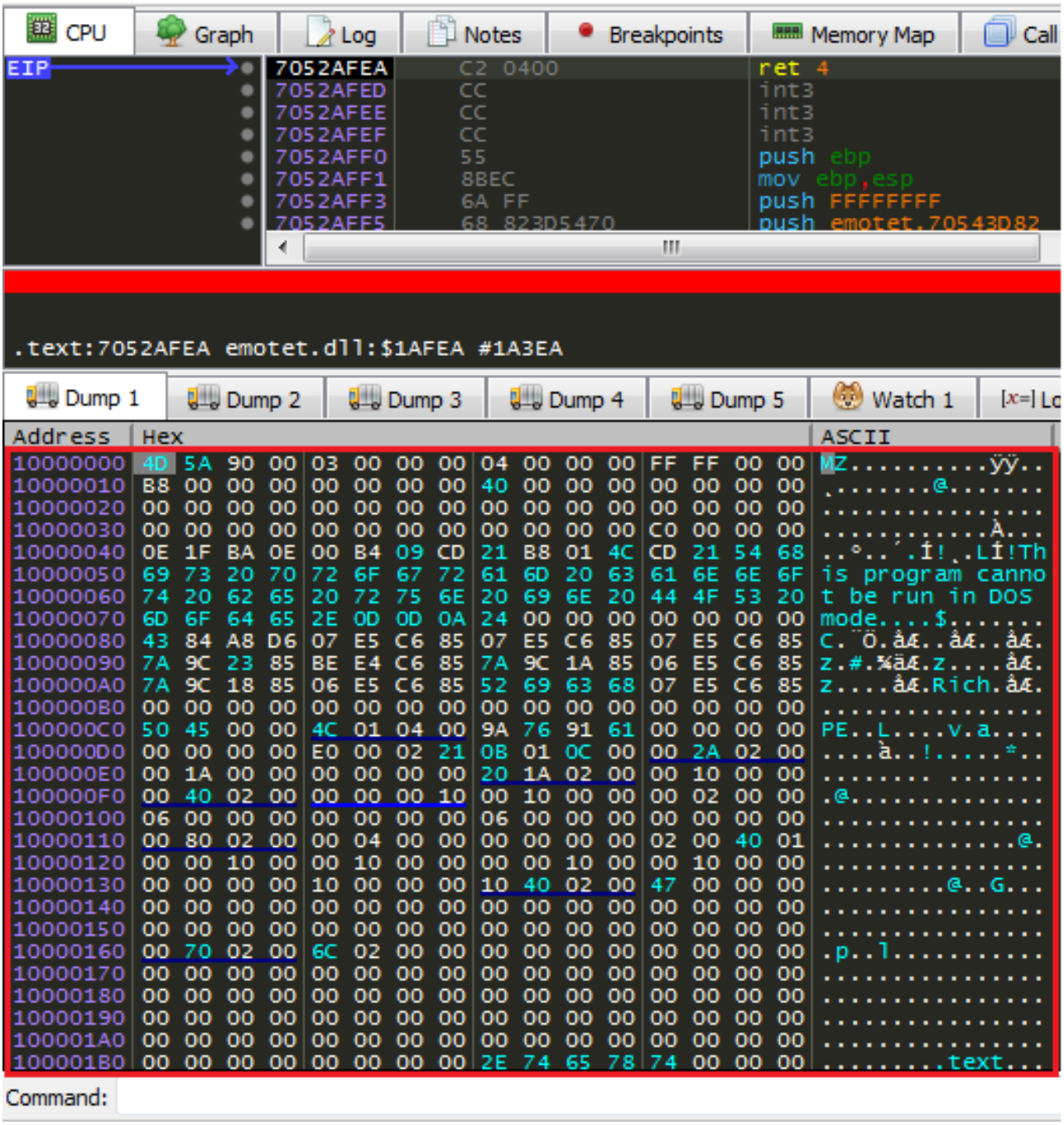

Check the `EAX` register it contain the return adress address of the allocated memory by VirtualAlloc, right click on that value and click on `Follow in Dump` .



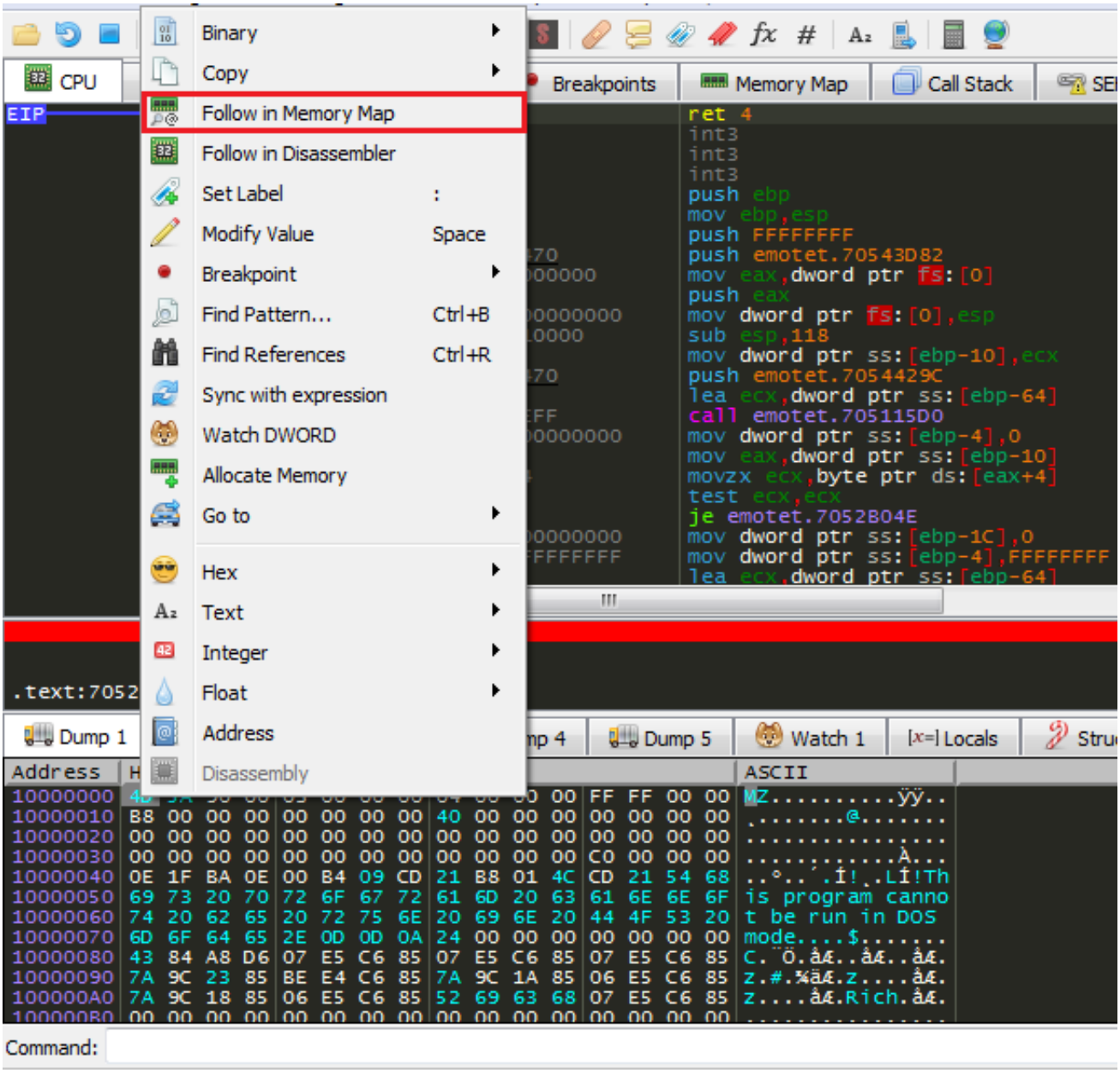

As we said earlier that the function after VirtualAlloc is responsible for unpacking, step over it and keep your eyes on the dump window at the bottom.



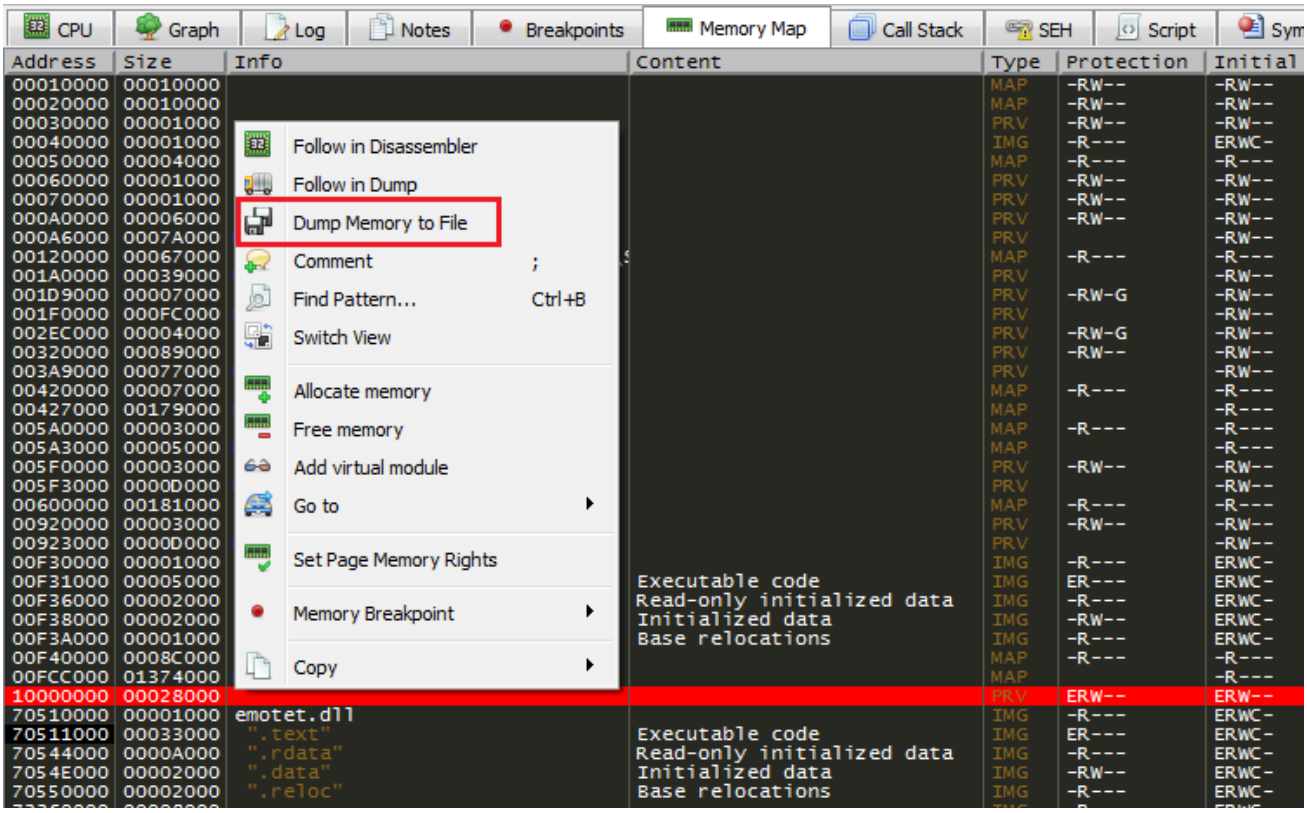

After executing `sub_10022C40` function we can finally see our unpacked malware, dump it and save it somewhere in your machine.



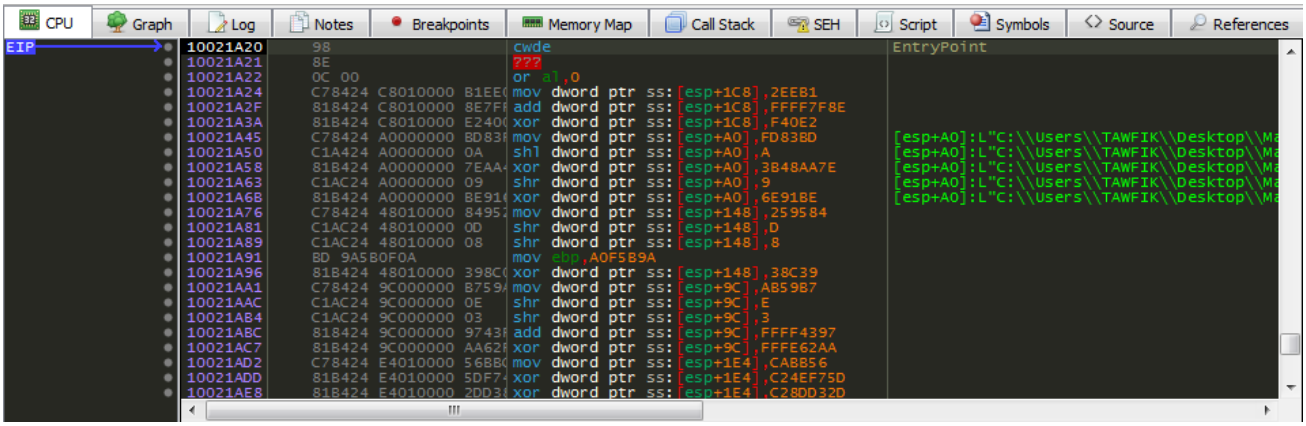

Right click on dump windows and `Follow in memory map`.




Another right click on the address of the unpacked binary then `Dump memory to file`.



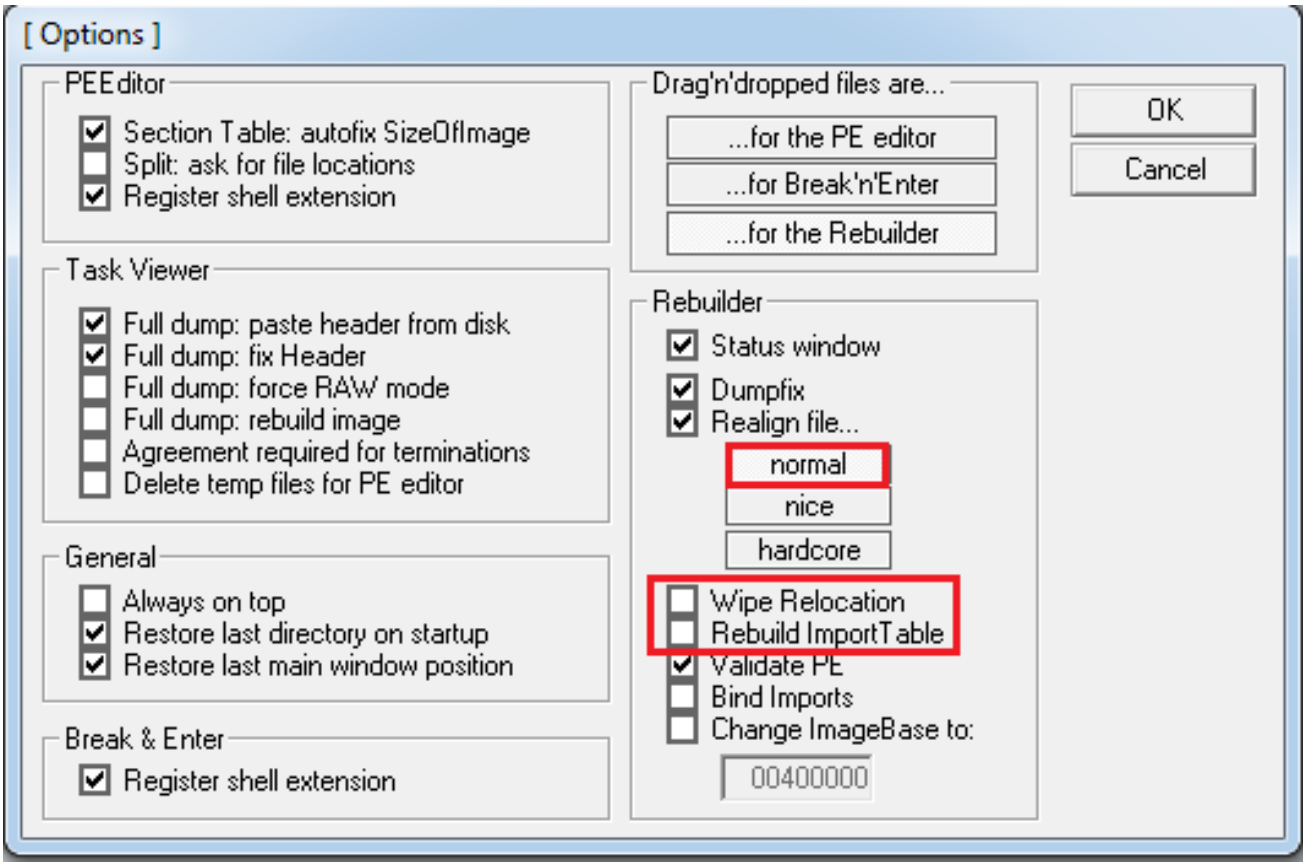

Now that we have our sample unpacked and ready for analysis let's open it in `X32dbg` .



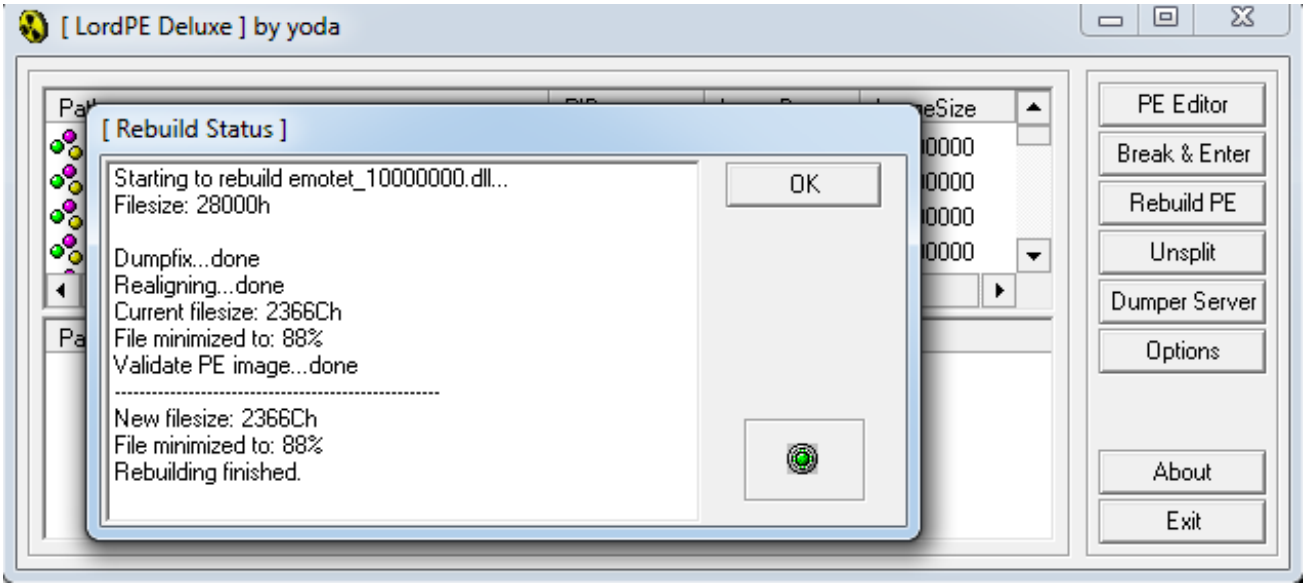

It seems that our unpacked binary is missed and it should be fixed.

## Fixing

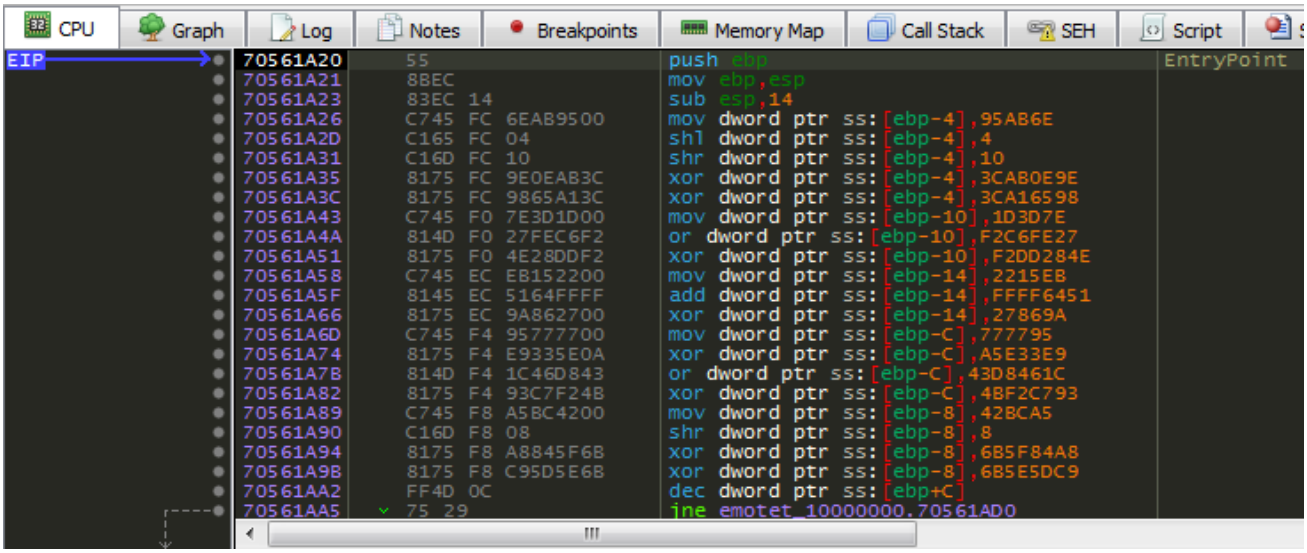
To fix the unpacked binary there are several methods to do that, we will use `LordPE` to automate the fixing, so all we should do is to open LordPe and click on options, then uncheck `Wipe Relocation` and `Rebuild ImportTable` options, finally click on `normal` then `OK` .




Drag your unpacked sample to `LordPe` and it will be fixed automatically.




Finally open your fixed binary in `X32dbg` and notice that it's more readble right now.




## Reference


1. New Emotet 11/2021 - Reverse Engineering VBA Obfuscation + Unpacking
2. How to Unpack Malware with x64dbg