

codejuggle

juggling code and scratching my head like a dj vinyls

≡ Menu



Uncategorized

Unpack packed DOS binaries with DOSBox debugger

2015-05-04 0 Comments

Set up the DOSBox debugger

First off, you will need a DOSBox version that is compiled with the built-in debugger. If you are using Windows and don't want to compile it yourself, you can grab the latest version of the DOSBox debugger from <http://www.vogons.org/viewtopic.php?t=7323>. The unpacking target of my choice is Omniscient, one of the most impressive 4kb DOS intros that exist. It uses a custom packer to compress the executable. I wanted to have the unpacked version of this intro and compare how the common DOS executable packer compress this file. This tutorial works for both .com and .exe files.

Start DOSBox but do not run your targeted executable yet. Enter DEBUG followed by the name of your executable that you want to unpack and press Return. It seems like the DOSBox Debugger has a bug that causes the window to not refresh itself after trapping into the program. Click into the Debugger window and press a key like Space to refresh the window. Your screen looks now like this:

DOSBox Debugger

---(Register Overview)---

EAX=00000000	ESI=00000100	DS=01FE	ES=01FE	FS=0000	GS=0000	SS=01FE	Real
EBX=00000000	EDI=0000FFFE	CS=01FE	EIP=00000100	C0 Z0 S0 00 A0 P0 D0 I1 T0			
ECX=000000FF	EBP=0000091C				IOPL3 CPL0		
EDX=000001FE	ESP=0000FFFE				173198842		

---(Data Overview Scroll: page up/down)---

0000:0000	60 10 00 F0 08 00 70 00 08 00 70 00 08 00 70 00p...p...p..
0000:0010	08 00 70 00 60 10 00 F0 60 10 00 F0 60 10 00 F0	..p.....
0000:0020	A5 FE 00 F0 87 E9 00 F0 55 FF 00 F0 60 10 00 F0U.....
0000:0030	60 10 00 F0 60 10 00 F0 80 10 00 F0 60 10 00 F0e.....
0000:0040	20 13 00 F0 20 11 00 F0 40 11 00 F0 60 11 00 F0e.....
0000:0050	C0 11 00 F0 E0 11 00 F0 00 12 00 F0 40 12 00 F0e.....
0000:0060	E0 12 00 F0 E0 12 00 F0 60 12 00 F0 60 10 00 F0
0000:0070	80 12 00 F0 A4 F0 00 F0 60 10 00 F0 00 05 00 C0

---(Code Overview Scroll: up/down)---

01FE:0100	33F6	xor	si,si
01FE:0102	BF0020	mov	di,2000
01FE:0105	B510	mov	ch,10
01FE:0107	F3A5	repe	movsw
01FE:0109	8CC8	mov	ax,cs
01FE:010B	050002	add	ax,0200
01FE:010E	50	push	ax
01FE:010F	681301	push	0113
01FE:0112	CB	retf	
01FE:0113	0E	push	cs

---(Variable Overview)---

---(OutPut/Input Scroll: home/end)---

14536: UGA:h total 100 end 80 blank (80/98) retrace (85/97)

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: SNC_OMNI

```

Welcome to DOSBox v0.74

For a short introduction for new users type: INTRO
For supported shell commands type: HELP

To adjust the emulated CPU speed, use ctrl-F11 and ctrl-F12.
To activate the keymapper ctrl-F1.
For more information read the README file in the DOSBox directory.

Press alt-Pause to enter the debugger or start the exe with DEBUG.

HAVE FUN!
The DOSBox Team http://www.dosbox.com

Z:\>SET BLASTER=A220 I7 D1 H5 T6

Z:\>MOUNT C "."
Drive C is mounted as local directory .\

Z:\>C:

C:\>DEBUG SNC_OMNI.COM_
  
```

The normal DOS executable unpacking stub first copies itself plus the packed data to a higher memory location. It then starts to unpack the data to the address space where to execution started. As a result, the memory location will now contain the unpacked code with the Original Entry Point (OEP) of the real program. After the unpacking is done, the stub jumps back to the address of the

first executed instruction and starts the execution of the real program. This behaviour can be used as a quite easy method to unpack the packed executable now.

Debug the target program

Enter the command `BP CS:IP` to set a breakpoint at the current instruction. Execute the current instruction with the F11 key and press F5 to continue the normal execution. After a short time the debugger breaks again at the starting address 0100. Finally the execution stopped at OEP before the first instruction of the unpacked program executes.

Dump the program at OEP

Enter the command `MEMDUMPBIN CS:IP 60000` and press Return. The unpacked program should have been saved into the current directory now. The last parameter of the command is the number of bytes to dump and can be adjusted for larger programs. Your debugger window should look like this now:

```

DOSBox Debugger
--<Register Overview>--
EAX=00000100  ESI=00001100  DS=03FE  ES=01FE  FS=0000  GS=0000  SS=01FE Real
EBX=000012FF  EDI=000013AE  CS=01FE  EIP=00000100  C0 Z1 S0 00 A0 P1 D0 I1 T0
ECX=00007000  EBP=0000D000                                IOPL3 CPL0
EDX=00000100  ESP=0000FFFE                                18880929
--<Data Overview Scroll: page up/down>--
0000:0000    60 10 00 F0 08 00 70 00 08 00 70 00 08 00 70 00  \...-p--p--p-
0000:0010    08 00 70 00 60 10 00 F0 60 10 00 F0 60 10 00 F0  --p-...-...
0000:0020    A5 FE 00 F0 87 E9 00 F0 55 FF 00 F0 60 10 00 F0  \...U...-...
0000:0030    60 10 00 F0 60 10 00 F0 80 10 00 F0 60 10 00 F0  \...e...-...
0000:0040    20 13 00 F0 20 11 00 F0 40 11 00 F0 60 11 00 F0  \...e...-...
0000:0050    C0 11 00 F0 E0 11 00 F0 00 12 00 F0 40 12 00 F0  \...e...-...
0000:0060    E0 12 00 F0 E0 12 00 F0 60 12 00 F0 60 10 00 F0  \...-...-...
0000:0070    80 12 00 F0 A4 F0 00 F0 60 10 00 F0 00 05 00 C0  \...-...-...
--<Code Overview Scroll: up/down>--
01FE:0100  8CC8                mov  ax,cs
01FE:0102  8ED8                mov  ds,ax
01FE:0104  BF4842              mov  di,4248
01FE:0107  050010              add  ax,1000
01FE:010A  AB                  stosw
01FE:010B  050010              add  ax,1000
01FE:010E  AB                  stosw
01FE:010F  050010              add  ax,1000
01FE:0112  B91400              mov  cx,0014
01FE:0115  AB                  stosw
->
--<Variable Overview>--
--<OutPut/Input Scroll: home/end>--
14536: VGA:h total 0.03178 (31.47kHz) blank(0.02542/0.03114) retrace(0.0270
1/0.03082)
14536: VGA:v total 14.26806 (70.09Hz) blank(12.93341/14.04562) retrace(13.0
9230/13.15585)
14536: VGA:Width 640, Height 400, fps 70.086592
14536: VGA:normal width, normal height aspect 1.000000
11847386: EXEC:Parsing command line: DEBUG SNC_OMNI.COM
11847389: EXEC:Execute Z:\DEBUG.COM 0
11847389: FILES:file open command 0 file Z:\DEBUG.COM

```

About Me

```

DEBUG: set breakpoint at 01FE:0100
DEBUG: Memory dump binary success.

```

Rename the file MEMDUMP.BIN to .com or .exe depending on your program type. You can use a hex-editor to remove zero-bytes at the end of the program. Run the renamed executable to see if the unpacking worked. For my unpacking example, the DOS screen is now full of awesomeness:



This unpacking method should work on DOS packers such as 624 and UPX.

[debugger](#) [dos](#) [dosbox](#) [intro](#) [unpacking](#)


You may also

2015-05-04

29 byte DOS

2015-05-04

Converting a D

2015-05-04

Analysis of the 62-

0 Comments

codej

1 Login ▾

♥ Recommend

🔗 Share

Sort by Best ▾



Start the discussion...

I'm Cornel, a Senior Software Engineer with a focus on Golang backend development.

This is where I write about projects and technical stuff that I am interested in.

Be the first to comment.



Recent Posts

Anti-Adblock-Killer Guide to fight Anti-Adblock

How to Install Ubuntu on an Acer C710 Chromebook

17 comments • a year ago •

Bullz Shift — Pay me to view ads or accept the fact that I have no time to see them.

How to setup a development environment with Docker

Facts to know about MongoDB

Anti-Adblock-Killer Guide to fight Anti-Adblock

satish rao — How would you recommend to setup a development environment on a Mac OS X Development Setup Guide system ?
Common mistakes in software licensing systems

1 comment • a year ago •


János Juhász — Thank about your hints. In the age of megaIDEs it is so refreshing to read about that the big softwares made ...
Mac OS X Development Setup Guide

4 comments • a year ago •


Jorge Bucaran — Howdy. If you love Fish, then you should take a look also at Fisherman. It beats Oh My Fish! in every ...

Related Posts

- 29 byte DOS intro called fr29b

- Analysis of the 624 (Six-2-Four) packer
- Converting a DOS intro to JavaScript/HTML5

Tags

32b adblock assembly brew c chromebook css database datatables debugger demoscene dialog docker dos
dosbox facebook gcc golang html5 intro javascript jquery jqueryui licensing mac macos mingw mongodb nasm
packer python qt security stylish tinyexec unpacking visual studio wine

