- REVERSE ENGINEERING -

Zbot Malware Unpacking Process

MD5	8a0c95be8a40ae5419f7d97bb3e91b2b
SHA-1	3fb703474bc750c5e99da9ad5426128a8936a118
File Type	Win32 EXE
Packer	ASProtect v1.23 RC1

While start, typically we perform basic static analysis. For example strings, imported functions, sections so on.

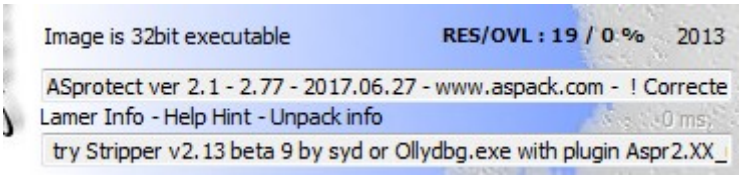
Analysing of strings don't provide human readable anything us. Lets analysis PE sections. I wonder, can we find sections as .text, .data so on? When we analysed sections with CFF Explorer, have some unnamed sections and .data and .rsc sections but we have not found .text section which contains program's codes.

	00004000	00036000	00003000	00022C00	00000000	00000000	0000
	0000B000	0003A000	00002E00	00025C00	00000000	00000000	0000
	0000B000	00045000	00002E00	00028A00	00000000	00000000	0000
	0000B000	00050000	00002E00	0002B800	00000000	00000000	0000
.rsrc	00015000	0005B000	00013000	0002E600	00000000	00000000	0000
	00019000	00070000	00000000	00041600	00000000	00000000	0000
.data	0005B000	00089000	00023E00	00041600	00000000	00000000	0000
.adata	00001000	000E4000	00000000	00065400	00000000	00000000	0000

Also we want to detect imported functions but what the hell? Only one function imported from used every library. This technique is commonly used by packers.

Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
000421D1	N/A	00042108	0004210C	00042110	00042114	00042118
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
kernel32.dll	3	00000000	00000000	00000000	00089A50	00089A3C
user32.dll	1	00000000	00000000	00000000	00089B94	00089C1B
user32.dll	1	00000000	00000000	00000000	00089B9F	00089C23
advapi32.dll	1	00000000	00000000	00000000	00089BAA	00089C2B
oleaut32.dll	1	00000000	00000000	00000000	00089BB7	00089C33
advapi32.dll	1	00000000	00000000	00000000	00089BC4	00089C3B
version.dll	1	00000000	00000000	00000000	00089BD1	00089C43
gdi32.dll	1	00000000	00000000	00000000	00089BDD	00089C4B
oleaut32.dll	1	00000000	00000000	00000000	00089BE7	00089C53
comctl32.dll	1	00000000	00000000	00000000	00089BF4	00089C5B

Because this status is a little suspected, we have used "exeinfope" to detect whether use any packer.

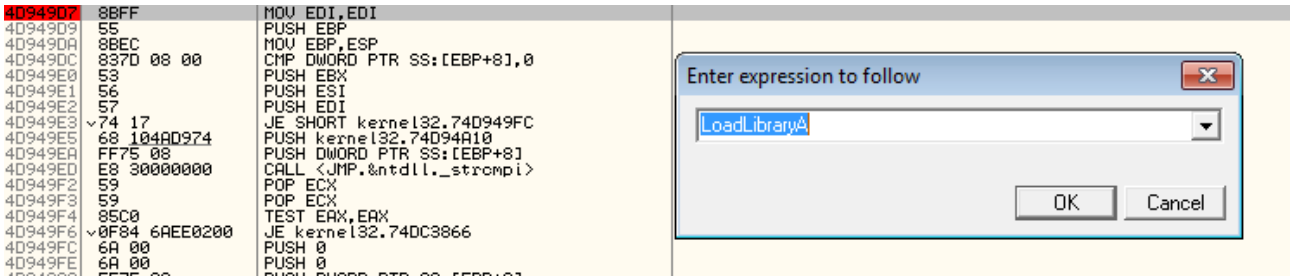


Alright! We have detected which used packer.

Aspack packer has been focusing on security. Therefore Aspack is difficult to unpacking. Also automated unpacking tools may be useless.

Lets explain how to unpacking?
Before "unpacking stub" transfers execution flow to OEP, it loads libraries and functions using two API functions : LoadLibrary and GetProcAddress. Each packed executables includes these API functions certainly. If we find last imported library and function before the execution transfers to "OEP", we are so near malware's main process.

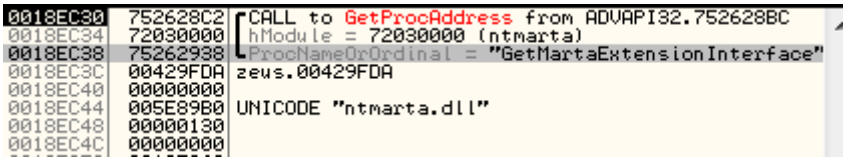
To do this, first in OllyDbg we set breakpoint at both LoadLibraryA and LoadLibraryW functions pressing “Ctrl + G” key combination.



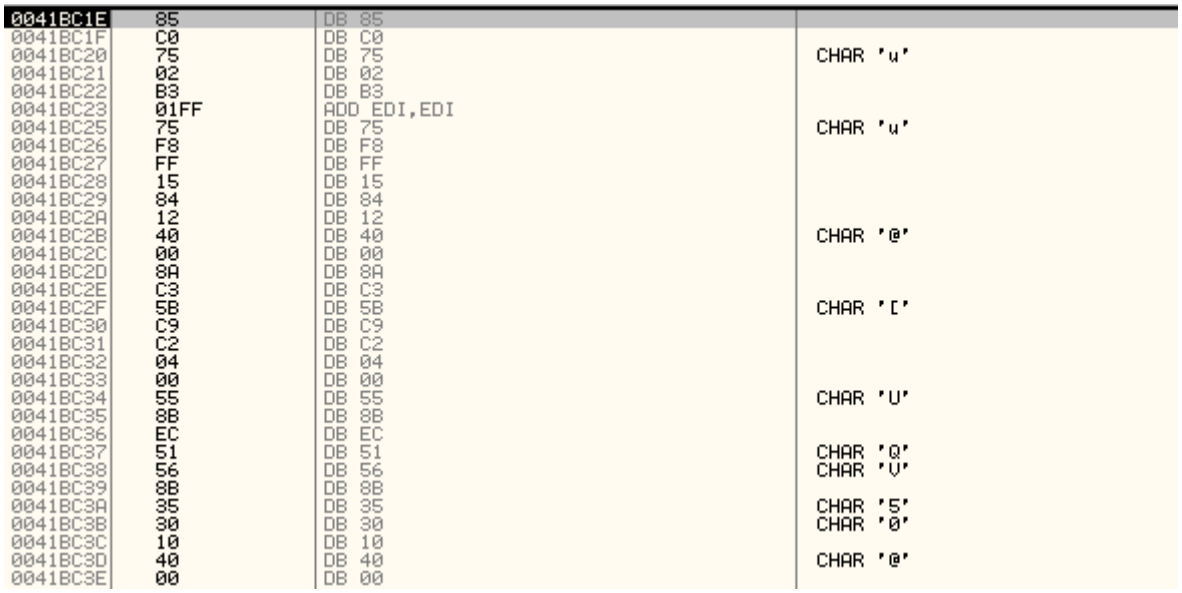
Perform same process for LoadLibraryw and run pressing F9 key until program running. Here so important. That's so important because after each library loaded program will be pause. You need to note last loaded library before program state is running.
Restart the program then you repeat above process and after last library have loaded, in additional set a breakpoint on GetProcAddress and progress until find last imported function. Keep in mind, note last imported function.

In my analysis, loaded last library is "ntmarta.dll".

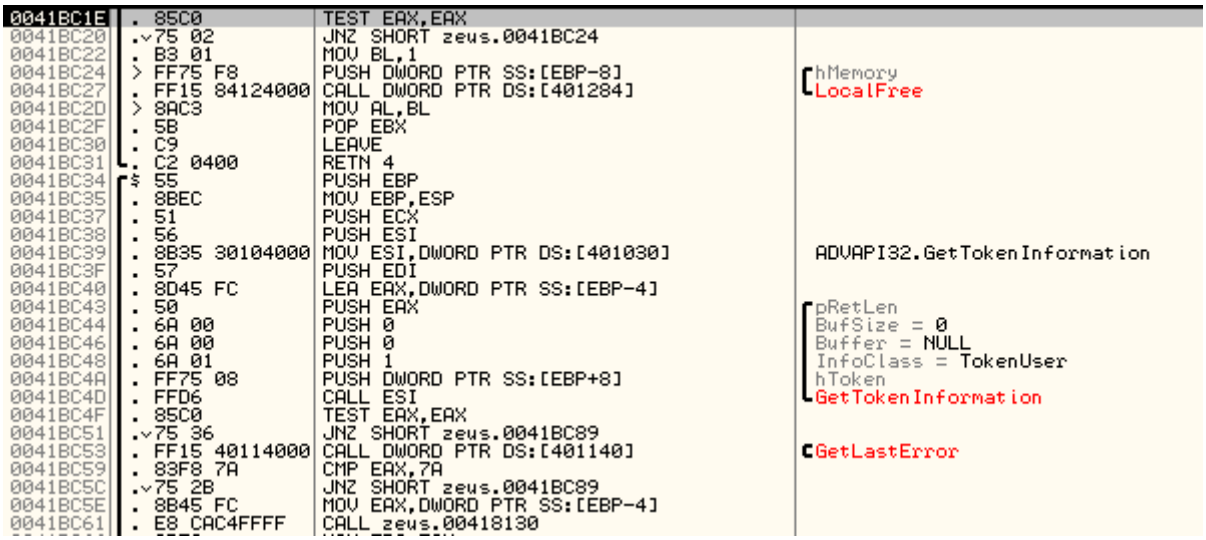
In my analysis, loaded last function is "GetMartaExtensionInterface".



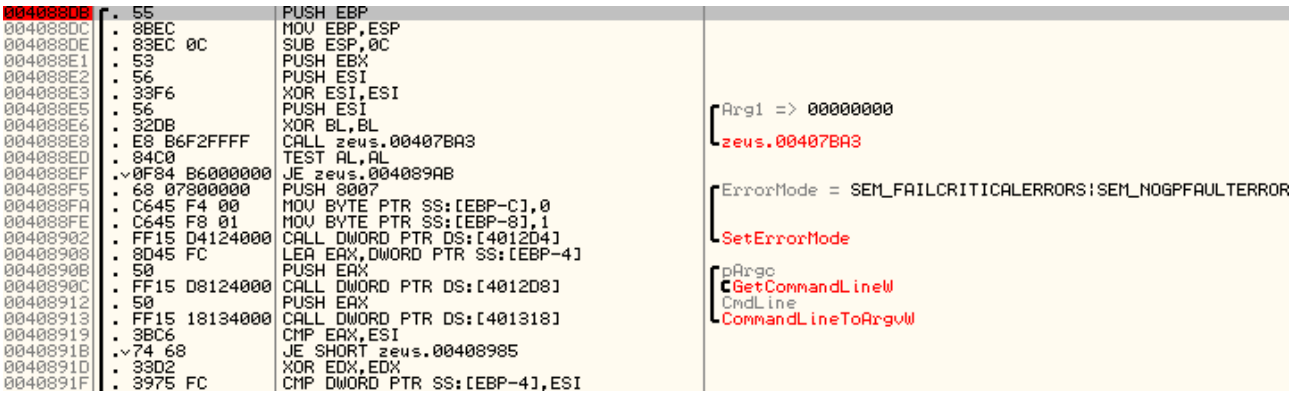
Then you can progress step over(F8) on program's code until find possible program start point.



We reached this point and on cursor point press rigth click and Analysis -> Analyse Code.

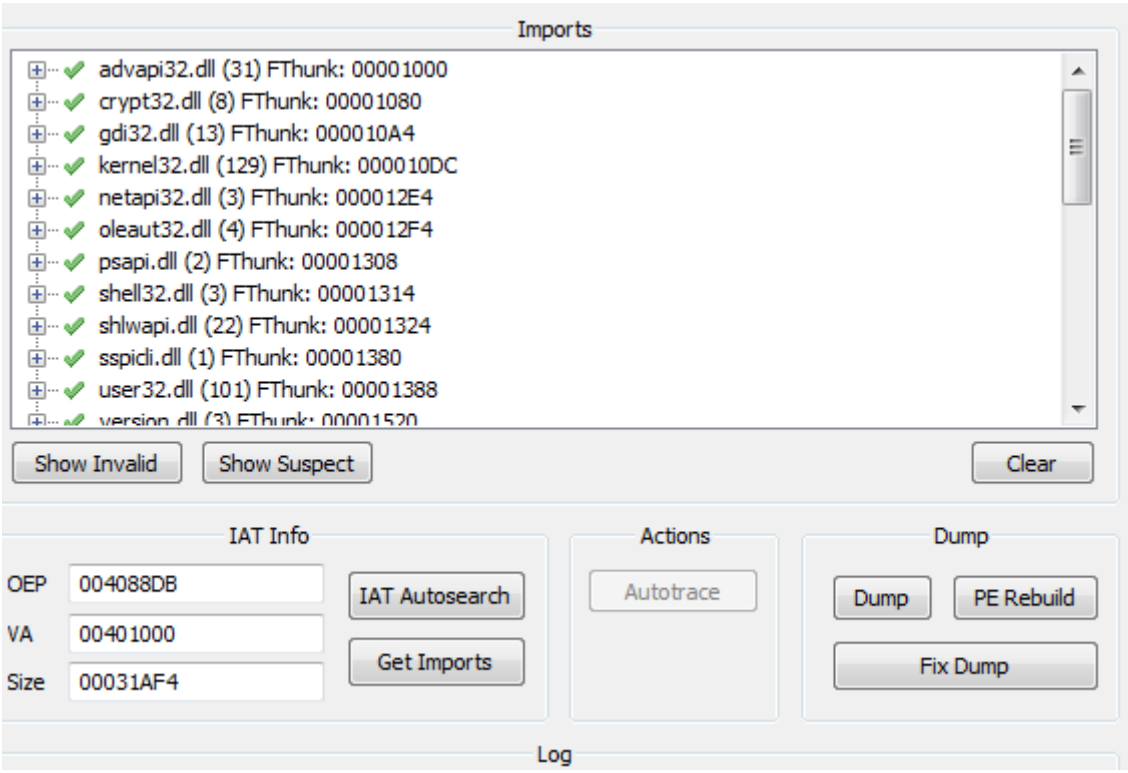


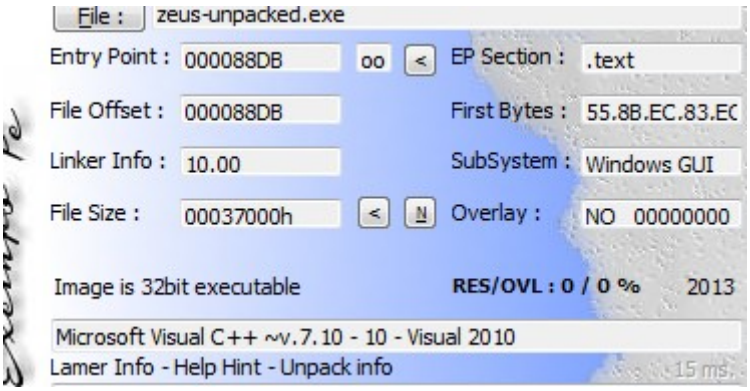
Because 41BC1E address do not present program's start point we progress with step by step.

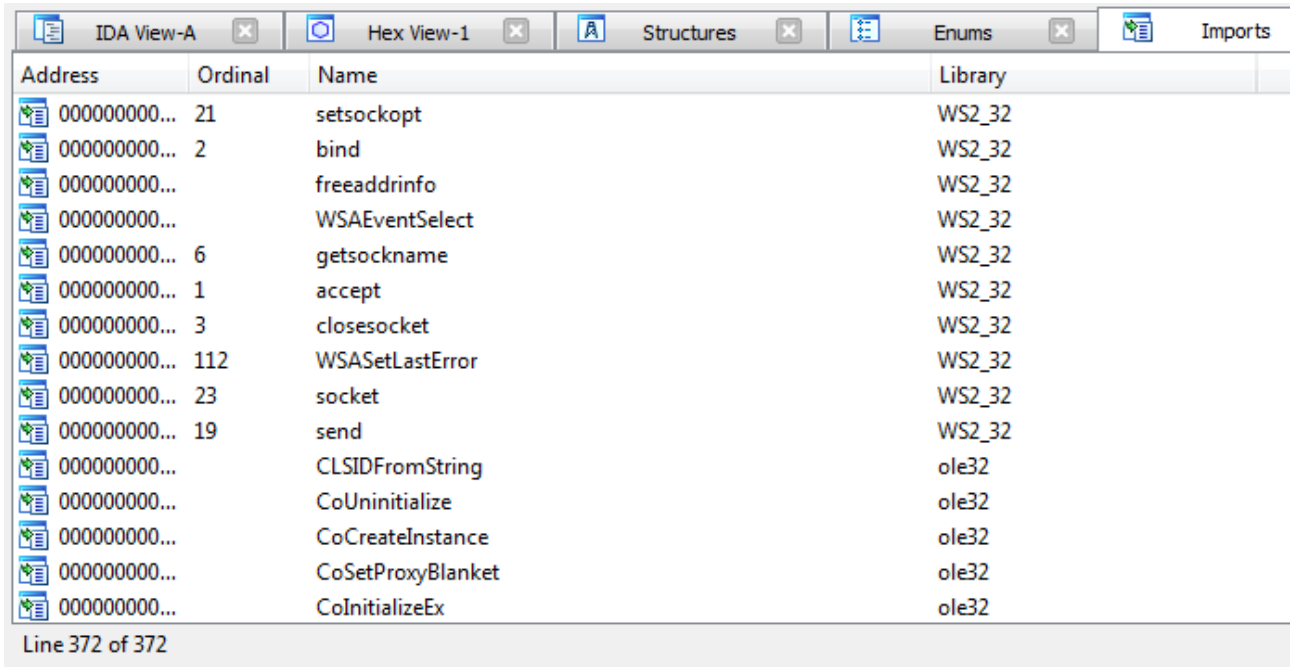


At last, we found possibily start point because executables commonly starts with GetVersionEx or GetCommandLine functions. Now we have to repair import table with Scylla or ImpRec.

We have found OEP as 4088DB, repair import table and reconstruct with Scylla.


Imports
advapi32.dll (31) FThunk: 00001000
crypt32.dll (8) FThunk: 00001080
gdi32.dll (13) FThunk: 000010A4
kernel32.dll (129) FThunk: 000010DC
netapi32.dll (3) FThunk: 000012E4
oleaut32.dll (4) FThunk: 000012F4
psapi.dll (2) FThunk: 00001308
shell32.dll (3) FThunk: 00001314
shlwapi.dll (22) FThunk: 00001324
sspicli.dll (1) FThunk: 00001380
user32.dll (101) FThunk: 00001388
Show Invalid
Show Suspect
Clear
IAT Info
OEP 004088DB
VA 00401000
Size 00031AF4
IAT Autosearch
Get Imports
Actions
Autotrace
Dump
Dump
PE Rebuild
Fix Dump
Log


File : zeus-unpacked.exe
Entry Point : 000088DB
EP Section : .text
File Offset : 000088DB
First Bytes : 55.8B.EC.83.EC
Linker Info : 10.00
SubSystem : Windows GUI
File Size : 00037000h
Overlay : NO 00000000
Image is 32bit executable
RES/OVL : 0 / 0 %
2013
Microsoft Visual C++ ~v.7.10 - 10 - Visual 2010
Lamer Info - Help Hint - Unpack info


IDA View-A
Hex View-1
Structures
Enums
Imports
Address Ordinal Name Library
000000000... 21 setsockopt WS2_32
000000000... 2 bind WS2_32
000000000... freeaddrinfo WS2_32
000000000... WSAEventSelect WS2_32
000000000... 6 getsockname WS2_32
000000000... 1 accept WS2_32
000000000... 3 closesocket WS2_32
000000000... 112 WSASetLastError WS2_32
000000000... 23 socket WS2_32
000000000... 19 send WS2_32
000000000... CLSIDFromString ole32
000000000... CoUninitialize ole32
000000000... CoCreateInstance ole32
000000000... CoSetProxyBlanket ole32
000000000... CoInitializeEx ole32
Line 372 of 372

zeus-unpacked.exe

Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	129	0002F2D0	00000000	00000000	00030152	000010DC
USER32.dll	101	0002F57C	00000000	00000000	00030816	00001388
ADVAPI32.dll	31	0002F1F4	00000000	00000000	00030ADC	00001000
SHLWAPI.dll	22	0002F518	00000000	00000000	00030C68	00001324
SHELL32.dll	3	0002F508	00000000	00000000	00030CAE	00001314
Secur32.dll	1	0002F574	00000000	00000000	00030CCC	00001380
PSAPI.DLL	2	0002F4FC	00000000	00000000	00030D0C	00001308
ole32.dll	7	0002F7E0	00000000	00000000	00030D94	000015EC
GDI32.dll	13	0002F298	00000000	00000000	00030E6E	000010A4
WS2_32.dll	24	0002F77C	00000000	00000000	00030ED4	00001588
CRYPT32.dll	8	0002F274	00000000	00000000	00030FB0	00001080
WININET.dll	21	0002F724	00000000	00000000	00031192	00001530
OLEAUT32.dll	4	0002F4E8	00000000	00000000	0003119E	000012F4
NETAPI32.dll	3	0002F4D8	00000000	00000000	000311E0	000012E4