Emotet Analysis Part 1: Unpacking

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

Player-V

April 2, 2022

Player-V on Apr 22022-04-02T15:26:00+08:00

Updated Apr 102022-04-10T22:41:15+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.

<u>Emotet</u> 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 <u>CFF Explorer</u>, 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	0000000	
NumberOfSymbols	00000118	Dword	0000000	
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		(nFunction	ns)	Dword		Dword	Dword		
KERNEL32.dll	KERNEL32.dll 69					0000000	0000000		
OFTs	OFTs FTs (IAT)				Name				
Dword	Dwo	rd	Word	d	szAn	si			
0003D60C	0003	D60C	05E0		Virtu	alAlloc			
0003D61C	0003	D61C	05E6		VirtualProtect				
0003D62E	0003D62E 0003D62E)	GetP	rocAddress			
0003D640 0003D640			03D7		Load				
0003D650	D650	0461		QueryPerformanceCounter					

Two intersting API functions are used:

- 1. VirtualAlloc
- 2. VirtualProtect

The <u>VirtualAlloc</u> function allocate memory while the <u>VirtualProtect</u> 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 <u>Die(Detect-It-Easy)</u>.

Type PE32	Offse	et 9	Size 0003f200		Count	Size 00000a19	Reload
Total 7.34408	Status	packed(91%)				Save	Save diagram
Entropy By Regions	tes						
Offset	Size	Entropy	Status		Ν	lame	
0000000	00000400	2.75679	not packed	PE Header			
00000400	00033000	7.49175	packed	Section(0)['.text']			
00033400	00009c00	6.02595	not packed	Section(1)['.rdata']			
0003d000	00000e00	3.29659	not packed	Section(2)['.data']			
0003de00	00001400	6.48266	not packed	Section(3)['.reloc']			

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.

THE OWNER	IDA View-A 🛛 🖸	Hex View-1	🗵 🖪	Structures	🗵 🔋	Enums	📔 Imports	🗵 📝	Exports	×
		arg_0	= dword	ptr 8						
		arg_4	= dword	ptr 0Ch						
		arg_C	= dword	ptr 14h						
	.text:10001070		push	ebp						
	.text:10001071		mov	ebp, esp						
	.text:10001073		push	ecx						
	.text:10001074		mov	eax, dword	_1003E024					
	.text:10001079		push							
	.text:1000107A		call	sub_1001A0	E0					
	.text:1000107F		add	esp, 4						
	.text:10001082		mov	ecx, offse	t unk_1003	BECEC				
	.text:10001087		call	sub_1001AF	F0					
	.text:1000108C		cmp	eax, 1						
	text:1000108F		jnz	short loc_	10001098					
	/ .text:10001091		mov							
	text:10001096		jmp	short loc_	100010CE					

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

	IDA View-A	🗵 💿	Hex View-1	🗵 🚺	Str	ructures	×	E	Enums	×	1	Imports	×
Addre	ss	Ordinal	Name						Library				
100 📷	00000010034000		VirtualAlloc						KERNEL	.32			
100	00000010034004		VirtualProte	ct					KERNEL	.32			
100	00000010034008		GetProcAdo	fress					KERNEL	.32			
9 00	0000001003400C		LoadLibrary	A					KERNEL	.32			
90 📷	00000010034010		QueryPerfo	rmanceCou	unter				KERNEL	.32			
9 00	00000010034014		GetCurrent	ProcessId					KERNEL	.32			
100 📷	00000010034018		GetCurrent	ThreadId					KERNEL	.32			

Search for <u>VirtualAlloc</u> and double click on it.



<u>VirtualAlloc</u> function is used two times by the same function <u>sub_1001AFF0</u>, double click on <u>sub_1001AFF0</u> and scroll down we notice that the first function called after <u>VirtualAlloc</u> is <u>sub_10022C40</u>, 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 <u>VirtualAlloc</u> and hit run.

Dump 1	L		Dur	np 2			Dum	ip 3			Dump	94	Į		ump	5	🛞 Watch 1 🛛 🗱 🖉 Struct	
Address	He	х															ASCII	
774A0000	8B	44	24	04	CC	C2	04	00	CC	90	C3	90	CC	C3	90	90	D\$.1A1.A.1A	١
774A0010 774A0020	88	4C	24	04	F6	41	04	90	74 80	05 84	E8	A1 DC	10	01	00	B8 64		 1
774A0030	88	OD	00	00	00	00	BA	10	00	4A	77	89	08	89	50	04	°JwP.	
774A0040	64	A3	00	00	00	00	58	8D	7C	24	0C	FF	DO	8B	8F	cc	d£X. \$.ÿÐ1	
774A0050	02	00	88	64 E0	89	E S	1R	88	00	00	6A EB	01 F8	62	10 10	8E 00	90		
774A0070	64	8B	OD	30	00	00	00	8B	49	10	F6	41	0A	08	75	15	d0I.ÖAu.	
774A0080	FF	74	24	0C	FF	74	24	08	E8	A5	07	09	00	88	00	00	ÿt\$.ÿt\$.è¥	 -
774A0090	00	00	C2	10	00	53	88	SC.	24	08	F6	43	04	06	88	03	··A··S·\\$.0C····	
Command: b	рV	lirt	ual	A11	.oc													
Deveed		ma I													1-1		a verb	

Paused INT3 breakpoint "entry breakpoint" at <emotet.EntryPoint> (7053134F)!

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

CPU	n Graph	📝 Log	📄 Notes	Breakpoints	Memory Map	o 🗐 Call Stack	SEH SEH	Script	🖭 Symbols	<> Source	References
EIP	→ •	76521856	8BFF		mov edi,edi			Virtua	lAlloc		
	۰	76521858			push ebp						_
	•	76521859	8BEC								
	۰	7652185B			pop ebp						
	•	7652185C	EB 05		jmp <jmp.&vi< th=""><th>rtualAlloc></th><th></th><th></th><th></th><th></th><th></th></jmp.&vi<>	rtualAlloc>					
	•	7652185E									
	•	7652185F	90		nop						
		76521860	90		nop						
		76521861	90		nop						
		76521862	90		nop			7117 414	inturlallar		
		76521003	× FF25	08035276	Jub aword br		ATTOCAL	JMF.QV	ii cualArioc		
		76521005	90		nop						
		7652186R	90		nop						
	ě	76521860	90		nop						
		7652186D	90		nop						
		7652186E	8BFF		mov edi.edi			Virtua	lFree		
		76521870			push ebp						
	•	76521871			mov ebp_esp						
	•	76521873			pop ebp						
	•	76521874			jmp <jmp.&vi< th=""><th>rtualFree></th><th></th><th></th><th></th><th></th><th></th></jmp.&vi<>	rtualFree>					
	•	76521876									
	•	76521877									
	•	76521878									
	•	76521879									~
	•	4	90		non						
	\downarrow										P

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

🕮 CPU 🧁 Graph 📝 Log 📋 Not	es 🔹 Breakpoints 🛲 Memory Map	🗍 Call Stack 🛛 📆 SEH 🗾 Script	: 🎱 Symbols 🗘 Source 🚽	References	Threads 🛛 🔒 Handles	Trace	
7052AEFF 33C 7052AF01 × E9	E1000000 imp emotet.7052	2AFE7		∧ Hid	e FPU		
7052AF06 B8 7052AF08 680 7052AF08 680 7052AF01 885 7052AF11 837	08000000 mov eax,8 8 0E imul ecx,eax,E 5 FC mov eax,dword pt COA 78 00 cmp dword ptro	ptr ss:[ebp-4] ds:[edx+ecx+78],0		EAX EBX ECX EDX	10000000 00000000 I Mo D0190000 00000E28 III Fo	dify value Enter	
7052AF18 33C 7052AF1A 532 7052AF1A 884 7052AF15 884	O/ Je emotet.705/x C8000000 jmp emotet.7057 S5 F8 mov eax,dword j mov eax,dword j mov ecx,dword j	AFIF 2AFE7 ptr ss:[ebp-8] ptr ds:[eax+3C]		EBP ESP ESI EDI	0036F324 0036F304 4 Fo 00000001 8 Fo	low in Dump	
7052AF25 885 7052AF28 808 7052AF2F 894 7052AF32 6A 7052AF32 6A	5 F8 mov edx, dword p 1/411 F8000000 1ea ex, dword p 5 F0 mov dword ptr s 40 push 40 push 40 00300000 push 40 push 40	ptr ds:[ecx+edx+F8] ss:[ebp-10],eax [ebp-:		EIP	7052AF4D 👼 Fol	low in Memory Map py value Ctrl+C	
7052AF39 884 7052AF30 885 7052AF3C 885 7052AF3F 52	D FC mov ecx, dword p 1 50 mov edx, dword p push edx	ptr ss: [ebp-4] [ebp-4] ptr ds: [ecx+50]		OF C CF C	1 PF 1 AF (1) Co 5 F 0 DF 0 TF 0 IF (2) Hig	py all registers hlight H	
7052AF40 8B4 7052AF43 8B4 7052AF43 8B4 7052AF46 51 052AF47 FF1	IS FC mov eax, dword p 18 34 mov ecx, dword p push ecx IS 00405470 call dword ptr	ptr ss:[ebp-4] ptr ds:[eax+34] ds:[<&virtualAlloc>]		Last	tError 000 000 Zei tStatus COC	ro 0 rrement +	
EIP 7052AF4D 894 7052AF50 837 7052AF50 7052AF54 75	5 F4 mov dword ptr s D F4 00 cmp dword ptr s 10 ine emotet 705	SS: [ebp-C], cax SS: [ebp-C],0 24573		The fault of the f	DO2B ES OC BB De	crement -	Unlocked
dword ptr [ebp-C]=[0036F318]=0 eax=10000000	m			1: [2: [3: [esp+4] 003 esp+8] 000 85 De esp+C] 000	crease 4	
.text:7052AF4D emotet.dll:\$1AF4D #1	LA34D			4: [5: [esp+10] 00 🔮 Pu esp+14] 00	sh	
Ump 1 Ump 2 Ump 3	Ump 4 Dump 5 SWatch 1	Ix=I Locals 22 Struct	0036F30 0036F30 0036F30 0036F33	04 7054ECEC emot 08 0038CFF0 00023800 000202000	et.7054ECE T Po Dis	p play x87rX	^
10000000 00 00 00 00 00 00 00 00 00 00 10000010 00 00 00 00 00 00 00 00 00 10000020 00 00 00 00 00 00 00 00 00 10000020 00 00 00 00 00 00 00 00 00	0 00 00 00 00 00 0. 0 0 00 00 00 00 0. . <t< td=""><td></td><td>0036F3: 0036F3: 0036F3:</td><td>14 0038D1A8 ".te 18 00000000 10 0038CFF0</td><td>ext"</td><td></td><td></td></t<>		0036F3: 0036F3: 0036F3:	14 0038D1A8 ".te 18 00000000 10 0038CFF0	ext"		
10000040 00 <	00 00<		0036F3 0036F3 0036F3 0036F3 0036F3	20 00380080 "PE" 24 0036F454 28 705280A4 retu 20 0036F390 30 00000001	urn to emotet.7052	BOA4 from emotet.7052AE	70
Command:							Default 🔻
Paused Dump: 10000000 -> 10000000 (0x0	00000001 bytes)					Time Wasted Debu	ugging: 0:00:11:54

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

CPU	🍨 Graph	Log	Notes	Breakpoints	Memory Map	Call Stack	SEH (Script 	🖭 Symbols
		7052AF8F	50		push eax				
FIN	<u> </u>	7052AF90	E8 AB/C0		call emotet./0	JS 32C 40			
		7052AF95	63C4 UC	0000000	mov dword ntr	ss [abn=14] 0			
		7052AF96	V FR 09		imp emotet.705	33.[COP-14];0			
		7052AFA1	8B4D FC		mov ecx.dword	ntr ss:[ebn-1	41		
	•	7052AFA4	83C1 01		add ecx 1	her and from -			
	•	7052AFA7	894D EC		mov dword ptr	ss:[ebp-14],e			
		7052AFAA	8855 FC		mov edx,dword	ptr ss: [ebp-4		[ebp-4]]:"PE"
	٠	7052AFAD	0FB742 (06	movzx eax word	d ptr ds:[edx+			
	٠	7052AFB1	3945 EC		cmp dword ptr	ss:[ebp-14],e			
		7052AFB4	7D 2E		jge emotet.705	SZAFE4			
	•	7052AFB6	8B4D F0		mov ecx,dword	ptr ss: ebp-1	0]	[ebp-10	0]:".text"
	•	7052AFB9	8B51 10		mov edx,dword	ptr ds:[ecx+1	0]		
		7052AFBC	52		pusn eax	nen and take a		Cale of	
		7052AFBD	8845 FU		mov eax, dword	ptr ss: eop-1		[eop-10	J:text
		7052AFC0	0240 50		add ecy dword	ptr us: [edx+1	* J		
		70524FC5	51		nush ecv	hri 22. Tenh-0			
		7052AFC0	8855 FO		mov edv dword	ntr ss.[ehn-1	01	Eebn-1(hl." text"
		7052AFCA	8842 OC		mov eax dword	ntr ds: [edx+C		Leob T	J cexe
	i i	7052AFCD	0345 F4		add eax dword	ptr ss: ebp-C			
	i i	7052AFD0	50		push eax	her and from a			
	•	7052AFD1	E8 6A7C	0000	call emotet.70	0532C40			
	•	7052AFD6	83C4 OC		add esp,C				
	•	70524ED9	RR4D FO		mov ecy dword	ntr ss [ehn-1	0]	Eebn-1(hl." text"
	÷	•							
emotet.70	0532C40								
.text:705	2AF90 emo	tet.dll:\$	LAF90 #1A390						
💷 Dump 1	L 🛄 Dun	np 2 🛛 💷 C	ump 3 🛛 💷 D	ump 4 🛛 💷 Dur	mp 5 🛛 🧶 Watch 1	[x=] Locals	Struct		
Address	Нех				ASCII				
10000000	00 00 00	00 00 00	00 00 00 00	00 00 00 00 0	00 00				
10000010	00 00 00	00 00 00	00 00 00 00	00 00 00 00 0	00 00				
10000020	00 00 00	00 00 00	00 00 00 00	00 00 00 00 00	00 00				

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

🕮 CPU	👰 Gra	aph		Log		<u>n</u>	lote	s	•	Bre	akpo	oints			Memory Map 🛛 🗍 Call
EIP		• • • • • •	7052 7052 7052 7052 7052 7052 7052 7052	AFEA AFED AFEF AFFO AFF1 AFF3 AFF5		C C C S 8 6 6	2 0 C C BEC A F 8 8	400 F 23D	547	0			r i i p m p	et nt3 nt3 nt3 ush ov ush	4 ebp ebp,esp FFFFFFF emotet.70543D82
.text:70524	FEA	emot	tet.o	111:	\$1AF	EA	#1/	3EA	L.						
🛄 Dump 1		Dum	p 2		Dum	р3)ump	94	, U	D	ump	5	💮 Watch 1 🛛 [x=] Lo
Address He	ex														ASCII
10000010 84 10000010 84 10000020 00 10000030 00 10000050 69 10000050 74 10000050 74 10000050 74 10000050 74 10000050 74 10000050 00 10000050 00 10000050 00 10000100 00 10000100 00 10000100 00 10000120 00	3 00 3 00 0 00 2 1F 7 3 4 20 5 73 4 20 6 73 4 20 6 73 4 20 0 6F 3 84 4 9C 0 00 0 40 5 00 0 0	00 00 00 00 00 62 62 64 18 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00	000 000 000 000 000 000 000 000 000 00	000 000 009 67 75 00 06 67 75 00 06 00 000 000 000 000 000 000 000 0	000 000 000 000 000 000 000 000 000 00	40 000 21 61 20 24 07 7A 52 09 A 00 00 00 00 00 00 00 00 00 00 00 00 0	00000880005900701100000000000000000000000000000	00000000000000000000000000000000000000	000004C 63000 463200 85560 610000000000000000000000000000000000	00000000000000000000000000000000000000	000001 6E 4F005555000210000000000000000000000000000	0000054 6E3006666000200004000000000000000000000000	0000068 670055550000000000000000000000000000000	

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.

🕮 CPU	🍨 🖗 Graph		Log	📄 Notes	•	Breakpoints	Memory Map	Call Stack	🥂 😚 SE	:H 💿 Script	📃 🖭 Sym
Address	Size	Info					Content		Туре	Protection	Initial
00010000	00010000								MAP	-RW	-RW
00020000	00010000								MAP	-RW	-RW
00030000	00001000	Source of							PKV	-KW	-RW
00040000	00001000	100	Follow	in Disassemble	er 👘					-R	ERWC-
00050000	00004000	-								-R	
00060000	00001000	0-0	Follow	/ in Dump						-NW	
00040000	00006000	5	Duran	Mamanu ta Eila					PRV	-RW	-RW
00046000	00074000	lar"	Dump	Memory to File					PRV		-RW
00120000	00067000	0	Comm	opt						-R	-R
001A0000	00039000	4 -2	Comm	ent		· ·			PRV		-RW
001D9000	00007000	6	Find P	attern		Ctrl+B			PRV	-RW-G	-RW
001F0000	000FC000	200	1 11 10 1	accentin		Carro					-RW
002EC000	00004000		Switch	n View						-RW-G	-RW
00320000	00089000									-RW	-RW
003A9000	00077000										-RW
00420000	00007000		Alloca	te memory						-R	-R
00427000	00179000								MAP		-R
005A0000	00003000	-	Free r	nemory					MAP	-R	-R
005A3000	00005000								MAP		-R
005F0000	00003000	6-0	Add v	irtual module					PRV	-RW	-RW
005F3000	000000000		C						PKV		-KW
00600000	00181000		GO tO			· · ·				-R	-R
00920000	00003000								PRV	-KW	-RW
00F30000	00001000		Set Pa	aae Memory Ria	hts					-R	FRWC-
00F31000	00005000	•					Executable code		IMG	ER	ERWC-
00F36000	00002000						Read-only initia	lized data		-R	ERWC-
00F38000	00002000	•	Memo	ry Breakpoint		•	Initialized data	1		-RW	ERWC-
00F3A000	00001000						Base relocations			-R	ERWC-
00F40000	0008C000	PA .	Contra			I				-R	-R
00FCC000	01374000	41	Copy			· ·					-R
10000000	00028000									ERW	ERW
70510000	00001000	emote	et.dl	1						-R	ERWC-
70511000	00033000						Executable code		IMG	ER	ERWC-
70544000	0000A000		data"				Read-only initia	lized data	IMG	-R	ERWC-
7054E000	00002000		ata"				Initialized data		IMG	-RW	ERWC-
70550000	00002000		eroc				Base relocations			-R	ERWC-

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

CPU	n 🖗 Graph	📄 Log	Notes	Breakpoints	Memory Ma	ap 🗐	Call Stack	SEH	Script	🐏 Symbols	<> Source	References
EIP	\rightarrow	10021A20							EntryPo			A
	۲	10021A21										
	•	10021A22										
	•	10021A24		C8010000 B1EE	mov dword p	tr ss:[e	sp+1C8]					
	۰	10021A2F	818424	C8010000 8E7F	add dword p	tr ss:[e	sp+1C8]	,FFFF7F8E				
	۲	10021A3A	81B424	C8010000 E240	xor dword p	tr ss:[e	sp+1C8	,F40E2				
	٠	10021A45	C78424	A0000000 BD83	mov dword p	tr ss:[e	sp+A0],	FD 83BD	[esp+A0	D]:L"C:\\Use	'S\\TAWFIK\	\Desktop\\Ma
	•	10021A50	C1A424	A0000000 0A	sh1 dword p	tr ss: e	sp+A0],	A	[esp+A0	0]:L"C:\\Use	'S\\TAWFIK\	\Desktop\\Ma
	•	10021A58	81B424	A0000000 7EAA	xor dword p	tr ss: e	sp+A0,	3B48AA7E	Lesp+A0	o]:L"C:\\Use	'S\\TAWFIK\	\Desktop\\Ma
	•	10021A63	C1AC24	A0000000 09	shr dword p	tr ss: e	sp+A0,	9	Lesp+A0	oj:L"C:\\Use	'S\\TAWFIK\	\Desktop\\Ma
		10021A68	818424	A0000000 BE91	xor aword p	tr ss: e	sp+A0	6E918E	[esp+A0	o]:L"C:\\use	'S//IAWFIK/	\Desktop\\Ma
		10021A76	C78424	48010000 8495	mov dword p	tr ss: e	sp+148	,259584				
		10021481	CIAC24	48010000 0D	shr dword p	tr ss: e	SP+148					
		10021489	CIAC24	48010000 08	Shir aword p	ur ssile	sb+148]					
		10021491	60 9A51	40010000 2000	won dwond n	569A +n 55. F o	CD+1491					
		10021456	010424	9C00000 8759	mov dword p	tr 55. 0	50+140J	, DC 007				
		10021441	C10424	9000000 6759	shr dword p	tr 55. 0	spraci,	AD3 3D7				
		10021484	C14C24	90000000000	shr dword p	tr ss e	s_{n+9}	2				
		10021480	R1R474	9000000 9743	add dword p	tr ss e	sn+9C1	5 EEEE4397				
	ě	10021AC7	81R424	9000000 4462	xor dword p	tr ss: e	sp+9C1	FFFF62AA				
		10021AD2	C78424	E4010000 5688	mov dword p	tr ss: e	sp+1F4	CARRSS				
	ě	10021ADD	81B424	E4010000 5DE7	xor dword p	tr ss: e	50+1E4	C24EE75D				
	ě	10021AE8	81B424	E4010000 2DD3	xor dword p	tr ss: e	SD+1E4	C28DD32D				*
		•										4

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.

	huild Status 1		eSize 🔺	PE Editor
Sta	arting to rebuild emotet_10000000.dll	ОК	10000	Break & Enter
	esize: 28000n		0000	
Re	imprixdone aligningdone		•	Dumper Serve
Pa File	rrent filesize: 2366Ch e minimized to: 88%			Optione
Val	lidate PE imagedone			Options
Ne	w filesize: 2366Ch			
Re	building finished.	<u> </u>		About

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

🕮 CPU	🍨 Graph	Log	Notes	Breakpoints	Memory Map 🗍 Call S	Stack 🛛 🗠 SEH	💿 Script 🛛 🔮 🤮
EIP	\longrightarrow	70561A20	55		push ebp		EntryPoint
	٠	70561A21	8BEC		mov ebp esp		
	•	70561A23	83EC 14		sub esp,14		
	٠	70561A26		6EAB9500	mov dword ptr ss:[ebp-	4],95AB6E	
	٠	70561A2D	C165 FC	04	shl dword ptr ss: [ebp-	4],4	
	٠	70561A31	C16D FC		shr dword ptr ss: ebp-	4,10	
	٠	70561A35	8175 FC	9E0EAB3C	xor dword ptr ss: [ebp-	4],3CABOE9E	
	٠	70561A3C	8175 FC	9865A13C	xor dword ptr ss: [ebp-	4],3CA16598	
	٠	70561A43		7E3D1D00	mov dword ptr ss:[ebp-	10],1D3D7E	
	٠	70561A4A	814D F0	27FEC6F2	or dword ptr ss: [ebp-1	0],F2C6FE27	
	٠	70561A51	8175 FO	4E28DDF2	xor dword ptr ss: [ebp-	10],F2DD284E	
	٠	70561A58		EB152200	mov dword ptr ss:[ebp-	14],2215EB	
	٠	70561A5F	8145 EC	5164FFFF	add dword ptr ss:[ebp-	14],FFFF6451	
	٠	70561A66	8175 EC	9A862700	xor dword ptr ss:[ebp-	14],27869A	
	٠	70561A6D		95777700	mov dword ptr ss:[ebp-	C],777795	
	٠	70561A74	8175 F4	E9335E0A	<pre>xor dword ptr ss:[ebp-</pre>	C],A5E33E9	
	٠	70561A7B	814D F4	1C46D843	or dword ptr ss: [ebp-C	,43D8461C	
	٠	70561A82	8175 F4	93C7F24B	xor dword ptr ss:[ebp-	C],4BF2C793	
	٠	70561A89	C745 F8	A5BC4200	mov dword ptr ss:[ebp-	8],428CA5	
	٠	70561A90	C16D F8	08	shr dword ptr ss:[ebp-	8],8	
	٠	70561A94	8175 F8	A8845F6B	xor dword ptr ss:[ebp-	8],685F84A8	
	٠	70561A9B	8175 F8	C95D5E6B	xor dword ptr ss:[ebp-	8],685E5DC9	
	٠	70561AA2	FF4D OC		dec dword ptr ss:[ebp+	2]	
	e1	70561AA5	75 29		ine emotet_10000000.70	561AD0	
		•					

Reference

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