[RE019] From A to X analyzing some real cases which used recent Emotet samples

some-real-cases-which-used-recent-emotet-samples/

13/01/2021

1. Introduction

Emotet (*also known as Heodo, Geodo*) is one of the most dangerous Trojan today. Through mass email spam campaigns, it targets mostly companies and organizations to steal sensitive information from victims. Recent records show that **Emotet** is often used as a downloader for other malware, and is an especially popular delivery mechanism for banking Trojans, such as *Qakbot* and *TrickBot*, and also lead to ransomware attacks using *Ryuk*.

<u>ANY.RUN's annualreport</u> pointed out that the most active malware in 2020 is **Emotet**.



Fig 1. Statistics of top threats by uploads for 2020

In this article, we analyze in detail full attack flow in some real cases of recent **Emotet** samples which were discovered and handled by us while providing cyber security services to our customer:

" Sample 1:

- Document template: <u>b836b13821f36bd9266f47838d3e853e</u>
- Loader binary: 442506cc577786006da7073c0240ff59

" Sample 2:

- Document template: 7dbd8ecfada1d39a81a58c9468b91039
- Loader binary: e87553aebac0bf74d165a87321c629be

Sample 3:

- Document template: d5ca36c0deca5d71c71ce330c72c76aa
- Loader binary: 825b74dfdb58b39a1aa9847ee6470979

2. Type of infection

The main distribution method of Emotet malware is malicious email campaigns, using infected attachments, as well as embedded URLs. These emails may appear to come from trusted sources (*cause the victim's email account was taken over*). This technique helps trick users into downloading the Trojan onto their machine. Some illustration image of emails spread Emotet:

	Te Trân Attachments	Wed 10/28/2020 11.33 AM		 hanindo@hanindo.net>
	Hoing		Cyber S	
From 'Phạm Subject thông tin liên lạc 05 tháng To Phạm	1 2021	@iliad.fr	>☆ ☆	
Ngày tốt,				
Cảm ơn sự giúp đỡ của ba	an. Vui lè	òng xem phá	ần đính kèm.	
Tệp zip: hình thức 05 thái	ng 1 202	21.zip		
Mật khẩu: VKPTV <mark>SBEX</mark>				
-	Cyber Se	curity Services		
✓ 🛛 🖞 1 attachment: hình thức 05 t	tháng 1 202	t 1.zip 75.7 KB		

Fig 2. Examples of malicious emails with attachment

3. Document template and VBA code

Emotet templates are constantly changing, the final target of attackers for leveraging templates to trick the victims into enabling macros to start the infection.

3.1. Sample 1

Document template:

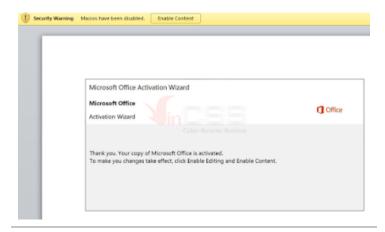


Fig 3. Sample 1's document template

This sample still acts in the usual way:

- Execute VBA code when opening document through **Sub Document_open()**.
- VBA code spawns powershell to execute encoded Base64 script.

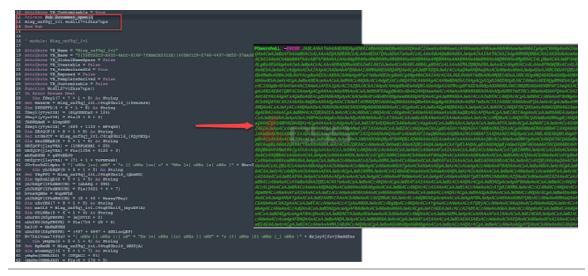


Fig 4. VBA code spawns powershell to execute script

The powershell script after decoding and deobfuscating usually look like the image below. It will download the payload which is an exe file to execute:



Fig 5. Powershell script downloads payload from the C2 list for execution

3.2. Sample 2

Document template:

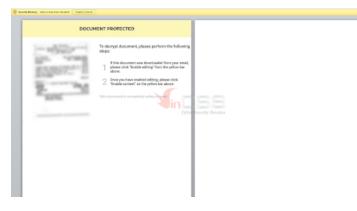


Fig 6. Sample 2's document template

This template also uses VBA, but there are some differences with **Sample 1** as follows:

- VBA code is executed after closing document through Sub Document_Close().
- Instead of using **powershell**, this sample spawns **certutil.exe** for decoding enncoded Base64 payload and then call **rundll32** for executing the decoded payload. The payload and related information are hidden in the document in white font.



Fig 7. VBA code uses certutil for decoding payload and calls rundll32 to load payload

Decode encoded base64 content will get **VideoDownload.dll**, this file has an exported function is **In**. This function is executed with the help of **rundll32.exe**.

Offset				• •		6					A.							Ancii		Offset				2				6		g		9	٨	8	e	ъ	R	P	Asaii
00000000	54 56	71.5	1.6	1 41	14	0.4		- 61	. 61	L 4.	1 (15	41	41	. 4		61	TVQGAABAAABAAAA		00000000	41	5	ь 9	0 0	0.0	13 1	80	00	00	04	0	0 0	0 0	0 2	7.1	rr.	00	00	M3
00000010	28 28	38.4	14	1 40	:6	74		- 61	1.01	L 4.	1 (11	41	-01	. 4		41	//BAALgAAAAAAAA		00000010	10	0	0 0	0 0	0 0	10 1	80	00		- 60	0	D 0	0 0	0.0	0.1	00	00	00	
00000020	51 41	41.4	14	1 41	14	14		- 61	1.41	L 4	1 (11	41	41	. 4		41	CALAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA		00000020	00	0	0 0	0 0	0 0	10 1	80	00	00	- 00	0	0 0	0 0		0 (00	00	00	
00000030		41.4	14	1 41	14	14		- 61	- 41	L 4	1 (11	61	41	14		41	*******		00000030	00	0	0 0	0 0	0 0	10 1	80	00	00	- 00	0	0 0	0 0		8.	01	00	00	
00000040	41 41	41.4	14	1 41	14	14		- 41	. 01	14	1 (11	61	-91	. 4		41	ALLAALAALAALAA		00000040	00	11	F 0.		8.0	10 1	54	89	CD	Z1	. 61	5 0	14		Ð	21	56	68	
00000050	43 41							75		, ,	4 (11	74	41	. 6		42	CARAAASTugSAttAnk		00000050	6	17	32	07	07	12	ഔ	67		61	6	DΞ		36	a i	68	62	æ	is.program.canno
00000060	49 62	67 4	25	4 40			10	- 56	- 41	16	0 '	10	63	79	1.4			Ibg07H0h9ShpcyDw	Base64	00000060	74	, 21	06	Z 6	8 2	10 '	72		68	20	6	96	8 2	04	4.1	67	53	20	t.be.run.in.DOG.
00000070	63 68	39.6	86	3 60	94	67		49	- 45	74	B (20	62	60			76	on ProsPtIGShbm5v	da anda	00000070	6	6	F 6	46	8 2		DID-	DD	ᄡ	- 24	DI	0 0	0 0		0 (00	00	00	mode
00000000	64 42	42.6	95.	A 52	24	27		- 64	1 31	, 3	۰.	7	61	57	12		67	dCR128BydiHgalHg	aecoae	00000080	21	4	E 2	8 A	26	ab 1	237	46	PD	68	21	F 4	63	а 6	10 I	8 8 -	66	20)H(.m/P.m/P.m/P.
00000090	52 45	39.5	44	P 41	73	17		54	- 45	15	5.	15	44	51	. a		48	REPETSIV29DubQ0R	_	00000090	D	F 80	38	79	06	67 1	28	46	P0	89	8	28	53	01		2 8	46	90	g/P/P.
00000040	48, 41							- 41	- 48	14	£ 1	10	54	69	1.6	9	68.	TAXAAAAAAAAAAA		000000A0			38							- 27	. 47	74	2.8	1.4	81	ar -	46	20	u/P.70C.H/P.
00000080	62 53	39.4	73	8 41	73	07		52	: 76	54	2 '	14	4c	30	1.6	2	77	h8P0000vRv8t50ke		00000080	34	• •	74	28	16	82	28	46	80	- 37	. 47	74	5 2	11.7	* 1	2 1 7	46	20	208.3/P.208/P.
000000000	32 62	48.3	a a	8 41	16	37		52	: 76	64	4 3	ы.	73	37	1.5	8	77	2b03BBowkvD2a72w		000000e0	64	5	70	5 8	06	88 (22	46	80	62	-21	r 4	7 2	00	9	817	46	20	4002/P.m/0/P.
000000000	47 69	39.4	73	8 41	56	07		- 74	50	14	8 (ы.	40	30	1.6	8	77	01908NuztFELLORM		000000000	2	•	64		11	80 1	۲r.	46	F0 -	r7	-	64	6 2	16	ia (27	46	F0	.ro.1/ere.1/e.
00000080	50 30	64.4	4.3	8 55	58	0 7	6	52	: 74	5.4	1 :	12	52	30	1		78	PO40800 VRVA/NOTA:		00000080	r.	1	68	9 7	0 6	6C (87	16	F0	61	- 21	r 0	1 1	10 6	a i	81	16	20	.F
000000100	59 69							52	6	5.6	61	38	60		I é	8	77	119080988.fr+50ke		000000#0	r		64	6 7	16	80 i	27	46	F0 -					86					.FD.1/F.Richm/F.
00000100	5A 46	66.5	63	8 41	16	77	6	52	: 76	5.6	2 ·	14	60	30	1.6	6	77	IF EVBOUVEVELOD EN		00000100	00	0	0 0	0 0	0 0	10 1	00	00	00	- 50		50	0 0	0 4	σI	01	05	00	······································

Fig 8. Decoded payload is a DLL

Disasm: .text	t General I	DOS Hdr	Rich Hdr	File Hdr	Optional Hdr	Section Hdrs	Exports	IIII Imports	Resources	BaseReloc.	Debug	LoadConfig
÷											-	
Offset	Name		Value	M	eaning							
16D10	Characteristics		0									
16D14	TimeDateStarr	P	FFFFFFFF	Su	nday, 07.02.2106	06:28:15 UTC						
16D18	MajorVersion		0									
16014	MinorVersion		0									
16D1C	Name		18542	Vic	leoDownload.dl							
16020	Base		1			-						
16D24	NumberOfFur	ctions	1									
16D28	NumberOfNar	mes	1									
16D2C	AddressOfFun	ctions	18538		_							
16030	AddressOfNan	nes	1853C									
16D34	AddressOfNan	neOrdinal	s 18540									
Exported Fur	ctions [1 entry	1										
Offset	Ordinal	Func	tion RVA	Name RV	A Name	For	warder					
16038	1	78F0		18554	In							

Fig 9. The expored function of DLL

There is an embedded PE file in resource section of the above dll. The resource data is encoded.

Analysis [Resource Directory]																		
Resources Tree	Offset	0	1	2	3	4	5	6	7		9	A	в	с	D	E	F	Ascii
🔻 🛅 RCData	0000000	0 0	6E	90	00	03	00	SC.	00	- 04	00	19	00	91	FF	E1	00	≣n\
101 - [lang:1033]	00000010	38	00	FF		00	00					00			00	00	00	· · · · · · · · · · · · · · · · · · ·
Configuration files	0000020	00	00	00		00	00				00	00			00	00	00	
	00000030	00	00	00		00	00	00	00	00	00	00	00	80	00	00	00	
	00000040	0E	17	Вλ	ΘE	60	В4	09	CD		88	01	40	CD	21	54	68	!L.!Th
	0000050	69		20				67				20						is.program.canno
	00000060	74	20	62		20			6E	20	69	6E	20	-44	47		20	t.be.run.in.DOS.
	0000070	61	67	64	65	22	ΟD	OD	40	24	00	00	00	00	00	00	00	mode\$
	00000080	BE	AD	10	77	FA	cc	72	AC	78	cc		¥C.	73	cc		AC	EEE.
	00000090	79	DQ		AC	FB	сc		AC		D3		¥C.	17			AC	9. r(r.
	00000010	13	03	77	AC	FB	cc.	72	AC.		69		68	78	cc		AC	r.Richr.
	0000080	00	00														00	PEL
	00000000	DB	92	8E	57	00	00	00	00	00	00	00	00	Ξ0	00	OF	01	
	00000000	08		06		00	90		00		A0				00	00	00	
	00000080	64	20	00	00	00	10	00	00	00	λ 0	02	00	00	00	40	00	d
	00000000	00								04		00			00	00	00	
	00000100		00	00	00	00	00	00	00	00	60	05	00	00	10	00	00	
	00000110										00							1
	00000120	00	00	10	00	00	10	00	00	00	00	00	00	10	00	00	00	

Fig 10. DLL has a PE file that has been encoded

The **dll**'s code when executed will load the content of a porn site, then retrieve the link of the **.mp4** file (*which is a hot keyword-related leaked sex clip of Vietnamese figure*). It read bytes from mp4, through the loop, by using the read bytes as **xor_key** for decoding the above resource to get the complete PE file. Then it saves the decoded file to **%temp%/tmp_e473b4.exe** and execute this payload.

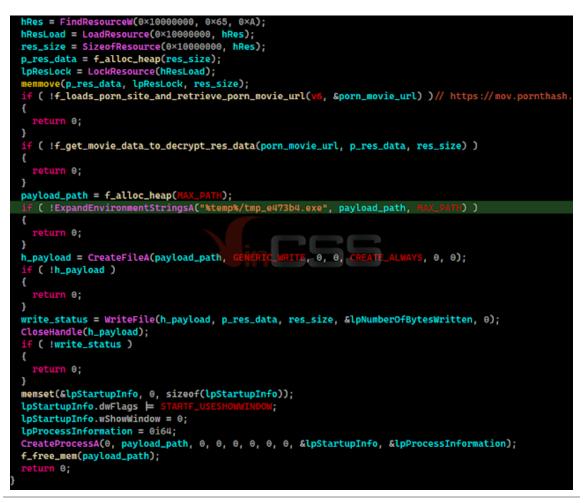


Fig 11. Pseudocode performs decoding resource data and spawns new process

3.3. Sample 3

Document Template:

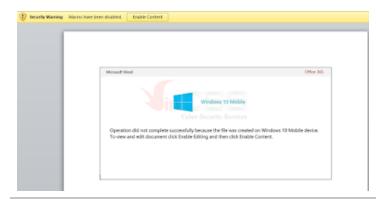


Fig 12. Sample 3's document template

Same as Sample 1:

- Execute VBA code when opening document through **Sub Document_open()**.
- VBA code also spawns powershell to execute encoded Base64 script.

7 Attribute VS_FrederlaredId = True 8 Attribute VS_Exposed = True	
9 Avtribute VS TexplateDerived - True	
10 Attribute VS Customizable = True	
11 Private 3ub Document open()	
12 Ko22se9520popper?	
12 Addrewstropyger/	
	Dowershell -# hiddon -ENODO cn8FAFQALQ80AFQARQBINACAALAB2AGEAcq8JAEEAYg85AGUADg8trADYA5QAgACAANABQACAANABUAHKAUAB1AFBANQ85A
14	ИАЕВАКУВЕНССАК. АЛЛАНЧАн (BEZAHQA ZQA HACHA, IMBKACCAL. AMIA GKAC gBFAGNA, IMASAC CAT QAWA GKAbud WACCAH QAQAACKA I AAQAAHAASAA GAQAAHAASAA JAQQA I QAN
16 ' module: Filóyarakt wabubr	ВРАНЬАНиВРАНЬАЛДВРАНЬАЛЛВРАНЬАЛДВРАНЬАЛИВРАНЬАЛДВРАНЬАЛДВРАНЬАЛДВРАНЬАЛАВРАСТАТАЛЬАЕХ 2018 ГАГСАЛАВАССАГАДДАССАГАЛАНИМИДВТАССАГАЛ
17	HBFACEALAANAFQAZQBMCHATgANACHAJHBUACMAHBAFZAGgACHACHAYQAGAAQHAGACQARQByAHZAbHBYAEZAYHBBAGMAbHBUAFAACgBLAGYAZQBYAGMAbgB JAGUALA
18 ACCUSATE VS Dame - "Flidyarnat aubuje"	zpвилнpAbanačuA3н85AEMÜini+AceAbiduAнpAupitAckAkiuAnAGIA3iiA+AceAdg81AceAkipi7Acpapit6ADEAGAA1AKiANppi9AcpAQiabAciAiiAbAciMui
19	AŠAALADIANGASACGAJUBJACCANUMOACCANMANAČSAJUAZAEGAJUADACNADUNGACGAJAAGAECAŠQAgAČAÄVGBBAETASQBBAGTATABEADoAuuAZAGNÄNĢŠUAHYAVQBSAHUN.
20	MAYABUAESAUgBZACTAKAARAEgATWBMAEUAZAArACAAKAAAACgAJWBJAFAANQAUACsAJWBEACcAKQAUACsAJWBEACcAKQAUACsAJWBIADKAJWArACgAJWBLADKAJWArACcAAQBJAFA
21	TEA JUATAC CAANB JACCANDATACCANDATACCANDAUACTACQBQAEJACABBAEEAYABDAEWATgAUACQAWNBDAGgAYOBSAFAADAAACSAWNBDAGgAYOBSAFAADAAAACSAWNBDAGG
22 ' module: Eazé hižižoémég7e	AGBAYOBSAFBAOQAYACKARDADAD <mark>BAJABC</mark> ADUAI H DXXAD A KAADACCAABAKAAAACSAJWAJADZAJWAJACXAJWB7ACCARDA7ACAAZAADAECARDBUNCBASDBASURTDADACAAKAAAA
23	SADDARDARACCANDARA CUADORACOADDO A CONDUCTOR DA LANDARDO DA FUNYADDACKAVADZANAA CODORADALDACDA VHORAGHA TOADADAZAA DA COADHUN COANHANAGHA CHAN
24 Attribute VS Name - "Eard h1212odmig7a"	лил селини полизили наскионила и така и бака так всялина на селопина селини на наска интерсении таком чели на насна ина и селини
28 Function Rozzsesszopogess7()	gBaABBAPARAEgATuBINEUARuinaACgAKAAnAEBAJuArACCANgAUAECANgAUAECANuuACKANuuACKAJUATACgAJuASAHUAOgAnACSAJuAgACcANuuAAEBANgAnACSAJUAUAEYAZg
16 On Error Resume Next	The Flore And Anna Anna Anna Anna Anna Anna Anna
27 YekizuWazp0al7ybz = "Tophivifasio" + "Ciscual	area a loor#
25 pf4 = Regeldislesilowty + Udirf42304_w6el2p.5	
19 GoTo HithdFit	
30 Dim KVuld As Object	
31 Set dibED = FWOfsD	
	ann (300)ann (303055) ann (300 - Fillann (3050-6) ann (3000) ann (3000) ann (371)
55 Dim Edwd75 As Object	
14 Set RdwdFR = kVuLJ.CreateTextFile("adov@EgR(\y	ANTIFICATION NOTICE NOTICE NOTICE TO A STREAM ST
35 EdwdFB.WriteLine "ALenfOWTEnb"	
16 RowdFB.WriteLine "BJLVYtDgyYtJIRBS"	
37 MdwdFB.WriteLine "XDvjsENohoWI"	
10 Set kUbvyLD = 9JyeIHD	

Fig13. VBA code spawns powershell to execute script

The powershell script after decoding and deobfuscating will also performs the task of downloading the payload to execute:



Fig 14. Powershell script downloads payload from the C2 list for execution

Differ from **Sample 1** (*use powershell to download loader is an exe file*) and **Sample 2** (*decode DLL and use this DLL to decrypt the loader as an exe file*), in this **Sample 3**, the downloaded payload is a DLL file, exports **Control_RunDLL** function. Script uses **rundll32** to execute this payload. So that, the downloaded payload is considered as a DLL loader.

4. Loader payload

4.1. Execution flow of loaders

The payloads of **Sample 1** and **2** (PDB path information: **eeeggggggggseb.pdb**) were built with *Visual Basic*:

	Eie : test.exe	Sample 1		Eie: tmp_e473b4.exe Sample 2	,₽ ш
	Entry Point : 000021E4	00 C EP Section : .text		Entry Point : 00002064 00 < EP Section : .text	3
a	Fie Offset : 000021E4	First Bytes : 68, F0, 22, 40, 00.	m	Fie Offset : 00002064 First Bytes : 68.A8.B6.40.00	0
	Linker Info: 6.00	SubSystem : Windows GUI	E	Linker Info : 6.00 SubSystem : Windows GUI	PE
o utovet	File Size : 00008E000h	< N Overlay : NO 00000000	Info	Fie Size : 0005402Eh S Overlay : 0000002E	0
	Image is 32bit executable	RES/OVL: 23 / 0 % 2020	X	Image is 32bit executable RES/OVL : 48 / 0 % 2020	Alla
-1	MS Visual Basic 5.0-6.0 EXE			MS Visual Basic 5.0-6.0 EXE	Scan / t
	Larner Info - Help Hint - Unpa	ick info 0 ms.		Larner Info - Help Hint - Unpack info 0 ms.	
	Not packed , try www.olydby	g.de or IDA www.hex-rays.com or x64		Not packed, try www.olydbg.de or IDA www.hex-rays.com or x64	0 2

Fig 15. Loaders of Sample 1 and 2 were built with Visual Basic

Sample 3 was built with Visual C++ (PDB path information: E:WindowsSDK7-Samples-

masterWindowsSDK7-Samples-

masterwinuishellappshellintegrationRecipePropertyHandlerWin32ReleaseRecipePropertyHandler.pdb)



Fig 16. Loader of Sample 3 was built with Visual C++

When first infected, the **Emotet** payload runs through two stages. During the first stage, it checks the victim system, if it's running with high privilege, it drops binary to **CSIDL_SYSTEMX86**, otherwise to **CSIDL_LOCAL_APPDATA**. Finally, it launches the second instance. Payload running at the second stage will communicate with C&C servers that embedded in its binary.

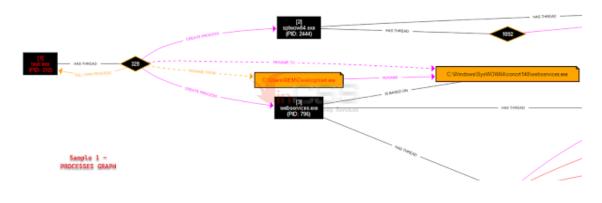


Fig 17. Sample 1 execution flow



Fig 18. Sample 2 execution flow

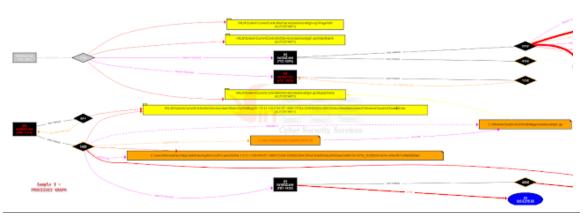


Fig 19. Sample 3 execution flow

4.2. Technical analysis of the loader

4.2.1. Sample 1 and 2

These loaders when executed will allocate and unpack the main payload to the allocated memory and execute this payload:

AT AND UT USED 8	E8 78880890	call	0x1F06550							-
01F05C05	A1 C4E2F001	nov	eax, dword ptr dst (Address	Size	Info	Content	Type	Protection	[Initia]
01F05C0A	8560	test	eax, eax	090392080	20024020	"-nanc"	Resources	THE	-R	ERVC-
0 01F05C0C	75 18	ine	0x1F05CF9	603A2080	00001000	".relec"	Base relocations	THE	-R	ERVC-
0 01F05C0E	B9 3F09CER2	mow.	eck, BrA2CE 093F	00380080	00002000			MAP	-RM	-89
0 01F05CE0	CO OBC2FFFF	ca11	0×1F93F70	000300000	00039000	Reserved		PRU		-89
01F05CE8	BA C8708189	nov	edx, 8x898170C8	003550000	00007000			PRU	-RM-G	-89
01F0SCED	8968	nov	eck eak	00400000	80081890	test.exe		THE	-g	ERVC-
01F0SCEF	E8 DCE1FFFF	call	0x1F03ED0	89481989	00063000	"_text"	Executable code	THE	ER	ERVC-
01F05CF4	A3 C4E2F001	nov	dword ptr dss 0x1F0E	8946C089	20024020	".data"	Initialized data	INC	-RW	ERVC-
01F85CF9	68 80	push	8x 8	00470000	00021000	"ursne"	Resources	ING	-R	ERVC-
e1FeSCFD	FFDB	call	eax	094A 3020	00001000			PRU	-84	-89
e1FeSCFD	C2 1989	er er til	8x18	004A 1000	0007F000	Reserved (004A0000)		PRU		-89
01F05000	51		ecx	0052 000	60055990			Hist		-R
01F05D01	C70424 84210000	nov	dword ptr ss:[esp]	8823 1080	00026000			P.80	-RV	-89
e nicronna	R16575 P3548888		duned alter confirmal	0055 000	00010000	Reserved (00530000)		PRF		-89
				0057 1000	000EE000			PRU	-RM	-89
01F06550				0065 000	00012000	Reserved (00570000)	Sample 1 -	PRU		-R¥
				0047 000	60095.660			HIRP	-R	-R
				8967 020	20171220	Reserved (80570880)	unpack main	HIRP		-R
01F05C0-0				0071 1080	00035000	And the second second second	payload	HAP	- R	-8
21 Dunp 1 21 Dunp 2	Dump 3 🔛 Dump 4 🔮	E Oumo - 5	Watch 1 e- Localf	0076 080	00005000	Reserved (00570000)	paycoad	PRU	-84	-84
	a control a control a			0081 080	00002000				-8	-89
Address Hex			ASCII	0002 000	20032000			1607	-84	-8
01F00000 40 50 90 00 00			e ee 📴	0052 000	20024220			P.89	-89	-89
01F00010 88 00 00 00 00	2 eo eo eo 140 oo oo oo	00 00 0	0 00	8051 020	00020000	Reserved (80830880)		2.00	-89	-89
		00 00 0	0 00	0052 020	00181000	merced (serves)		HINP	-8	-8
		CO 00 0	0 00	000 000	00157000			HINP	-8	-8
			4 68º'.It.LitTh		01209000	Reserved (00900000)		HIGP	-8	-8
01F00050 69 73 20 70 72		61 6E 6	E 6F is program canno	0002 000 0100 020	8080F880	meserved (40400000)		MAP	-8	-R
01F00068 74 28 42 45 20			1 20 t be run in DOS	0121 000	202022800	Reserved		1.00		-84
	80 80 86 24 88 88 88			01E 980	00007000	Reserved		280	-RW-G	-89
01F00080 68 DF FB DE 2F			5 80 k060/k/k/k	A100 000	00001000			DE	COM-	COL
			5 80 "J.J., N., BSp., N.,	THEOLOGY	00010000			PRU	-RN	ERW
01F00000 52 C7 48 80 2E			5 80 RCK	01210000					-89	-89
		00 00 0		01F11080	20021000	Reserved (01F10000)		1.50		-89
01F000C0 50 45 00 00 40				01/50000	20022890	active (all local)		PAU	-89	-89
01F00000 00 00 00 00 E0		00 65 0	0 00	01F52080	00002000	Reserved (01F50000)		PRU		-89
01F000E0 00 22 00 00 00				01F60080	00010000	an art out (err suesu)		PRU	-84	-84
	<u>0 00 40 00</u> 00 10 00 00			01F70080	003F0000	Reserved (@1F60000)		PRU		-89
01F00100 05 08 08 08 08	<u>a sa sa salat ao ao ao</u>	00 00 0	e oe	10000	and an allow	ac served (all solutio)		11/0		

Fig 20. Sample 1's loader unpacks the main payload

COLO BUSCISCES	C0 FD002020 A1 88085500	call.	exsonate execution	Memory N	Asp					- 0
0 005050E8	8508	test	PAX PAX		Size	Info	Contest	Tune	Protection	Initial
0 005050EC	¥ 75 18	ine	0x505009	00103-000	00052000	Reserved (02100200)	A MARKAUN.	1997	Trotectur	- Ream
eesesee	B7 12000078	800	ecx, exys	R 001E0000	00001000	articles (activities)		PRU	-89	- 69
005050F3	E8 08E2FFFF	call	0x503E00	09178080	00000000			P 80	ER	
e65656F8	BA 8407F389	101	edx, exer	001FA080	00006-000	Reserved (001F0000)		PRU	-	- 89
e65656FD	8808	100	ecx eax	00208080	00170202	Reserved		12 10 10		
005656FF	E8 SCE1FFFF	ea11	ex50366 @	00370080	00006-202	PEB		PRU	-89	- RW
e 80505084 +e 80505089	A3 88005C80 68 80	a ush	dword ptr	00382080	ABA7F BAB	Reserved (ARTRARA)		1990		
0 005050 00	FFDB	1011	PAK .	10102020	00001000	tmp_e473b4.exe		EHG	-8	ERVC-
0 005050 00	C2 1000	of each 1	0(10	00401080	00029-000	".text"	Executable code	THE	ER	ERVC-
0 00555010	51		ecx	00A2A020	00052000	".data"	Initialized data	ING	-84	ERVC-
0 00505011	C70424 00920000	100	dword ptr	00420 000	00029-000	".rsro"	Resources	THE	-8	ERVC-
 B6565018 	B8 398E2336		eax, ex35		000051000	\Device\HarddiskUplume2\Him	6N	1000	-84	
0 00505010	880024		ecx. duor	0053 000	00081000			P 80	-81	-RW
0 00505020	F7E1	mark a	OCK.		00001000		C	P 100	-89	- 89
e anc/cn >2	DP BDA7784E		ALC: NO. OF STREET, ST	0051 000	96021202	Reserved (08540808)	Sample 2 -	PRU	-8	- 89
 C 				0051 010	00004000		unpack main	100	-8	-B
6568				0057 020	00055000	\Device\HarddiskUelume2\Win		100		-8
				0.755 0.20	00001000	(seates (sarbers wateres (stars	payload	100	-84	- RW
				0.05 000	08081808	e	puycouu	P 80	-84	-89
55650				0051 000	00022000			P 89	ERM	ERW
Durio 1 💭 Durio 2 👫 Durio 3	🔁 Dune 4 👔 Dune 5 💆 Welch 1		2 Stat	0051 000	08001400	Concernation of the second sec		PRU	-89	-89
	esci		/ Seuce	NOTE DAD	00002200	Received (195010340)		0.011		a Distant
ress Hex second An so on an as an a	12 04 08 08 08 FF FF 08 08 H2		Co. I I	0 00500000	00010000			PRU	-89	ERV
				00502000	000022000	Reserved (00500000)		PRU		-89
2020 00 00 00 00 00 00 00 00				00528080	00057000	neserves (ouseoute)		1100	-89	
Ce030 00 00 00 00 00 00 00 0				00585080	00091000	Reserved (00620000)		PRU		-89
	D 21 88 81 46 60 21 54 68	it t	ÍTTB -	0 00728080	00070-000	Reserved		100		- 69
	2 61 60 20 63 61 6E 6E 6F 15 p	rograń c	anne	00010020	00022000	Thread 8F0 Stack		PRU	-84-6	- RW
20060 74 20 62 65 20 72 75 6	E 20 69 6E 20 44 4F 53 20 t be			00828080	000F0-000	Reserved		P 800		-89
20070 60 6F 64 65 2E 00 00 0		\$		00910020	00022000	Thread 100% Stack		P-938	-RM-G	- 89
20080 68 DF FB DE 2F BE 95 8	0 2F BE 95 80 2F BE 95 80 k00P	PiPi	/h	00924080	000F8408	Reserved		PRU		- 89
	0 52 67 78 80 86 86 95 80 "LJ.	-¥RCp.	-No.	BBA10 886	00003000	Thread FED Stack		P 899	-89-6	- 89
C80A0 52 C7 48 80 2E BE 95 8		A. Rich	AL.	00020000	00007000			HMP	-8	-R
2000 00 00 00 00 00 00 00 00 00 00 00 00				00427020	00012/009	Reserved (08020000)		1007		- 8
20000 50 45 00 00 <u>40 01 04 0</u>	28 31 88 5F 00 00 00 00 PE			00820000	00101000			1995	-8	-R
Ceaba en na an en ca en ez o		à		00032020	00050808			1966	-8	-R
COCED 00 20 00 00 00 00 00 0	10 E0 50 00 00 00 10 00 00	····· <i>y/</i> ···		00088090	00681000	Reserved (00C30000)		1992		-R

Fig 21. Sample 2's loader unpacks the main payload

These main payloads are quite small in size and were built with Visual C++:

				tie: s	ample_2_main_pa				РШ	
	Entry Point : 00005CD0 00	EP Section : .text		Entry Point	00005CE0	00 <	EP Section :	.text		•••
e	File Offset: 000050D0	First Bytes : E8,7	8,08,00,00	File Offset :	000050E0		First Bytes :	E8,FB,08,00,00	0	Plug
8	Linker Info : 12.00	SubSystem : Wind	aws GUI		12.00		SubSystem :	Windows GUI	PE	
Excinto	File Size : 0000C000h	Overlay : NO	0000000		0000C200h	< N	Overlay :	NO 0000000	0	2
No.	Image is 32bit executable	RES/OVL : 0 / 0 %	2020	Image is 3.	2bit executable		RES/OVL:0	/0% 2020	A I	=
	Sign. 2268 : [Microsoft Visual C++ 8]				: [Microsoft Visual				Scan / t	Rip
	I anno Tafa - Hala Hint - Hannah infa		HashMyFiles	Cyber Sec	curity Service	5		-		- 6
TO	Inpacked_payload		File Edit View	Options Help						
			🌒 🕾 🗀 🕲 😂	🖵 😫 🖻 😭	🖏 📲					
Organize	e Include in library Share with	New folder	Filename /		MD5		SH/	u.		CRC32
🚖 Favo	arites Name	Size	Sample_1_main	_payload.exe	5401212667 <i>a</i> 6c7d8686	\$736503e41	1 734	07c353055bd4692aaa68	3117fe727825b4fe30	455be79
De De	esktop 📧 sample_1_main_payload	f.exe 48 KB	Sample_2_main	_payload.exe	746b4112b84d663efac	e3092c9a72c	89 664	8aceeb1055e6cab16d1	6<7885933649464134	: b5ea694
	ownloads	d.exe 49 KB								

Fig 22. The main payload of Sample 1 and 2

4.2.2. Sample 3

This sample, when executed, will get the address of two undocumented functions LdrFindResource_U and LdrAccessResource from ntdll.dll. These functions are used to access resource data embedded in the loader:



Fig 23. Sample 3's loader accesses resource data

Next, it computes the **MD5 hash** of the pre-initialized data and generates an **RC4 key** based on the computed hash. Then, use this **RC4 key** to decrypt the above resource data and execute the main payload:

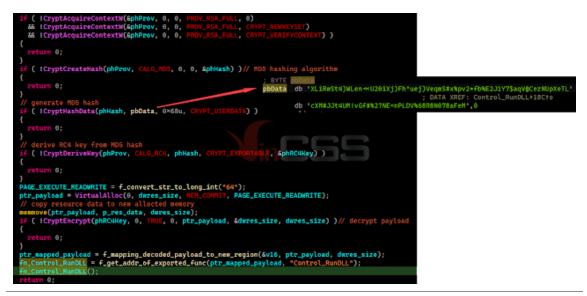


Fig 24. Pseudocode performs decoding and executing the main payload

The main payload is another DLL and also has an exported function is **Control_RunDLL**:

Ele : main_payload_dl.bi	1		Functions	counter : 0002 decimal
Entry Point : 0000CB91	00 S EP Section : .t	ext 500	1 - X.dl 0001	Control_RunDLL .
File Offset : 00008F91	First Bytes : Fl	F.4C.24.08.75 000	2 - X.dl 0002	RunDLL
Linker Info: 12.00	SubSystem : W	/indows GUI end.		
File Size : 0001D600h DLL 32 bt- Library image I xx - Std Compiler section , M	RES/OVL : 0 / 0 15 C++ Visual Studio 2013	3 [Win Vista]	Cip Ch	arset
Larner Info - Help Hint - Unpar Cick - [Scan / t] Button or		0 ms, http://otiofa.h		
		http://numo.a		
Filename / MD5		SHAI		CRC32
main payload dll.bin 3464d1af7	0e8H74525e87c6cebd0e6d	#6df5ab3d5a064ecceR	1e1e0ae39821f34	6ff53a 29ee5d4a

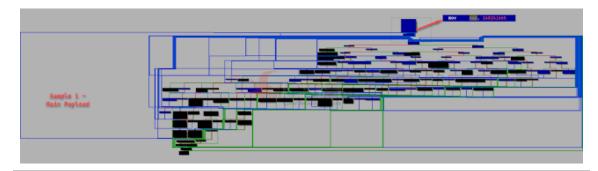
Fig 25. The main payload of Sample 3

5. Some techniques used in the main payload

5.1. Control Flow Flattening

A program's control flow is a path created out of the instructions that can be executed by the program. Disassemblers, like IDA, Ghidra, visualize control flow as a graph by creating a series of connected blocks (called "basic blocks"). In order to make reverse engineering more difficult, thwart the analysis and avoid detection, the main payload of **Emotet** usuallu apply an obfuscation technique is **Control-flow flattening**.

Basically, this is a technique used to break the flow of a program's execution by flattening it. When the control flow is flattened, the program is divided into blocks, all of which are at the same level. Therefore, it will be difficult to determine the execution order of the program at the first glance. After divided into blocks, there is a control variable to determine which basic block should be executed. Its initial value is assigned before the loop. At each block, will update the value of the control variable to redirect the program flow to another branch.



Below is the illustration for the main function of each above payload:

Fig 26. The main function of the main payload of Sample 1

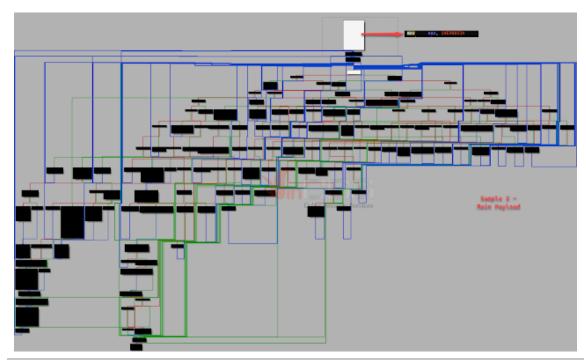


Fig 27. The main function of the main payload of Sample 2

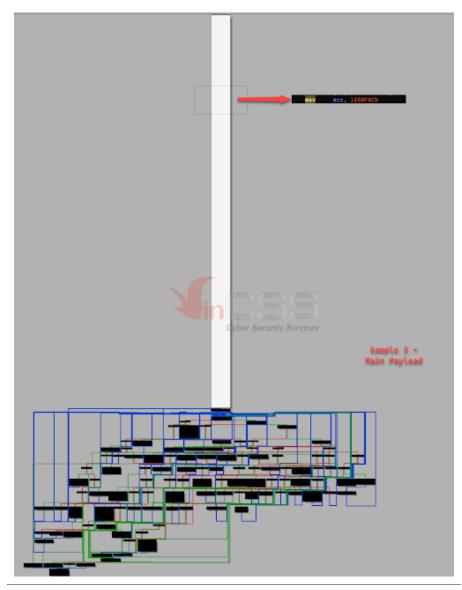


Fig 28. The main function of the main payload of Sample 3

In order to deobfuscate this technique takes a lot of time and effort to do, so my personal experience as follows:

- Try using <u>HexRaysDeob</u> plugin that was developed by <u>RolfRolles</u>.
- Perform static analysis using IDA, trying to guess the purpose of the functions, and name them.
- Perform debug and synchronize function names, variables that set in IDA with debugger with the help of <u>Labeless</u>plugin. During debugging, note the order in which the functions are executed and make a comment back to IDA.

5.2. Dynamic modules resolve

All payloads will rely on a pre-computed hash by the names of the DLLs to retrieve the base address of these DLLs when it needs to be used. In **Sample 1** and **2**, these hashes are passed directly to a function responsible for obtaining the base address of the DLL (**f_resolve_modules_from_hash**):



Fig 29. Sampe 1 and 2 call f_resolve_modules_from_hash

Particularly in **Sample 3**, there is a little bit of change, hash values are pre-computed according to the name of the DLL and the API function passed to the same function (**f_get_api_funcs**). Within this function, it uses these hash values to retrieve the base address of the DLL:

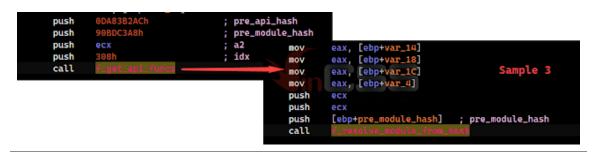


Fig 30. Sample 3 call f_resolve_modules_from_hash

The search algorithm in all three payloads is similar, only difference in the xored value:

<pre>if (uc = 'A' & uc = '2')</pre>	<pre>) + uc = calced_hash; if (uc ≥ 'A' is uc ≤ '2') (</pre>
<pre>) sample_1_nain_payload</pre>	break;) v2 = v2+InLoadOrderLinks.FLink; if (v2 = v1)
<pre>for (i = vi+InLoadOrderLinks.Flink; ; i = i+InLoadOrderLinks.Flink)</pre>	(
if (i = v1)	retern 0; }
(return 0;) return v2→011Base; sample_2_main_payload
))) if ((f_calc_hash_w(i→BaseDllName.Buffer, 0×C02) / 0×IFC32508) = pre_module_hash)	
break;	
<pre>> sample_3_main_payload roturn i+DllBase;</pre>	

Fig 31. Pseudocode performs looking up the hashes of the DLL name

Rewrite the hash function, combined with IDAPython to get a list of DLLs that **Emotet** uses:



The list of major DLLs that Emotet uses:

- [+] userenv.dll
- [+] wininet.dll
- [+] urlmon.dll
- [+] shlwapi.dll
- [+] shell32.dll
- [+] advapi32.dll
- [+] crypt32.dll
- [+] wtsapi32.dll
- [+] kernel32.dll
- [+] ntdll.dll



Fig 33. List of major DLLs that Emotet uses

5.3. Dynamic APIs resolve

In all three payloads, when need to use which API function **Emotet** will search and call that function. Based on the base address of the given **DLL**, payloads resolve APIs by looking up the pre-computed hash. In **Sample 1** and **2**, , these hashes are passed directly to a function responsible for obtaining API address (**f_resolve_apis_from_hash**):



Fig 34. Sampe 1 and 2 call f_resolve_apis_from_hash

In **Sample 3**, as mentioned above, hash values are passed to the same function (**f_get_api_funcs**). Within this function calls to function (**f_resolve_apis_from_hash**) to retrieve the address of the API:



Fig 35. Sample 3 call f_resolve_apis_from_hash

The search algorithm in all three payloads is similar, only difference in the xored value:



Fig 36. Pseudocode performs looking up the hashes of the API name

Rewrite the hash function that payload uses, combined with IDAPython to retrieve all APIs and annotate to related code. The list of APIs used in these payloads are similar and similar to the other variants. The final result is as follows:

	hash_val api_name api_name for i in api_ hash # xored	<pre>i_hash(api_name): .ue = 0x0 .list = [] .list = list(api_name_l .range(len(api_name_l .name_per_byte = ord(a .value = ((hash_value .value need to change (hash_value ^ 0x5A80E)</pre>	<pre>ist)): pi_name_list[i]) << 0x10) & 0xFFFFFFFF) For each payload</pre>	+ ((has	h_value << 0x6) & 0xFFFFFFF) + api_name_per_byte - hash_value
	toc_605CF9: eax, [abp+var_8] eax, [abp+var_6]	movr OCX, 43X ; mod Coll f_restive.apis_frem.ht mov g_func_werel32_ExitPr push 0 ; cool coll ess ; g_func_hermel32_	Torocass ; pre_api_hash le_base sh occess, taxanin_payl XMSF: mectiontext+Ctj		ecx, mininet.dll,hash ; pre_medule_hash f_resolve_medules_frue_hash
push push push call	eax, [obp+var_10]	dLibraryW ; pre_api_hash ; pre_module_hash ; a2 ; idx ; func_kernel32_LoudLib	VinCS	mov call mov	edx, func_wininet_InternetOpenW ; pre_api_hash eCx, eax ; medule_base f_rosolve_apis_from_hash ; func_wininet_InternetOpenW g_func_wininet_InternetOpenW, eax sample_2_main_paylead
add push call	esp, 14h esi eax	; lpLibfileHame ; g_func_kwrmel32_Loadt	loc_402C78:	push push push push call	<pre>; COOE XREF; sub_U02BE0+70+j 0 ; dwFlags 0 ; lpszProxy0ypgass 0 ; lpszProxy 0 ; lpszProxy 0 ; lpszProxy 0 ; dwaccessType Eesp+UDb+lpszAgent] ; legaent exx; g_fusc_wininet_InternetOpenW</pre>

Fig 37. The final result when using IDAPython to annotate related code

5.4. Decrypt strings

All strings are encrypted and only decrypt at runtime. The structure of the encrypted data is shown as below. The decryption algorithm of the payloads is the same:

		iov	ecx, offset dword_u00200 ; encStr
	9	all	f_decrypt_string; POST
٢		iov iov inp	ebx, eax [esp+38h+saved_verb], ebx addr db e xor_key short loc_462F86 db e db e
			; enc_data dword_u <u>00200</u> xored_length
Ц.	2:	10V	; CODE XREF: f_do_PO: dword_40D200 dd 51924690h ebx, [esp+38h+saved_verb] dd 51924694h dd 5510950h
			db 46h; F dec_data = enc_data ^ xor_key
Ļ	t	iov est	; CODE XREF: f_do_PO: eax, g_func_wininet_HttpDpenRequestW db a9h; f eax, eax db a9h; f db a9h;
	j	nz	short loc_402FAA db 40h : 0
•		ov all	ecx, wininet.dll_hash; pre_module_h; db 086h; 1 f_resolve_modules_from_hash db 23h; #
•		IOV	edx, 2275881483 ; pre_api_hash
		IOV	ecx, eax ; module_base
	c	all	f_resolve_apis_from_hash
•		IOV	g_func_winimet_HttpOpenRequestW, eax

Fig 38. The payloads call the string decryption function

Based on the above information, can use IDApython to create a script to decrypt data as follows:



Fig 39. Python code is used for decrypting data

The list of strings obtained in payloads is quite similar:

📴 reds in CalcopyCaloig	Sa web to adu/454C8	I with the f decourt mines	
$ \begin{array}{l} \hline \label{eq:second} \hline \\ \hline \end{rmatrix} & \left[\end{rmatrix} & rmat$	Densitian Spy Address Text	$ \begin{array}{c c c c c c c c c c c c c c c c c c c $	Stars "MAN, Control, Northit. a Na Sample 3 main payload dentities Marcant Printers Committee of Nav.
Bits Figure of Langering Victore Sample Langeling Langeling Langeling Langeling Langeling Langeling <thlangeling< th=""> <thlangeling< th=""> L</thlangeling<></thlangeling<>	 The p shall be and shall be a straight the straight of the straight th	2 Da. p. Control and Section 201 of Computing Stream Dis. p. Control and Section 201 of Computing Stream Dis. p. Control and Section 201 Occupating Window Dis. p. Control and Section 201 Occupating Window Dis. p. Control and Section 201 Occupating Window	Default 19 an "Nel / Nel / Control, Burdhill, 19 Microsoft Windowski Connert Version / Run.

Fig 40. List of strings obtained after using the script

5.5. List of C2 (IP & Port)

A list of C2 IP addresses and ports of **Emotet** payloads is stored in .data section as 8-byte blocks:



Fig 41. List of C2s is stored in each payload

Through script can quickly retrieve the entire list of this C2:

_		610 · · ·	_					
1	179.15.102.2:80	12		177.130.51.198:80	8		125.0.215.60:80	
2	91.121.200.35:8080	18		91.121.87.90:8080			163.53.204.180:443	
3	159.203.16.11:8080	18-1		104.131.144.215:8080			89.163.210.141:8080	
64	188.226.165.170:8080			188.226.165.170:8080			203.157.152.9:7080	
5	5.2.164.75:80	E I		2.58.16.86:8080	NE I		157.245.145.87:443	
6	54.38.143.245:8080			79.133.6.236:8080			82.78.179.117:443	
7	200.243.153.66:80	19 I		125.200.20.233:80	SIF .		85.247.144.202:80	
8	2.58.16.86:8080			109.206.139.119:80			37.46.129.215:8080	
9	185.142.236.163:443			188.40.170.197:80	lite -		110.37.224.243:80	
10	203.56.191.129:8080	RE I	10	121.117.147.153:443	B.	10	192.210.217.94:8080	
11	109.13.179.195:80	105	11	221.147.142.214:80	19.	11	2.82.75.215:80	
12	46.32.229.152:8080		12	88.247.58.26:80		12	69.159.11.38:443	
13	192.210.217.94:8080		13	37.205.9.252:7080		13	188.166.220.180:7080	
14	190.85.46.52:7080		14	213.165.178.214:80		14	103.93.220.182:80	
15	36.91.44.183:80		15	27.83.209.210:443		15	198.20.228.9:8080	
16	213.165.178.214:80		16	24.231.51.190:80		16	91.75.75.46:80	
17	103.80.51.61:8080		17	192.210.217.94:8880		17	88.247.30.64:80	
18	126.126.139.26:443		18	123.216.134.52:80		18	189.211.214.19:443	
19	91.75.75.46:80		19	179.5.118.12:80		19	203.160.167.243:80	
20	95.76.142.243:80		20	103.80.51.61:8080		20	178.33.167.120:8080	
21	181.59.59.54:80		21	172.96.190.154:8880		21	178.254.36.182:8080	
22	190.192.39.136:80		22	223.17.215.76:80		22	70.32.89.105:8080	
23	190.55.186.229:80	sample_1	23	46.105.131.68:8080	sample_2	23	103.80.51.61:8080	sample_3
24	188.80.27.54:80	_main_payload	24	116.91.240.96:80	_main_payload	24	54.38.143.245:8080	_main_payload
25	41.185.29.128:8080		25	118.243.83.70:80		25	113.203.238.130:80	
26	177.130.51.198:80		26	190.117.101.56:80		26	50.116.78.109:8080	
27	185.208.226.142:8080	C&C lists	27	103.229.73.17:8080	C&C lists	27	195.201.56.70:8080	C&C lists
28	190.194.12.132:80	COLC CARCO	28	5.79.70.250:8080		28	109.99.146.210:8080	cac cists
29	47.154.85.229:80		29	172.105.78.244:8080		29	75.127.14.170:8080	
30	85.246.78.192:80		30	95.76.142.243:80		30	172.193.14.201:80	
31	143.95.101.72:8080		31	113.193.239.51:443		31	203.56.191.129:8080	
32	75.127.14.170:8080		32	113.161.148.81:80		32	157.7.164.178:8081	
33	109.206.139.119:80		33	180.148.4.130:8080		33	46.32.229.152:8080	
34	197.221.227.78:80 58.27.215.3:8080		34	172.193.79.237:80 42.200.96.63:80		34	78.99.78.210:80 116.202.10.123:8080	
			35			35		
36	61.118.67.173:80		36	110.37.224.243:80		36	189.34.18.252:8080	
37 38	179.5.118.12:80 195.201.56.70:8080		37	212.198.71.39:80 185.80.172.199:80		37	114.158.126.84:80 201.193.160.196:80	
38	195.201.56.70:8080		38 39	185.80.172.199:80		38	201.193.160.196:80 79.133.6.236:8080	
39	190.164.135.81:80		39	153.229.219.1:443		39	202.29.237.113:8080	
48	187.193.221.143:80		40	190.55.186.229:80		40	202.29.237.113:8080	
41	78,90,78,210:80		41	86.123.55.0:80		41	172.96.190.154:8080	
42	117.2.139.117:443		42	94.212.52.40:80		42	74.208.173.91:8080	
43	120.51.34.254:80		43	37.46.129.215:8080		43	139.59.61.215:443	
44	139.59.12.63:8080		45	82.78.179.117:443		44	117.2.139.117:443	
40	139.09.12.03.0000		40	62.76.179.117:443		45	117.2.159.117:443	

Fig 42. List of IP:Port used by payloads

5.6. RSA Public Key

Through analysis, Emotet embeds an RSA public key in payloads. This RSA public key is also stored as a regular encrypted string and is decoded just like we did with strings. This key will then be used for the secure communication with the the C2 above.

All three payloads above after decrypt have the same RSA Public Key:



Fig 43. RSA Public Key after decrypted

5.7. Enumerating running processes

To get the list of the processes running on the victim machine, the payloads use APIs function **CreateToolhelp32Snapshot**; **Process32FirstW**; **Process32NextW**. List the processes are guaranteed:

- No process names where parent process ID is 0.
- No process is executed by Emotet.
- No duplicated process names.

			1005			_		<i>.</i>	er-						_		0241005303
	н		96C					30			96			Ċ.			eax 0x2F037A25
		884						74		11920	2.0				γ.		0x406679
		884						30		778	24			Ċ.			eax 0x31740781
		884						BF 8									0x406888
		884						8D4						16			eck, dward ptr ss:[esp+0x74]
		004								SEE							<f call="" get="" list="" process="" running="" to=""></f>
		0.04				_				EPR							eax, \$120907080
		10.04						60 60						10			0x406573
		884						69 88		811				10			
		885							122					10			eax, 0:210058FD 0:405572
_		4	1004	<u></u>	_	-	-	EV				_				_	1010577
-	_	• U	_	_	_	_	_	_		-	_			_	_	_	
eax=1																	
								-									
.text:00	106	E F	16	acı	5.C	e B	205		500	Æ							
El Duno 1		412.0			1.	i Du	-	1	-	Due		Τ.	12.0		e	-	Watch-1 le-Locals 3 Struct
	_		, and			8.04	-	2	1.1	o'unit	- 1			Citto	Sr.	5.22	unity Seriates
Address	He:	ĸ											_				ASCII
GOSCDADO	54	65	61	72	63	68	46	69	6C	74	65	72	48	6F	73	74	SearchFilterHost
00SCDAE0	2E	65	78	65	20	53	65	61	72	63	68	58	72	68	74	6F	.exe.SearchProto
eescoare	64	6F	60	48	6F	73	74	2E	65	78	65	20	41	75	74	őF	colHost.exe.#wto
00500800	72	75	6E	78	36	34	26	65	78	65	20	69	64	61	2E	65	runs64.exe.ida.e
005CD010	78	65	20	78	60	75	67	69	6E	SF	68	6F	73	74	2E	65	xe,plugin hest.e
005CDB20	78	65	20	78	75	62	60	69	60	65	SF	74	65	78	74	2E	xe, subline text,
005CDB30	65	78	65	20	45	76	65	72	79	74	68	69	6E	67	2E	65	exe Everything.e
00500840	78	45	26	78	61	73	68	6.8	65	72	78	26	65	78	45	20	xe,taskhost.exe,
005CDB50	65	78	78	6C	6F	72	65	72	2E	65	78	65	20	64	77	60	
00500060	21	65	78	65	20	53	65	61	72	ő.	68	49	66	64	45	78	.exe.SearchIndex
005CDB70	65	72	2E	65	78	65	20	60	73	64	74	63	2E	65	78	65	er_exe_msdtc.exe
005CDB80	20	57	60	69	50	72	76	53	45	2E	65	78	65	20	64	60	.UniProSE.exe.dl
BRSCDB9B	40	68	6F	73	74	2E	65	78	45	20	76	6D	74	45	45	60	
BESCHBAR		64	2E	65	78	65	20	56	17	41	75	74	68		65	72	
RESCORD	76	6.9	63	65	25	65	78	45	28	73	78	67	61	60	73	76	vice.exe.spoolsv
BBSCDBCD	26	65	78	65	20	76		33	64	73	65	72	76	69	63	65	.exe.un3dservice
BISCOBD B	21	65	78	65	20	73	76	63	68	6F	78	74	2E	45	78	65	.exe.suchest.exe
BBSCDRED	20	60	72	60	26	60	78	4.5	26	6C	72	61	72	72	25	65	.1sn.exe.1sass.e
BASCOBER	78	45	26	78	65	72	77	10	63	65	73	2E	65	78	45	20	xe,services.exe,
eescocee	77	40	45	6C	6E	67	10	10	36	65	78	65	20	77	10	6E	winlogon.exe.win
005CDC10	-	65	69	78	25	45	78	46	20	63	73	72	73	73	26	65	init_exe_csrss_e
005CDC20		65	20	28	60	73	23	22	65		65	00	00			88	
					122				110		12		122		20	12	

Fig 44. The payloads collect a list of the processes running on the victim machine

6. Conclusion

Emotet was first discovered in 2014 as a banking Trojan, over time it continues to evolve and has always been a leading threat to organizations around the world. Emotet has once again proven to be an advanced threat capable of adapting and evolving quickly in order to wreak more havoc. This malware is mainly distributed through email spam campaigns, so to prevent it, organizations should regularly train information security awareness for end users.

7. References / Further Reading

Click here for Vietnamese version.

Tran Trung Kien (aka m4n0w4r)

Malware Analysis Expert

R&D Center – VinCSS (a member of Vingroup)