# **Unpacking Emotet malware part 02**

muha2xmad.github.io/unpacking/emotet-part-2/

January 7, 2022





#### Muhammad Hasan Ali

Malware Analysis learner

5 minute read

As-salamu Alaykum

# Part 01 summary

Download the sample: <u>Here</u>

found VirtualAlloc call in sub\_417D50 and its address.

we search for abnormal jumps. we found jmp ecx and its address.

#### Introduction

we will debug our sample with x32dbg tool to unpack the Emotet malware

#### Notes to be taken

- What is Packing: A trick which is Used to avoid AV detection and analysis.
- What is a packer: A tool that compresses, encrypts, and/or modifies a malicious file format. <u>1</u>
- Why using packers: To avoid AV detection and analysis to make it harder for researchers to analyze the code
- We need to find the the original entry point (OEP). What is the OEP: is the address of the malware's first instruction (where malicious code begins) before it was packed. <u>2</u>
- How to find the OEP: find the tail jump . the tail jump It's an unconditional jump exists in the tail of stub code , it points to address of unpack file. <u>3</u>

How to the unpack happen?  $\underline{3}$  As we see in the figure (1). OS create stub code with packed file

What is stub code? <u>3</u> Stub code is responsible for unpacking packed sections, when you are running the file ,the address of unpack file exists in the stub code to unpack file. So at the end of the stub code we will see an unconditional jump (tail jump), that is meant after execute the stub code will jump to the address of unpacking file.



Figure(1):

What is stack string? answer

We need to know what is VirtualAlloc ? Here

Says that "Reserves, commits, or changes the state of a region of pages in the virtual address space of the calling process. Memory allocated by this function is automatically initialized to zero."

#### Syntax

```
LPVOID VirtualAlloc(

[in, optional] LPVOID lpAddress,

[in] SIZE_T dwSize,

[in] DWORD flAllocationType,

[in] DWORD flProtect

);
```

The most important parameter of this function is **lpaddress**, **which returns the starting offset of the newly allocated memory**. where we will extract the malware then dump it.

# Start Debugging

Open our sample by x32dbg and hit the entry point

CPU	📄 Log 🛛	Notes		Breakpoints	🛲 Mer	mory Map		Call Stack	<b>~</b> 2	SEH	0	Script	2	Symbols		Source	2
EIP EDX		e	00417	7 <mark>E00</mark> 55			pus	h ebp					Entry	/Point			
		•	00417	'E01 8B	EC		mov	ebp,esp									
			00417	'E03 83	EC OC		sub	esp,c									
			00417	[E06] 57			pus	h edi									
			0041/	(E0/ C/	45 FC 00	000000	mov	dword pt	r ss	: Lepb-	4],0						
			00417	EOE 88	55 08	100	mov	eax, dword	α ρτι	SS:L	epp+	5					
			00417	ELL 89	15 DCC14	100	mov	dword pt	r as	[41C1		eax	eax:	0<111/LMC	EU		
			00417	2510 69	45 FC 00	100	mov	dword pt	r us r cc	·[41C1		anh					
			00417	7524 58	27EEEE	E	cal	1 emotet 4	4170	50	41,0						
			00417	7E29 E8	8204000	0	cal	1 emotet.	41826	RŐ							
			00417	7E2E - EB	00		imp	emotet.4	17F3(	0							
			00417	7E30 E8	6B02000	0	cal	1 emotet.4	4180	40							
			00417	7E35 68	582C000	0	pus	h 2C58									
			00417	7E3A E8	11FFFFF	F	cal	<pre>1 emotet.4</pre>	417D	50							
			00417	'E3F 83	C4 04		add	esp,4									
			00417	7E42 C7	05 C8C14	100 000000	C mov	dword pt	r ds:	:[41C1	C8],(	0					
			00417	'E4C A1	C8C1410	0	mov	eax,dword	d pti	_ds:[	41 <u>c</u> 1(	[8]					
			00417	[E51 A3	CCC1410	0	Jmov	dword pt	r ds	:[41C1	ccl,	eax					
			0041/	(E56 C/	05 C4C14	100 020000	Q mov	dword pt	r ds	:[41C1	C4],						
			00417	E00 C7	45 F4 00	1000000	mov	aword pti	r ss	:Lepb-		. 01					
			00417	1 E07 0B	UD AOCI4	100	niov	b ocy	a pri	r us:E	41CL/	40]					
			00417		15 00004	100	mov	edy dword	d nti	ds · D	4100	nc1	edv."	'u∠ìfì\fwc	eñ"		
			00417	7674 52	13 00004	100	nus	h edx	a pri	. C	4100	<u>,                                    </u>	edx.	'uzìfì\fwc	EÜ"		
			00417	F75 F8	в600000	0	cal	l emotet.4	417F	30			- Curr.	0.11.(114	20		
			00417	7E7A 83	C4 08		add	esp.8									
			00417	7E7D A3	00C2410	0	mov	dword pt	r ds	:[41C2	00].	eax					
			00417	7E82 A1	C8C1410	0	mov	eax, dword	d ptr	r ds:[	41c1	[8]					
			00417	7E87 3B	05 A4C14	100	cmp	eax,dword	d ptr	r ds:[	41C1/	44]					
			00417	7E8D 🚽 72	02		jb	emotet.41	7E91								
			1011-		70		1 1 1 1 1	omotot A	1760/	-			·				



We set a breakpoint over the jump instruction jmp ecx at the address 00417F1F by pressing f2 . because after this jump the unpack process will happen.

CPU	📄 Log	•	Notes	Breakpoint	ts 📟 Memory Maj	p 🗐 Call Stack	📽 SEH	🖸 Script	2	Symbols	
	-			Breakpoint           00417ED4           00417EDA           00417EE2           00417EE2           00417EE1           00417EF3           00417FF           00417FF           00417FF           00417FF           00417FF           00417F1           00417F13           00417F14           00417F15           00417F16           00417F17           00417F18           00417F20           00417F21           00417F25           00417F26           00417F27           00417F28           00417F29           00417F28           00417F28           00417F28           00417F28	IS I Memory Ma 8915 CCC14100 A1 OcC04100 0345 F4 0305 C8C14100 A3 C8C14100 880D A8C14100 890D A8C14100 833D E8C14100 0F85 54FFFFF E8 CFFEFFFF E8 CFFEFFFF 88FF B9 C47B4100 88FF 83E9 04 88FF 5F 88E5 5D C3 CC CC CC CC CC CC	<pre>mov dword ptr mov eax,dword add eax,dword add eax,dword add eax,dword add eax,dword sub ecx,dword sub ecx,dword sub ecx,dword fr cmp dword ptr jne emotet.42 call emotet.42 call emotet.43 call emotet.43 call emotet.44 call emotet.44 call emotet.45 call emot</pre>	7 3 5EH 5 ds:[41C1 6 ptr ds:[ 7 ds:[41C1 9 ptr ds:[ 9 ptr ds:[41C1 9 ptr ds:[ 9 ptr	CC],edx 41c00c] ebp-C] 41c1c8] c8],eax 41c1A8] 41c1AC] A8],ecx E8],0	edx:"	Symbols U<î∱î\f₩Ç	EU.
				00417F2B 00417F2C 00417F2D	cc						
				00417F2E 00417F2F	cc çç	int3 int3					

### Figure(3):

Then we press **f9** to hit the breakpoint.

CPU	🍃 Log	Notes	<ul> <li>Breakpoints</li> </ul>	📟 Memory Map	Call Stack	🕿 SEH	o Script	🔮 Symbols	Source	References	*
EIP			00417ED4         89           00417ED4         004           00417ED5         03           00417ED6         03           00417ED7         03           00417ED8         A3           00417E45         A3           00417E78         28           00417E78         83           00417FF8         83           00417F18         88           00417F18         88           00417F18         88           00417F18         88           00417F19         88           00417F18         88           00417F19         88           00417F21         56           00417F24         50           00417F25         50           00417F26         50           00417F27         50           00417F28         50           00417F29         50           00417F2	15 CCC14100 OCC04100 45 F4 05 C8C14100 C8C14100 00 A8C14100 00 A8C14100 00 A8C14100 00 A8C14100 00 A85 S4FFFFF CFFEFFF FF C47B4100 FF E1 E5	<pre>mov dword pt mov eax,dwor add eax,dwor add eax,dwor add eax,dwor mov dword pt mov ecx,dwor mov dword pt ine emotet. mov edi,edi sub ecx,4 mov edi,edi sub ecx,4 mov edi,edi sub ecx,4 mov edi,edi sub ecx,4 mov edi,edi int3 int3 int3 int3 int3 int3 int3 int</pre>	rr ds:[41C1 rd ptr ds:[ rd ptr ds:[ rd str ds:[41C1 rd str ds:[41C1 rd str ds:[41C1 tr ds:	CC].edx 41c00C] ebp-C] 41c1C8] C8].eax 41c1A8] 41c1AC] A8].ecx E8].0	0041C1CC:&"ຳb' ecx:"Uເງ້ຍື" ecx:"Uເງ້ຍື" ecx:"Uເງ້ຍື" 0041C1E8:&"mkr ecx:"Uເງ້ຍື" ecx:"Uເງ້ຍື"	ibibibibibibi	bîbîbîbîbîbîbî fw111112clddr11	ofbfbf

Figure(4):

Then press **f7** we jump to another function. If we analyze this function we will notice that:

#### epilogue

🖾 CPU 🍃 Log	📄 Notes 🎴 Breakpo	oints 🛛 📟 Memory Map	🧊 Call Stack	📽 SEH	o Script
EIP ECX	00417BC0 00417BC1 00417BC3 00417BC3 00417BC5 00417BCA 00417BD0 00417BD5 00417BD5 00417BD8 00417BDD	55 8BEC CEB 00 CA1 F0C14100 8B25 BCC14100 A1 F0C14100 58 8BE8 A1 F0C14100 A1 D8C14100	push ebp mov ebp,esp jmp emotet.41 mov eax,dword mov esp,dword mov eax,dword pop eax mov ebp,eax mov eax,dword mov eax,dword	.7BC5 ds:[   ptr ds:[   ptr ds:[   ptr ds:[   ptr ds:[	41C1F0] 41C1BC] 41C1F0] 41C1F0] 41C1F0]

Figure(5):

# Abnormal prologue

00417C34 00417C3A 00417C3B 00417C3C 00417C3D	8B15 B4C14100 52 C3 5D C3 C3	mov edx,dword ptr ds:[41C1B4] push edx ret pop ebp ret	mov edx,dword ptr ds:[41C1B4] push edx ret pop ebp ret
--	---	--	--

Figure(6):

The last figure shows the abnormal prologue (1). And (2) is a suspecious instruction which we will know late that is new VirtualAlloc.

So we set a breakpoint over this instruction mov edx, dword ptr ds: [41C1B4] by f2 and press f9 to hit the breakpoint.



Figure(7):

Now if we follow in dump We see that it's allocating memory.

Then Press f8 it will push edx to the stack which is the value of mov edx, dword ptr ds: [41C1B4].

Then Press **f8**. There is **abnormal ret**. Normal **ret** value will get back to wherever it was called from.

Here it return to this address 002302F0 . Which will be the address of the unpacking section.



Figure(9):

So step over it.

In the next part we will see functions (Unpacking routine) and we will explain it on the fly in the next figure

32	CPU	Log	1	Notes	<ul> <li>Breakpoints</li> </ul>	📟 Memory Map	Call Stack	۳	SEH	o	Script	2	Symbols	Sou	urce
EIP	EDX				002302F0         5           002302F1         81           002302F3         82           002302F3         82           002302F3         82           002302F3         82           0023030C         62           0023030C         83           0023030C         83           0023030F         83           0023031D         83           0023031B         64           0023031D         64           0023031D         64           0023032C         55           0023032D         81           0023032D         81           0023032D         81           0023032B         83           0023032B         83           0023033D         83           0023033D         83           0023033D         83           0023033D         83           0023033D         83           0023033C         83           00230340         64           00230340         84           00230340         84           00230340         84           00230350         88           00230340<	5 5 5 5 5 5 5 5 5 5 5 5 5 5	push ebp mov ebp,esp sub esp,80 mov dword ptr mov dword ptr jmp 230312 mov eax,dword add eax,1 mov dword ptr jae 230320 push 58 push 0 lea ecx,dword push ecx call 22F880 add esp,c jmp 230309 mov eax,dword ptr lea edx,dword ptr lea edx,dword push edx call 22F830 add esp,4 call 22F730 mov dword ptr mov eax,dword ptr lea edx,dword ptr lea edx,dword push edx call 22F830 add esp,4 call 22F730 mov dword ptr mov eax,dword ptr mov eax,dword ptr	d pt d pt d pt d pt d pt ss d pt	:[ebp- :[ebp- :[ebp- :[ebp- r ss:[ :[ebp- r ss:[ :[ebp- r ss:[ :[ebp- r ss:[ :[ebp- r ss:[	c],0 ebp-( c],e c],e ebp+: 8],e ebp+: 7c],, ebp-( 7c],.	c] ax 2DCD5 68] 4] ax 8] eax 8] eax 68] 68]	[ebp- [ebp- [ebp- [ebp-	-C]:"MZ" -C]:"MZ" -C]:"MZ" -C]:"MZ" -C]:"MZ" "U<ìë"		

Figure(10):

Keep stepping over untill you reach the breakpoint.

Then we see this funtion and step into f7.

/ -	00230320	0013 01		
•	00230330	8945 F8	mov dword ptr ss:[ebp-8],eax	
•	00230333	8B45 08	mov eax,dword ptr ss:[ebp+8]	[ebp+8]:"MZ"
•	00230336	8945 8C	mov dword ptr ss:[ebp-74],eax	[ebp-74]:"MZ"
•	00230339	896D 98	mov dword ptr ss:[ebp-68],ebp	
•	0023033C	8D55 98	lea edx,dword ptr ss:[ebp-68]	
•	0023033F	52	push edx	
EIP	00230340	E8 EBF4FFFF	call 22F830	
•	00230345	83C4 04	add esp,4	
•	00230348	E8 E3F3FFFF	call 22F730	
•	0023034D	8945 84	mov dword ptr ss:[ebp-7C],eax	
•	00230350	8B45 F8	mov eax, dword ptr ss:[ebp-8]	
•	00230353	8945 A0	mov dword ptr ss:[ebp-60],eax	
÷	00000000			

Figure(11):

It uses stack strings. which is mentioned above In introduction. (1) pushes them on the stack.

🕮 CPU 🍃 Log 📑 No	otes • Breakpoints	📟 Memory Map	Call Stack	📽 SEH	o Script	👰 Symbols	Source	,																		
CPU 💽 Log 🖬 N	Preakpoints           0022F830         55           0022F831         88           0022F833         83           0022F836         C64           0022F836         C64           0022F837         C64           0022F838         C64           0022F838         C64           0022F838         C64           0022F838         C64           0022F838         C64           0022F838         C64           0022F836         C64           0022F852         C64           0022F854         C64           0022F855         C64           0022F856         C64           0022F858         C64           0022F858         C64           0022F858         C64           0022F858         C64           0022F857         C64           0022F877         C64           0022F878         C64           0022F878         C64           0022F878         C64           0022F878         C64           0022F878         C64           0022F888         C64           0022F888         C64 <t< th=""><th>Memory Map Memory Memory Map Memory Map Memory Map Memory Map Memory Map</th><th>Call Stack push ebp mov ebp,esp sub esp,30 mov byte ptr mov byte ptr</th><th>%%         SEH           SS:         [ebp-2           SS:         [ebp-1           SS:         [ebp-1      SS:         [ebp-1      <tr <<="" t="" th=""><th>Script 8],4C 7],6E 6],64 4],4C 2],62 2],62 2],62 1],72 6],73 6],74 6],75 6</th><th>2 Symbols</th><th>Source</th><th></th></tr><tr><th></th><th>0022F88E C64 0022F892 C64 0022F896 C64 0022F896 C64</th><th>45 EF 32 45 FO 2E 45 F1 64 45 F2 6C</th><th>mov byte ptr mov byte ptr mov byte ptr mov byte ptr</th><th>ss:[ebp-1 ss:[ebp-1 ss:[ebp-F ss:[ebp-F</th><th>1],32 0],2E ],64 1.6C</th><th>32:'2' 2E:'.' 64:'d' 6c:'1'</th><th></th><th></th></tr><tr><th></th><th>0022F89E C64 0022F8A2 C66 0022F8A6 E8 0022F8A6 894 0022F8AE 894 0022F8AE 89 0022F8B3 894 0022F8B6 884</th><th>15 F3 6C 15 F4 00 65050000 15 D0 7DFEFFFF 15 D4 15 D4</th><th>mov byte ptr mov byte ptr call 22FE10 mov dword ptr call 22F730 mov dword ptr mov eax.dwor</th><th>ss:[ebp-D ss:[ebp-C r ss:[ebp- r ss:[ebp- d otr ss:[</th><th>],6C 1 x 2C],eax ebp-2C]</th><th>60:'1'</th><th></th><th></th></tr></th></t<>	Memory Map Memory Memory Map Memory Map Memory Map Memory Map Memory Map	Call Stack push ebp mov ebp,esp sub esp,30 mov byte ptr mov byte ptr	%%         SEH           SS:         [ebp-2           SS:         [ebp-1           SS:         [ebp-1      SS:         [ebp-1 <tr <<="" t="" th=""><th>Script 8],4C 7],6E 6],64 4],4C 2],62 2],62 2],62 1],72 6],73 6],74 6],75 6</th><th>2 Symbols</th><th>Source</th><th></th></tr> <tr><th></th><th>0022F88E C64 0022F892 C64 0022F896 C64 0022F896 C64</th><th>45 EF 32 45 FO 2E 45 F1 64 45 F2 6C</th><th>mov byte ptr mov byte ptr mov byte ptr mov byte ptr</th><th>ss:[ebp-1 ss:[ebp-1 ss:[ebp-F ss:[ebp-F</th><th>1],32 0],2E ],64 1.6C</th><th>32:'2' 2E:'.' 64:'d' 6c:'1'</th><th></th><th></th></tr> <tr><th></th><th>0022F89E C64 0022F8A2 C66 0022F8A6 E8 0022F8A6 894 0022F8AE 894 0022F8AE 89 0022F8B3 894 0022F8B6 884</th><th>15 F3 6C 15 F4 00 65050000 15 D0 7DFEFFFF 15 D4 15 D4</th><th>mov byte ptr mov byte ptr call 22FE10 mov dword ptr call 22F730 mov dword ptr mov eax.dwor</th><th>ss:[ebp-D ss:[ebp-C r ss:[ebp- r ss:[ebp- d otr ss:[</th><th>],6C 1 x 2C],eax ebp-2C]</th><th>60:'1'</th><th></th><th></th></tr>	Script 8],4C 7],6E 6],64 4],4C 2],62 2],62 2],62 1],72 6],73 6],74 6],75 6	2 Symbols	Source			0022F88E C64 0022F892 C64 0022F896 C64 0022F896 C64	45 EF 32 45 FO 2E 45 F1 64 45 F2 6C	mov byte ptr mov byte ptr mov byte ptr mov byte ptr	ss:[ebp-1 ss:[ebp-1 ss:[ebp-F ss:[ebp-F	1],32 0],2E ],64 1.6C	32:'2' 2E:'.' 64:'d' 6c:'1'				0022F89E C64 0022F8A2 C66 0022F8A6 E8 0022F8A6 894 0022F8AE 894 0022F8AE 89 0022F8B3 894 0022F8B6 884	15 F3 6C 15 F4 00 65050000 15 D0 7DFEFFFF 15 D4 15 D4	mov byte ptr mov byte ptr call 22FE10 mov dword ptr call 22F730 mov dword ptr mov eax.dwor	ss:[ebp-D ss:[ebp-C r ss:[ebp- r ss:[ebp- d otr ss:[	],6C 1 x 2C],eax ebp-2C]	60:'1'		
Script 8],4C 7],6E 6],64 4],4C 2],62 2],62 2],62 1],72 6],73 6],74 6],75 6	2 Symbols	Source																								
	0022F88E C64 0022F892 C64 0022F896 C64 0022F896 C64	45 EF 32 45 FO 2E 45 F1 64 45 F2 6C	mov byte ptr mov byte ptr mov byte ptr mov byte ptr	ss:[ebp-1 ss:[ebp-1 ss:[ebp-F ss:[ebp-F	1],32 0],2E ],64 1.6C	32:'2' 2E:'.' 64:'d' 6c:'1'																				
	0022F89E C64 0022F8A2 C66 0022F8A6 E8 0022F8A6 894 0022F8AE 894 0022F8AE 89 0022F8B3 894 0022F8B6 884	15 F3 6C 15 F4 00 65050000 15 D0 7DFEFFFF 15 D4 15 D4	mov byte ptr mov byte ptr call 22FE10 mov dword ptr call 22F730 mov dword ptr mov eax.dwor	ss:[ebp-D ss:[ebp-C r ss:[ebp- r ss:[ebp- d otr ss:[	],6C 1 x 2C],eax ebp-2C]	60:'1'																				

Figure(12):

To get out from this function find **ret** and set a breakpoint then press **f**9

And these functions do the same as above. So step over them **f8**. to see what inside a function without executing it: Double click over a function and press - button to get out.

😟 CPU	📄 Log	Notes	Breakpoints	🛲 Memory Map	🧊 Call Stack	📽 SEH	Script	🔮 Symbols	So
			0023033F         52           00230340         E8           00230345         832           00230345         832           00230345         894           00230350         884           00230350         894           00230350         894           00230355         894           00230356         884           00230357         894           00230358         894           00230359         894           00230356         884           00230357         895           00230368         844           00230368         844           00230368         894           00230368         894           00230374         895           00230377         676           00230378         894	EBF4FFFF 4 04 E3F3FFFF 5 84 5 78 15 A0 10 8C 10 8C 10 AC 15 AC 15 AC 15 AC 15 AC 15 AC 15 AC 15 AC 15 F0 15 F0 15 F0 15 90 1748 16 10 A4	push edx call 22F830 add esp.4 call 22F730 mov dword ptr mov eax,dworn mov dword ptr mov edx,dword mov dword ptr mov eax,dword mov eax,dword mov eax,dword mov eax,dword mov edx,dword mov edx,dword mov edx,dword mov eax,dword mov eax,dword mov eax,dword mov eax,dword mov zx ecx,word	r s:[ebp- d ptr ss:[ r ss:[ebp- d ptr ss:[ebp- r ss:[ebp-	7C],eax ebp-8] 60].eax ebp-74] 554].ecx ebp-54] ebp-54] ebp-10] eax+3C] 10],ecx ebp-10] [eax+16] 70],edx ebp-70] [eax+16] 560,ecx	[ebp-74]:"MZ" [ebp-54]:"MZ" 54]:"MZ" 1 4]:"MZ"	
EIP			0023037E         E8           00230383         05           00230388         894           00230388         894           00230388         894           00230388         894           00230388         894           00230388         894           00230390         894           00230390         894           00230390         894           00230390         894           00230390         894           00230390         894           00230340         884           002303A3         51           002303A4         E8           002303A9         836	ADE3FFFF 20114000 55 88 55 88 9853FFF 24114000 55 88 10 80 87F6FFFF 4 04	call 22F730 add eax,emot mov dword pt mov edx,dword mov dword pt call 22F730 add eax,emot mov dword pt mov ecx,dword push ecx call 22FA30 add esp.4	et.401120 r ss:[ebp- d ptr ss:[ d ptr ds] r s:[ebp- et.401124 r ss:[ebp- d ptr ss:[	75],eax ebp-78] edx] 80],eax 78],eax ebp-80]		

Figure(13):

Untill we get to this last function. step into f7.

0 0 0	002303D0 002303D3 002303D6 002303D7 002303DA	83C4 08 8B4D 94 51 8D55 98 52	add esp,8 mov ecx,dword ptr ss:[ebp-6C] push ecx lea edx,dword ptr ss:[ebp-68] push edx	[ebp-6c]:"MZ" ecx:"MZ"
ETP >=	002303DB	E8 A0EBEEEE	call 22FF80	
	002303E0	85C0	test eax,eax	
	002303E2	33C0	xor eax,eax	
· · · · · ·	002303E6 🗸 🗸	EB 1A	jmp 230402	
i>•	002303E8	837D A0 00	cmp dword ptr ss:[ebp-60],0	
	002303EC	74 09	je 2303F7	

Figure(14):

After we get into the function we need to analyze it **carefully** 

•	002303D0	83C4 08	add esp,8	
•	002303D3	8B4D 94	mov ecx,dword ptr ss:[ebp-6C]	[[ebp-6C]:"MZ"
•	002303D6	51	push ecx	ecx:"MZ"
•	002303D7	8D55 98	lea edx,dword ptr ss:[ebp-68]	
•	002303DA	52	push edx	
EIP	002303DB	E8 A0FBFFFF	call 22FF80	
•	002303E0	85C0	test eax,eax	
r®	002303E2	75 04	jne 2303E8	
•	002303E4	33C0	xor eax,eax	
•	002303E6	EB 1A	jmp 230402	
L>0	002303E8	837D A0 00	cmp dword ptr ss:[ebp-60],0	
®	002303EC	74 09	je 2303F7	

Figure(14):

As we can see **call edx** is calling **VirtualAlloc**:

CPU	📄 Log	📄 Notes	<ul> <li>Breakpoints</li> </ul>	📟 Memory Map	🧊 Call Stack	📽 SEH	o Script	🔮 Symbols	Source	🧢 References
			0022FF80 0022FF81 88E 0022FF83 892 0022FF85 884 0022FF85 884 0022FF85 882 0022FF92 0022FF92 0022FF92 884 0022FF98 884 0022FF98 884 0022FF98 884 0022FF98 884 0022FF98 884 0022FF84 886 0022F784 886 0022FF84 886 886 0022FF84 886 886 0022FF84 886 886 886 886 886 886 886 886 886 8	C C 28 C 28 S 0C S 0C S 0C S 0C S 0C S 24 S 24 S 24 S 24 S 24 S 24 S 24 S 24 S 25 S 24 S 24 S 24 S 25 S 24 S 24 S 24 S 25 S 24 S 24 S 25 S 24 S 24 S 25 S 24 S 24 S 24 S 25 S 25 S 26 S 2	push ebp mov ebp,esp sub esp,28 mov dword pt mov dword pt mov ecx,dwor mov edx,dwor mov edx,dwor mov edx,dwor mov edx,dwor mov edx,dwor mov edx,dwor mov edx,dwor mov edx,dwor mov edx,dwor push 40 push 3000 mov edx,dwor push 40, push 40, mov edx,dwor mov edx,dwor mov edx,dwor mov edx,dwor mov edx,dwor mov edx,dwor push 40, mov edx,dwor	d ptr ss:[ d ptr ss:[ d ptr ss:[ d ptr ds:[ d ptr ds:[]	ebp+C] 24],eax ebp-24] ebp+C] ecx+3C] 1C],edx ebp-1C] edx+ecx+18] 14],eax ebp-1C] ecx+30] 4],edx ebp-4] ebp+8] ecx+24]	[ebp+C] [ebp-24 [ebp-24 [ebp+C] [ebp+14 ecx+50	:"MZ"  ]:"MZ" ']:"MZ" :"MZ" 4]:".text" !]:".text"	
EIP			0022FFC5         FFL           0022FFC7         894           0022FFC0         884           0022FFC0         884           0022FFC0         884           0022FFC0         51           0022FFD1         885           0022FFD4         52	22 15 E8 15 E4 18 54 155 0c	call edx mov dword pt mov eax,dwor mov ecx,dwor push ecx mov edx,dwor push edx	r ss:[ebp- d ptr ss:[ d ptr ds:[ d ptr ss:[	18],eax ebp-1C] eax+54] ebp+C]	[ebp-18 [ebp+C]	3]:"мz"  :"мz"	
edx= <kern< td=""><th>el32.virtu</th><th>alAlloc&gt;</th><td></td><td></td><td></td><td></td><th></th><td></td><td></td><td></td></kern<>	el32.virtu	alAlloc>								
0022FFC5										

push 40 **RWX** which is our indicator to know that this call could be VirtuallAlloc

Figure(15):

One step over **f8** and we will get the adress of newly memory allocated in **eax** 

🔯 CPU 🍃 Log 📫 Notes	🍨 Breakpoints 🖙 Memory M	p 🗊 Call Stack 💁 SEH 🗾 Script	💁 Symbols 🗢 Source 🌛 References 🛸 Threads 🐇 Handles 🦿 Trace	
0	0022FFA2 8D440A 18 0022FFA6 8945 EC	<pre>lea eax,dword ptr ds:[edx+ecx+18] mov dword ptr ss:[ebp-14].eax</pre>	[ebp-14]:".text"	A Hide FPU
<b>cæ</b> →	0022FFFA         8450         141           0022FFAC         8451         10           0022FFAC         8631         10           0022FFAC         8631         10           0022FFAC         863         10           0022FFAC         863         17           0022FFAC         863         17           0022FFAC         863         17           0022FFAC         863         14           0022FFAC         864         14           0022FFAC         864         14           0022FFAC         864         14           0022FFAC         864         14           0022FFAC         853         14           0022FFFAC         853         14           0022FFFAC         853	<pre>mov text_duard prf st;[text_st; mov etxt_duard prf st;[text_st; mov etxt_duard prf st;[text_st; mov_etxt_duard prf st;[text_st] mov_etxt_duard prf st;[text_st] mov_etxt_duard prf st;[text_st] mov etxt_duard prf st;[text_st] mov etxt_st] mov etxt_st</pre>	Islop-1-1: Text     address of the new mamory       edx: & f & k vos <sup>m</sup> follow this in dump       Islop-1: %2 <sup>m</sup> follow this in dump       [ebp-1:3]: %2 <sup>m</sup>	Construction     Construction
dword ptr ss:[ebp-18]=[0018FED	8 &"MZ"]=75A30000 "MZ"			3: [esp+C] 00240088 4: [esp+10] 75A30000 "MZ"
0022FFC7				3: [esp+14] 00240160 .text
Dump 1 Dump 2	Dump 3 🕮 Dump 4 🕮 Dum	5 😂 Watch 1 🖂 Locals 🔑 S	0018FEC8 00000000 0018FECC 00240000 "MZ"	
$\begin{array}{c} Address & bx \\ 0005000100000000000000000000000000000$	A         A           00 <td>new</td> <td>00135500 0005000 00135500 0005000 errel 32.73A30000 00135500 0024000 errel 32.73A30000 00135550 00000 errel 32.73A30000 00135500 errel 32.73A30000 errel 32.73A30000 00135500 errel 32.73A30000 errel 32.73A30000 errel 32.73A30000 001355500 errel 32.73A30000 errel 32.73A30000 errel 32.73A30000 errel 32.73A30000 errel 32.73A30000 errel 32.73A300000 errel 32.73A300000 errel 32.73A300000 errel 32.73A300000 errel 32.73A30000 errel 32.73A300000 errel 32.73A300000000000000000000000000000000000</td> <td>róm 0022FF80</td>	new	00135500 0005000 00135500 0005000 errel 32.73A30000 00135500 0024000 errel 32.73A30000 00135550 00000 errel 32.73A30000 00135500 errel 32.73A30000 errel 32.73A30000 00135500 errel 32.73A30000 errel 32.73A30000 errel 32.73A30000 001355500 errel 32.73A30000 errel 32.73A30000 errel 32.73A30000 errel 32.73A30000 errel 32.73A30000 errel 32.73A300000 errel 32.73A300000 errel 32.73A300000 errel 32.73A300000 errel 32.73A30000 errel 32.73A300000 errel 32.73A300000000000000000000000000000000000	róm 0022FF80

Figure(16):

Then keep stepping over and get to this function **call 22FBC0** and then one more step over. As we see in the dump section, the function writes over the newly memory allocate with the exe file.

	0022FFD8 50 0022FFD9 E8 E2EBEEEE	push eax	
	0022FFDE 83C4 0C	add esp,C	
	0022FFE1 C745 F4 00000000 0022FFE8 × EB 09	jmp 22FFF3	
	0022FFEA 8B4D F4	mov ecx,dword ptr ss:[ebp-C]	
	0022FFED 85CI 01 0022FFF0 894D F4	mov dword ptr ss:[ebp-C],ecx	
	0022FFF3 8B55 E4	mov edx, dword ptr ss: [ebp-1C]	
	0022FFFA 3945 F4	cmp dword ptr ss:[ebp-C],eax	
[@	0022FFFD • 0F83 86000000	jae 230089	Falsa 147." asua"
	00230006 8379 14 00	cmp dword ptr ds:[ecx+14],0	[eop-14]: .text
r	0023000A V 74 6F	je 23007B	[ohn 14]," toxt"
	0023000F 837A 10 00	cmp dword ptr ds:[edx+10],0	[[eop-14]text
	00230013 • 74 66 00230015 8845 FC	je 23007B mov eax dword ptr ss:[ebp_14]	[ohn_14]." toxt"
	00230018 8B48 10	mov ecx, dword ptr ds:[eax+10]	[[eop-14] text
	< <b>C</b>		
0022FBC0			
0022FFD9			
0022FFD9 객패 Dump 1 객패 Dump 2 객패	Dump 3 🖷 Dump 4 🖷 Dump 5	😂 Watch 1 🖂 Locals 名 Struct	
0022FFD9 I Dump 1 Address Hex	Dump 3 🕮 Dump 4 🕮 Dump 5	😂 Watch 1 🖂 Locals 👂 Struct	
0022FFD9 The Dump 1 Address Hex 00250000 4D 5A 90 00 03 00 00 0025001 0 58 00 00 00 00 00 00	Dump 3 7 Dump 4 7 Dump 5	😂 Watch 1 🖂 Locals <b>2</b> Struct	
0022FFD9 THE Dump 1 Address Hex 00250000 4D 5A 90 00 03 00 00 00250010 B8 00 00 00 00 00 00 00250020 00 00 00 00 00 00	Dump 3         Image: Dump 4         Image: Dump 5           00         04         00         00         00         FF         FF         00         00         MZ            00         04         00         00         00         00         00         00         00            00         00         00         00         00         00         00	₩atch 1 I≃ Locals <b>2</b> Struct	
0022FFD9 The Dump 1 The Dump 2 The Dump 1 The Dump 2 The Dump 1 The Dump 2	Dump 3         Image: Dump 4         Image: Dump 5           00         04         00         00         00         FF         FF         00         00         MZ            00         04         00         00         00         FF         FF         00         00         MZ            00         00         00         00         00         00         00            00         00         00         00         00         00         00            00         00         00         00         00         00         00            00         00         00         00         00         00         00            00         00         00         00         00         00         00	Watch 1 I≃ Locals 2 Struct	
0022FFD9	Dump 3         Image: Dump 4         Image: Dump 5           00         04         00         00         00         FF         FF         00         00         MZ            00         04         00         00         00         FF         FF         00         00         MZ            00         00         00         00         00         00         00            00         00         00         00         00         00         00            00         00         00         00         00         00         00            00         00         00         00         00         00         00            00         00         00         00         00         00         00            00         00         00         00         00         00             00         00         00         00         00         00             00         145         145         145         165	Watch 1 I⊨ Locals 2 Struct 	
0022FFD9	Dump 3         Image: Dump 4         Image: Dump 5           00         04         00         00         FF         FF         00         00         MZ            00         04         00         00         00         FF         FF         00         00         MZ            00	Watch 1 I⊨ Locals 2 Struct ÿÿ 	
0022FFD9	Dump 3         Image: Dump 4         Image: Dump 5           00         04         00         00         FF         FF         00         04           00         04         00         00         00         FF         FF         00         04           00         04         00	Watch 1 I⊨ Locals 2 Struct ÿÿ 	
0022FFD9  Address Hex 00250000 4D 5A 90 00 03 00 00 00250020 00 00 00 00 00 00 00 00250020 00 00 00 00 00 00 00 00250020 00 00 00 00 00 00 00 00250030 00 00 00 00 00 00 00 0025004 0E 1F BA 0E 00 B4 09 00250050 69 73 20 70 72 6F 67 00250060 74 20 62 65 20 72 75 00250070 6D 6F 64 65 22 0D 00 00250080 AF DF 0D CB EB BE 63 00250090 96 C7 86 98 CF BE 63	Dump 3         Image: Dump 4         Image: Dump 5           00         04         00         00         0F         FF         00         00         MZ         ASCII           00         04         00         00         00         FF         00         00         MZ            00	Watch 1     Irel Locals     2     Struct	

#### Figure(17):

When keep stepping we see that it's copying files to the exe file

#### .text

0022FFDE 83C4 0C add esp,C 0022FFE1 C745_F4 00000000 mov dword ptr ss:[ebp-C],0	
0022FFE1 C745 F4 00000000 mov dword ptr ss:[ebp-C],0	
0022FFEA 8B4D F4 mov ecx.dword ptr ss:[ebp-C]	
0022FFED 83C1 01 add ecx.1 ecx:"MZ"	
0022FFF0 894D F4 mov dword ptr ss:[ebp-c].ecx	
→ 0022FFF3 8B55 E4 mov edx.dword ptr ss:[ebp-1C]	
0022FFF6 0FB742 06 movzx eax.word ptr ds:[edx+6] eax:".text"	
0022FFFA 3945 F4 cmp dword ptr ss:[ebp-c].eax	
(	
00230003 8B4D EC mov ecx,dword ptr ss:[ebp-14] [ebp-14]:".text"	
00230006 8379 14 00 cmp dword ptr ds:[ecx+14],0	
r 0023000A v 74 6F je 23007B	
0023000C 8B55 EC mov edx,dword ptr ss:[ebp-14] [ebp-14]:".text"	
0023000F 837A 10 00 cmp dword ptr ds:[edx+10],0	
00230013 × 74 66 je 230078	
00230015 8B45 EC mov eax,dword ptr ss:[ebp-14] [ebp-14]:".text"	
00230018 8B48 10   mov ecx,dword ptr ds:[eax+10]   ecx:"MZ"	
0023001B 8B55 E4 mov edx,dword ptr ss:[ebp-1C]	
0023001E  8B42 3C   mov eax,dword ptr ds:[edx+3C]   eax:".text"	
00230021 8D4401 FF  lea eax,dword ptr ds:[ecx+eax-1]  eax:".text"	
00230025 8B4D E4 mov ecx,dword ptr ss:[ebp-1C]	
00230028 33D2 xor edx,edx	
0023002A F771 3C div dword ptr ds:[ecx+3C]	
0023002D 8B55_E4 mov_edx,dword_ptr_ss:[ebp-1C]	
00230030 0FAF42 3C imuleax,dword ptr ds:[edx+3C] [eax:text_	
00230034 50 push eax  eax:".text"	
00230035 8845 EC mov eax,dword ptr ss:[ebp-14] [ebp-14]; ".text"	
00230038 8B4D 0C mov ecx,dword ptr ss:[ebp+C] [ebp+C]: "MZ"	
EIP add_ecx,dword ptr ds:[eax+14] ecx: MZ	

Figure(18):

then .rdata then .data then .reloc

Untill we get to the last ret 8 as shown.

>	002301DF 002301E4 002301E7 002301EC	E8 7CF9FFFF 83C4 04 B8 01000000 8BE5	call 22FB60 add esp,4 mov eax,1 mov esp.ebp
	002301EE	5D	pop ebp
EIP $\rightarrow 0$	002301EF	C2 0800	ret 8
•	002301F2	СС	int3
•	002301F3	сс	int3
•	002301F4	СС	int3

Figure(19):

**Stay awake** our file is almost finished. After the second **ret**.

🔯 CPU	🍃 Log	📫 Note	es • Breakpoir	its 🚥 Memory Map	🧃 Call Stack		Script	🙎 Sy	ymbols 🗢 Source	References		🔒 Handles						
EIP EDX			0040C9A0	55 8000	push ebp												Hide FPU	
			OddaCSAL         OddaCSAL	8814 - F8 814 -	mov ebp esp , FFF sub esp, FFF push esp, FFF push esp push esp call 408580 call 408580 call 408580 call 408580 call 408580 push 204 push eav push o call 408580 call 408580 push 204 mov ax word mov ax word test ax, ax je 40c365 movz cas, a leas cas, a movz ca	rd ptr ss: ptr ds:[41 ptr ss:[es d ptr ss:[es d ptr ss: i,1003 k rd ptr ds: k ord ptr ds nd ptr ds 11.77265A3	[esp+47C] 9868] [esp+478] [esp] [ecx+2] ::[ecx] BE		[esp+47C]:"1strca eax:"M2" ecx:"M2" ecx:"M2" eax:"M2" eax:"M2" eax:"M2" eax:"M2" eax:"M2" eax:"M2" eax:"M2" eax:"M2"		The OEP ad remmber it	ldress.			EAX EBX ECX ECX EDX EDX ESI EDI EIP EFLAG CF 0 LastE LastS GS 000 CS 00 CS 00 Default	00400000 7EFDE000 00400000 00400000 00405400 0018FF94 0018FF94 0018FF94 0018FF94 00000000 00000000 00000000 00000000 0000	HIDE PPU "M22" "-450" (ERROR_INVALID (STATUS_INVALID + + S	PARAMETER) _PAGE_PROTEC
				På FoFć4000	Inov ecx140F0	EV			ecx: MZ					<u> </u>	2: e	sp+8 0018FFD4 sp+6 77279832	ntd]].77279832	î
ebp=0018FF9	4														4: [es	sp+10] 7EFDE00 sp+14] 772B021	0 8 ntdll.77280218	
0040C9A0																		Ļ
Dump 1           Addr ess         He           00400000         4D           00400010         B8           00400020         00           00400030         00           00400040         00           00400050         69           00400060         74           00400070         61           00400080         AF           00400080         AF           00400080         AF           00400080         AF           00400090         96	X 5A 90 0 00 00 0 00 00 0 1F BA 0 73 20 7 20 62 64 5 6F 64 5 6F 64	mp 2 100 003 000 000 000 000 000 000 000 000	Dump 3         III           00         00         04         00         00           00         00         04         00         00         00           00         0	Dump 4         IIII Dump 5           00         FF         00         M2           00         00         00         00         M2           00         80         00         00         M2           01         86         00         M2         M2           02         44         47         53         21         T         T           00         00         00         00         M2         M2         M2         M2           98         EA BE 63         91         F         M3         M3         M3	Watch 1 						018FF9 018FF9 018FF9 018FF9 018FF9 018FFA 018FFA 018FFA 018FFA 018FFA	C 725443430 0 7EFDE000 4 0018FFD4 8 77279832 C 7EFDE000 0 77280218 4 00000000 0 77280218 8 00000000 C 7EFDE000 0 00000000 C 7EFDE000 0 00000000 0 00000000	return to ke ntdll.772802 return to 77	erne132.75A43: td11.77279832 218 7087ADF from :				Ĵ
Command: Com				assembly instructs														Default 👻

Figure(20):

Now we can dump the unpacked exe. right click over eax and press Follow in Mwmory map

		Show FPU		
EAX 0040000C EBX 7EFDE00C ECX 0040000C	I_	 Modify value	Enter	
EDX 0040C9AC EBP 0018FF94 ESB 0018FF94	i Mi	Follow in Dump		
ESI 00000000	i III i	Follow in Dump		•
EIF 0040C9AC	72	Follow in Disassemble	r	
1 s 0000	•@	Follow in Memory Map	)	
OF 0 SF 0 DF CF 0 TF 0 IF	-	Copy value	Ctrl+C	
LastError 000	2 	Copy Symbol Value	Ctrl+S	R)
	•	Copy all registers		011
GS 002B FS 00 ES 0025 DS 00	4	Highlight	н	
	000	Zero	0	Ð
Default (stdcall)	<u>358</u>	Increment	+	Jnk
1: [esp+4] /EF 2: [esp+8] 001 3: [esp+C] 772	<u>35ğ</u>	Decrement		
4: [esp+10] 7E 5: [esp+14] 77.	<u>358</u>	Increase 4		
	35 <u>8</u>	Decrease 4		
3D Trom ???	t	Push		
from ???	1	Рор		

Sorry for this Mistake in the next figure. It's Follow in Mwmory map

Figure(21):

Then right click and then press Dump memory to File



Figure(22):

Now if we tried to open it in IDA. We will notice that's can't be analyzed

public start												
start proc near												
push	esi											
lahf												
popf												
mov	dword ptr	[ebp-4DCh],	587279C7h									
mov	dword ptr	[ebp-4D8h],	0ED7EE61Fh									
mov	dword ptr	[ebp-4D4h],	0AD05AF9Ch									
mov	dword ptr	[ebp-4D0h],	0B2015FBCh									
mov	dword ptr	[ebp-4CCh],	645D1A4Eh									
mov	dword ptr	[ebp-4C8h],	2E4F310Dh									
mov	dword ptr	[ebp-4C4h],	5A11C3CDh									
mov	dword ptr	[ebp-4C0h],	0A4FF6433h									
mov	dword ptr	[ebp-4BCh],	8BF1A0E2h									
mov	dword ptr	[ebp-4B8h],	0A69D0C0Fh									
mov	dword ptr	[ebp-4B4h],	0BC5118DCh									
mov	dword ptr	[ebp-4B0h],	0E7B92C0Ch									
mov	dword ptr	[ebp-4ACh],	0AD9CB0F4h									
mov	dword ptr	[ebp-4A8h],	8CD776B8h									
mov	dword ptr	[ebp-4A4h],	0B60EE1B9h									
mov	dword ptr	[ebp-4A0h],	6682D4AEh									
mov	dword ptr	[ebp-49Ch],	7CA422F3h									
mov	dword ptr	[ebp-498h],	0A37A83F0h									
mov	dword ptr	[ebp-494h],	647CC756h									
mov	dword ptr	[ebp-490h],	0E5155930h									
mov	dword ptr	[ebp-48Ch],	4E97325Fh									
mov	dword ptr	[ebp-488h],	0D2A17ECh									
mov	dword ptr	[ebp-484h],	0D1FFE464h									
mov	dword ptr	[ebp-480h],	710B8734h									
mov	dword ptr	[ebp-47Ch],	0F493AA9Fh									
th Hey V	iew-1)											

Figure(23):

So we need to repair section headers using **PE bear** tool.

Before

D	isasm: .	.text Ger	eral D	OS Hdr Rich	h kidr 🛛 File	Hdr Optiona	I Hdr Sect	ion Hdrs	🖿 Ba	aseReloc.
T Na	me	Raw Addr.	Raw size	Virtual Addr.	Virtual Size	Characteristics	Ptr to Reloc.	Num. of	Reloc.	Num. of Linenur
⊳	.text	400	CE00	1000	CC24	6000020	0	0		0
$\triangleright$	.rdata	D200	C00	E000	B00	40000040	0	0		0
$\triangleright$	.data	DE00	1200	F000	3DE4	C0000040	0	0		0
$\triangleright$	.reloc	F000	600	13000	58C	42000040	0	0		0

Figure(24):

# After editing

D	isasm:	.tex Ger	nera <mark>l</mark> D	OS dr Ric	h H <mark>d</mark> r File	Hdr Optiona	I Hdr Sec	tion Hdrs	🖿 Ba	seReloc.
+	· 5	s 🕇 🕹	4	•	•					
Na	me	Raw Addr.	Raw size	Virtual Addr.	Virtual Size	Characteristics	Ptr to Reloc	Num. of R	loc.	Num. of Linen
⊳	.text	1000	D000	1000	D000	6000020	0	0		0
⊳	.rdata	E000	1000	E000	1000	40000040	0	0		0
⊳	.data	F000	4000	F000	4000	C0000040	0	0		0
⊳	.reloc	13000	600	13000	600	42000040	0	0		0

# Figure(25):

Then change the image base: if it's different value of the OEP.

Disasm:	text G	General	DOS Hdr	Rich Hdr	File Hdr	Optional Hdr	Section Hdrs	BaseReloc.
Offset	Name			Value		Value		
D0	Magic			10B		NT32		
D2	Linker \	/er. (Majo	or)	С				
D3	Linker \	/er. (Mind	or)	0				
D4	Size of	Code		CE00			<u> </u>	
D8	Size of	Initialize	d Data	5000				
DC	Size of	Uninitial	ized Data	0				
E0	Entry Po	oint		C9A0				
E4	Base of	Code		1000				
E8	Base of	Data		E000		(2)		
EC	Image	Base		400000	-			
FO	Section	Alignme	ent	1000				
F4	File Alig	gnment		200				
F8	OS Ver.	(Major)		6		Windows Vista / S	erver 2008	
FA	OS Ver.	(Minor)		0				
FC	Image	Ver. (Maj	or)	0				
FE	Image	Ver. (Min	or)	0				
100	Subsyst	em Ver.	(Major)	6				
102	Subsyst	Subsystem Ver. Minor)						
104	Win32	Win32 Version Value						
108	Size of	Image		14000				
10C	Size of	Headers		400				

Figure(26):

values.

### Unmap the unpacked file

How we edit the section headers? ordered steps.

```
first: copy Virtuall address values into Raw address values.
second: Raw size Raw size of .test = Raw adress of .rdata - Raw adress of .text
E000 - 1000 = D000
Raw size of .rdata = Raw adress of .data - Raw adress of .rdata
F000 - E000 = 1000
Raw size of .data = Raw adress of .reloc - Raw adress of .data
13000 - F000 = 4000
Raw size of .reloc = still the same third: copy Raw size values into Virtual size
```

After changing save the file. This is our unpacked malware

# Article quote

على الضفةِ الأخرى لن نخشى الغرق

## Refernces

1- <u>https://www.mcafee.com/blogs/enterprise/malware-packers-use-tricks-avoid-analysis-detection/</u>

2-https://www.oreilly.com/library/view/learning-malware-analysis/9781788392501/12556df2-7825-4e43-8811-c0fabeab78d8.xhtml

3- https://www.0xbyte.com/unpacking-mzp-ransomware-manually/

4-

https://isc.sans.edu/forums/diary/Stackstrings+type+2/26192/#:~:text=This%20is%20a%20te chnique%20that,the%20allocated%20chunk%20of%20memory

5- https://msdn.microsoft.com/en-us/library/windows/desktop/aa366887(v=vs.85).aspx