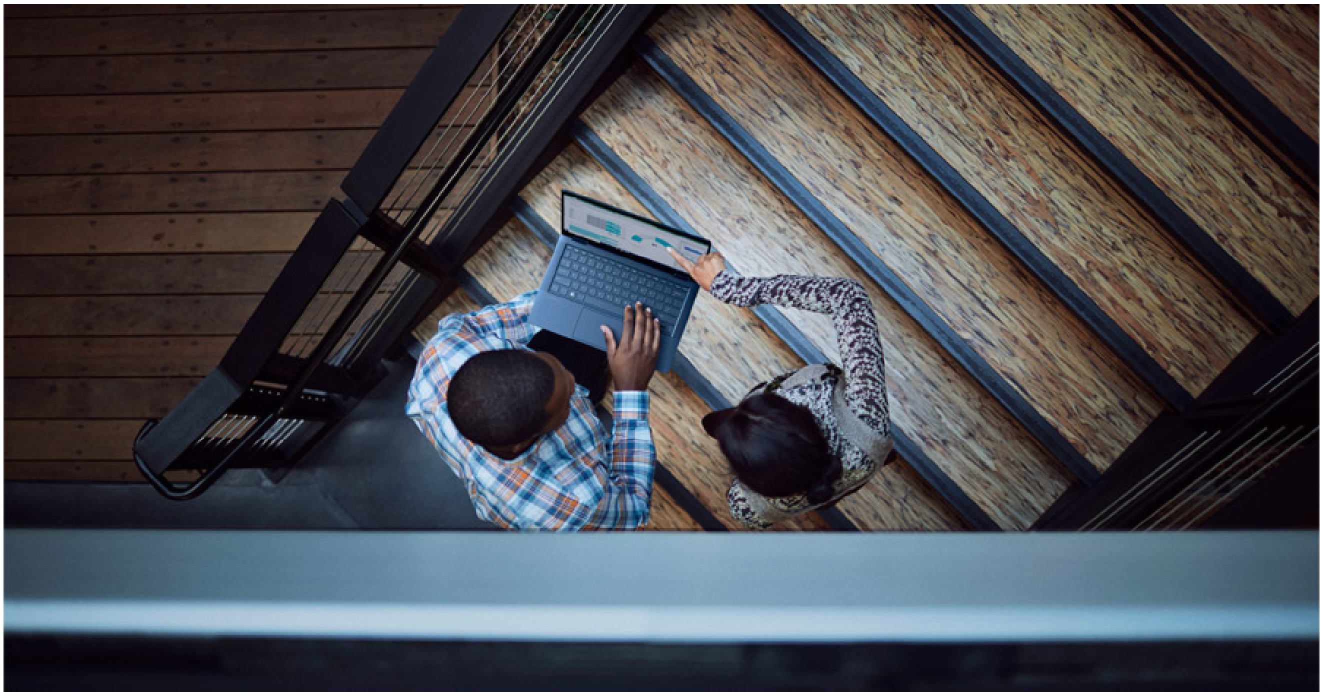
Recent Posts

December 9, 2021

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



Emotet's Return: What's Different?

On 15 November 2021, Emotet 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 law enforcement, 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 HP Wolf Security – 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).

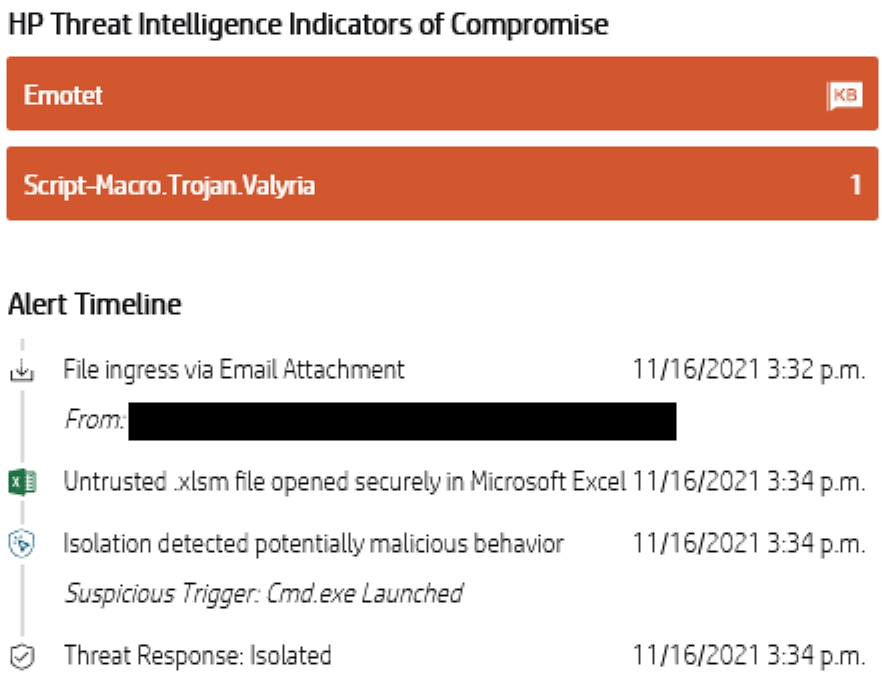


Figure 1 – Alert timeline showing user opening a malicious Emotet spreadsheet.

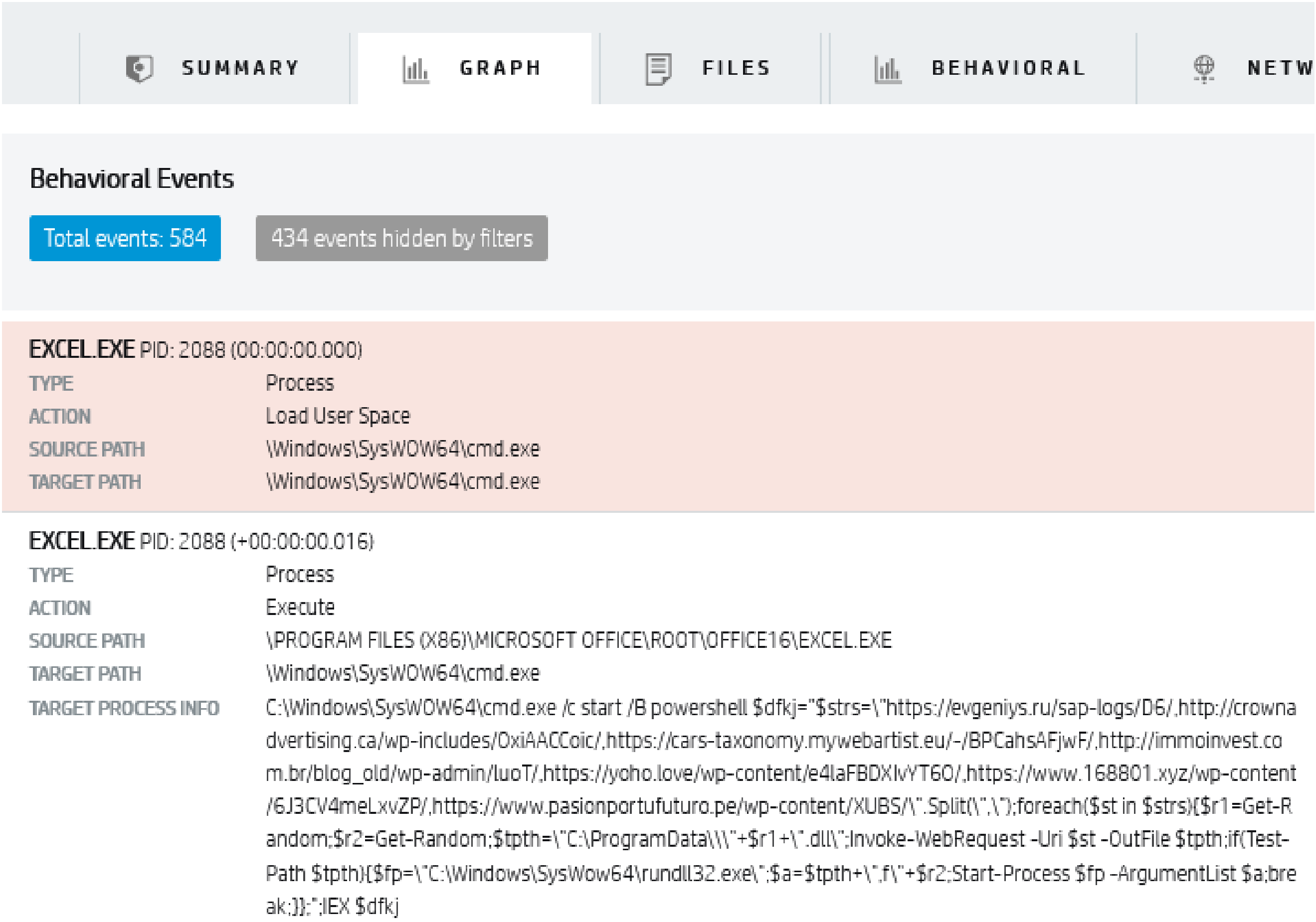


Figure 2 – Snippet from behavioral trace captured by HP Sure Click.

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 Threatray, 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
ba758c64519be23b5abe7991b71cdcece30525f14e225f2fa07bbffdf406e539	0x6c3f193c	61a47ebee921db8a16a8f070edcb86b5efd47a8d185bf4691b57e76f697981f9	0x6c408721
ba758c64519be23b5abe7991b71cdcece30525f14e225f2fa07bbffdf406e539	0x6c3f1db2	61a47ebee921db8a16a8f070edcb86b5efd47a8d185bf4691b57e76f697981f9	0x6c3f2c93
ba758c64519be23b5abe7991b71cdcece30525f14e225f2fa07bbffdf406e539	0x6c3f21c2	61a47ebee921db8a16a8f070edcb86b5efd47a8d185bf4691b57e76f697981f9	0x6c3fb79b
ba758c64519be23b5abe7991b71cdcece30525f14e225f2fa07bbffdf406e539	0x6c3f226a	61a47ebee921db8a16a8f070edcb86b5efd47a8d185bf4691b57e76f697981f9	0x6c407b8d
ba758c64519be23b5abe7991b71cdcece30525f14e225f2fa07bbffdf406e539	0x6c3f2735	61a47ebee921db8a16a8f070edcb86b5efd47a8d185bf4691b57e76f697981f9	0x6c3fbebd
ba758c64519be23b5abe7991b71cdcece30525f14e225f2fa07bbffdf406e539	0x6c3f4410	61a47ebee921db8a16a8f070edcb86b5efd47a8d185bf4691b57e76f697981f9	0x6c3f21c0
ba758c64519be23b5abe7991b71cdcece30525f14e225f2fa07bbffdf406e539	0x6c3f480a	61a47ebee921db8a16a8f070edcb86b5efd47a8d185bf4691b57e76f697981f9	0x6c3fae9e
ba758c64519be23b5abe7991b71cdcece30525f14e225f2fa07bbffdf406e539	0x6c3f4d32	61a47ebee921db8a16a8f070edcb86b5efd47a8d185bf4691b57e76f697981f9	0x6c4007d3
ba758c64519be23b5abe7991b71cdcece30525f14e225f2fa07bbffdf406e539	0x6c3f51c2	61a47ebee921db8a16a8f070edcb86b5efd47a8d185bf4691b57e76f697981f9	0x6c3f26a0
ba758c64519be23b5abe7991b71cdcece30525f14e225f2fa07bbffdf406e539	0x6c411b95	61a47ebee921db8a16a8f070edcb86b5efd47a8d185bf4691b57e76f697981f9	0x6c3f84d8
ba758c64519be23b5abe7991b71cdcece30525f14e225f2fa07bbffdf406e539	0x6c412081	61a47ebee921db8a16a8f070edcb86b5efd47a8d185bf4691b57e76f697981f9	0x6c3f3743
ba758c64519be23b5abe7991b71cdcece30525f14e225f2fa07bbffdf406e539	0x6c412a78	61a47ebee921db8a16a8f070edcb86b5efd47a8d185bf4691b57e76f697981f9	0x6c401f54
ba758c64519be23b5abe7991b71cdcece30525f14e225f2fa07bbffdf406e539	0x6c3f1c20		
ba758c64519be23b5abe7991b71cdcece30525f14e225f2fa07bbffdf406e539	0x6c3f1cf0		
ba758c64519be23b5abe7991b71cdcece30525f14e225f2fa07bbffdf406e539	0x6c3f3894		
ba758c64519be23b5abe7991b71cdcece30525f14e225f2fa07bbffdf406e539	0x6c3f3fd4		

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.

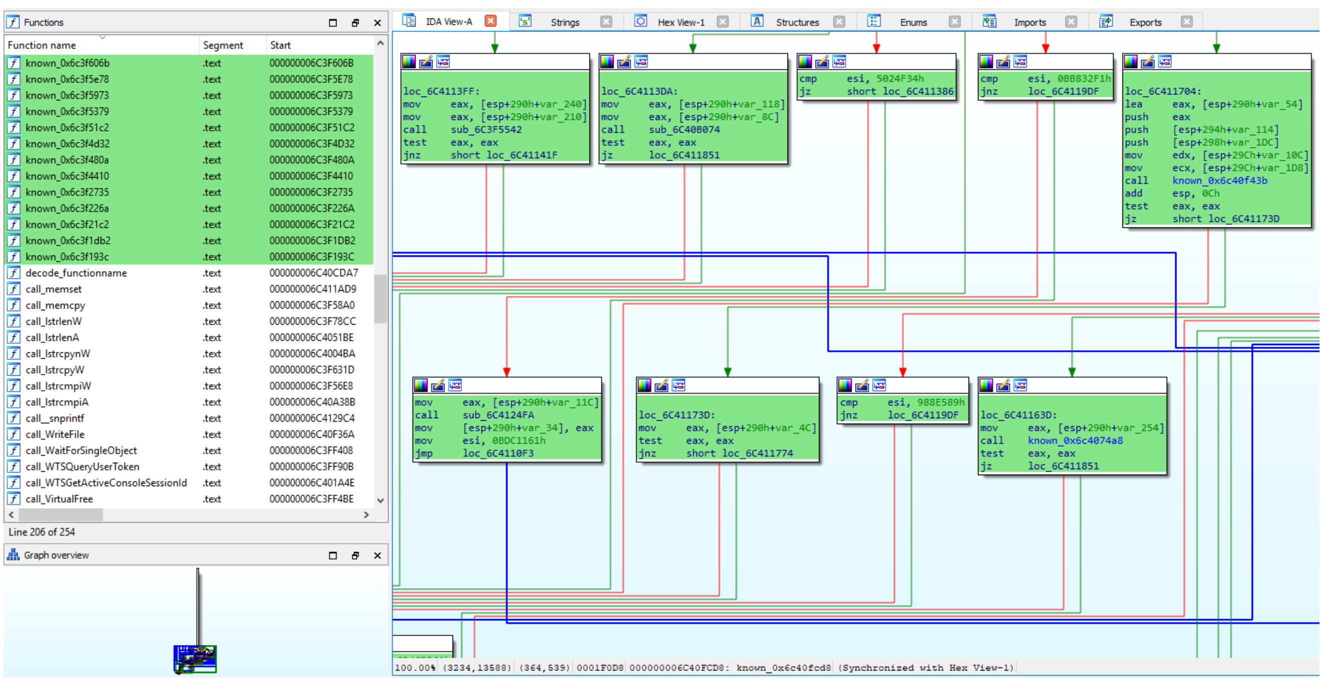
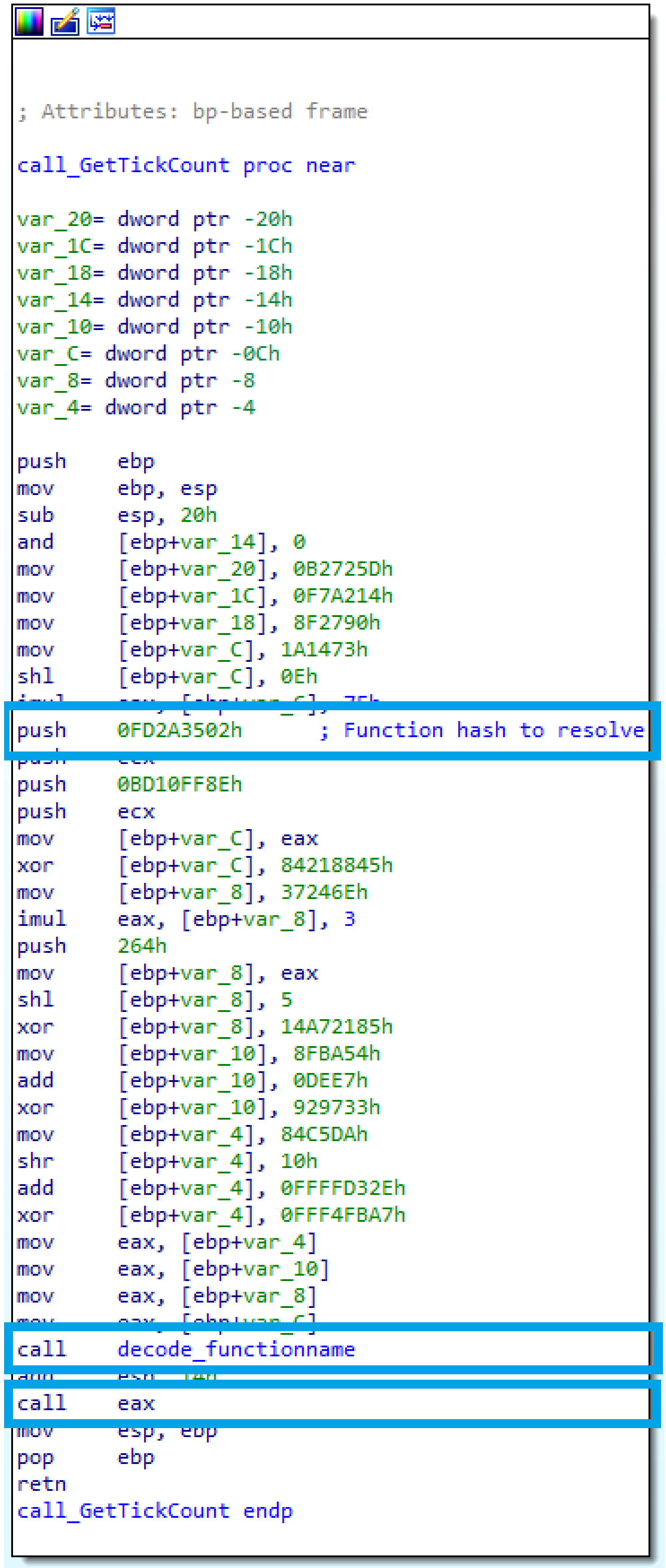


Figure 4 – IDA Pro disassembly of the November 2021 Emotet sample with known functions in green.

Windows API function resolution technique

One of the ways Emotet hides its capabilities is by resolving Windows API functions at runtime. 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.



```
; Attributes: bp-based frame

call_GetTickCount 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 -0Ch
var_8= dword ptr -8
var_4= dword ptr -4

push    ebp
mov     ebp, esp
sub     esp, 20h
and     [ebp+var_14], 0
mov     [ebp+var_20], 0B2725Dh
mov     [ebp+var_1C], 0F7A214h
mov     [ebp+var_18], 8F2790h
mov     [ebp+var_C], 1A1473h
shl     [ebp+var_C], 0Eh
push    0FD2A3502h      ; Function hash to resolve
push    ecx
push    0BD10FF8Eh
push    ecx
mov     [ebp+var_C], eax
xor     [ebp+var_C], 84218845h
mov     [ebp+var_8], 37246Eh
imul    eax, [ebp+var_8], 3
push    264h
mov     [ebp+var_8], eax
shl     [ebp+var_8], 5
xor     [ebp+var_8], 14A72185h
mov     [ebp+var_10], 8FBA54h
add     [ebp+var_10], 0DEE7h
xor     [ebp+var_10], 929733h
mov     [ebp+var_4], 84C5DAh
shr     [ebp+var_4], 10h
add     [ebp+var_4], 0FFFFD32Eh
xor     [ebp+var_4], 0FFF4FBA7h
mov     eax, [ebp+var_4]
mov     eax, [ebp+var_10]
mov     eax, [ebp+var_8]
call    decode_functionname
call    eax
mov     esp, ebp
pop     ebp
retn
call_GetTickCount endp
```

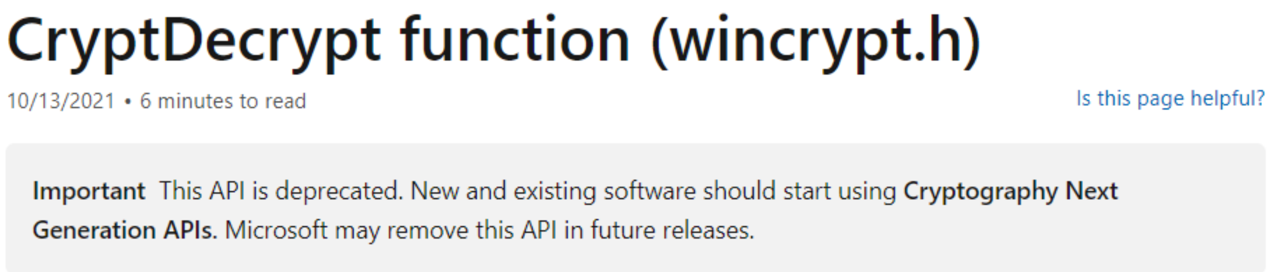
Figure 5 – Emotet’s Windows API wrapper function.

Since these wrapper functions are not classified as similar, we wrote a Python script 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 Bcrypt cryptography functions. 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 – CryptDecrypt API documentation from Microsoft.

In addition to the changes in cryptography, Emotet now uses the function RtlAllocateHeap to allocate heap memory. Normally a program calls HeapAlloc 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.

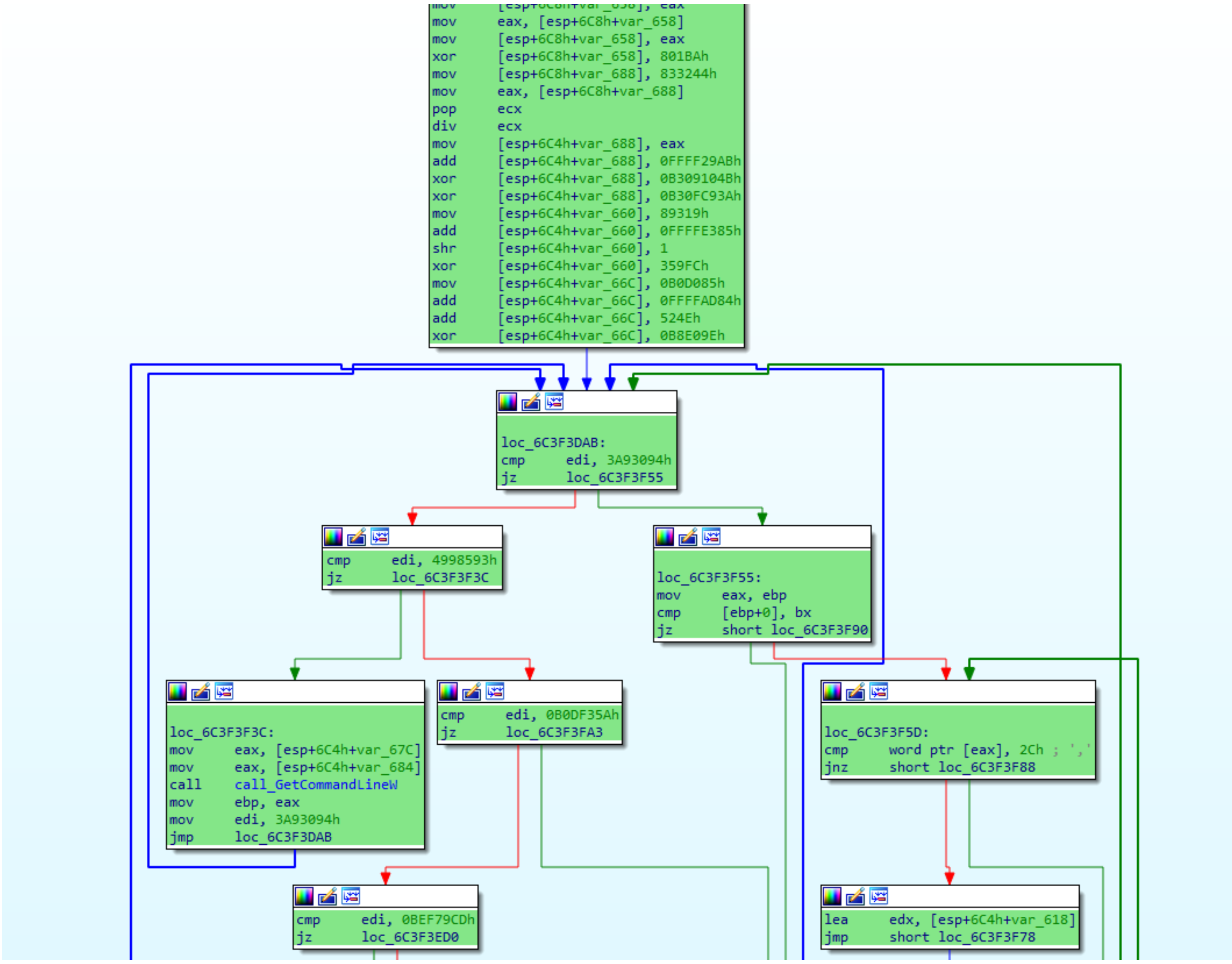


Figure 7 – Control flow graph showing switch case obfuscation.

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.

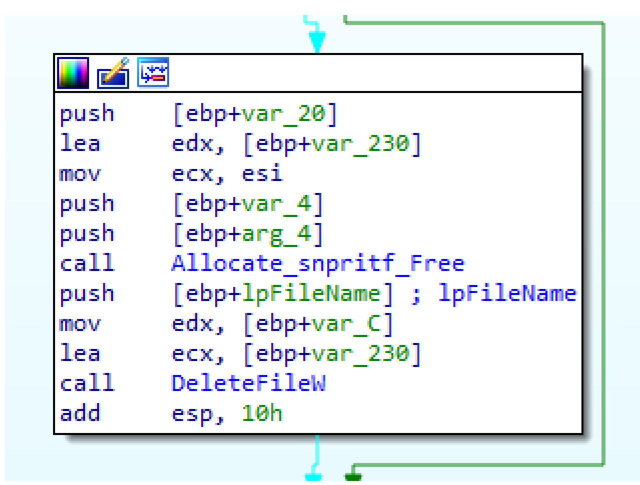


Figure 8 – Sample from January 2021 calling a sub-function leading to further execution and API calls.

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).

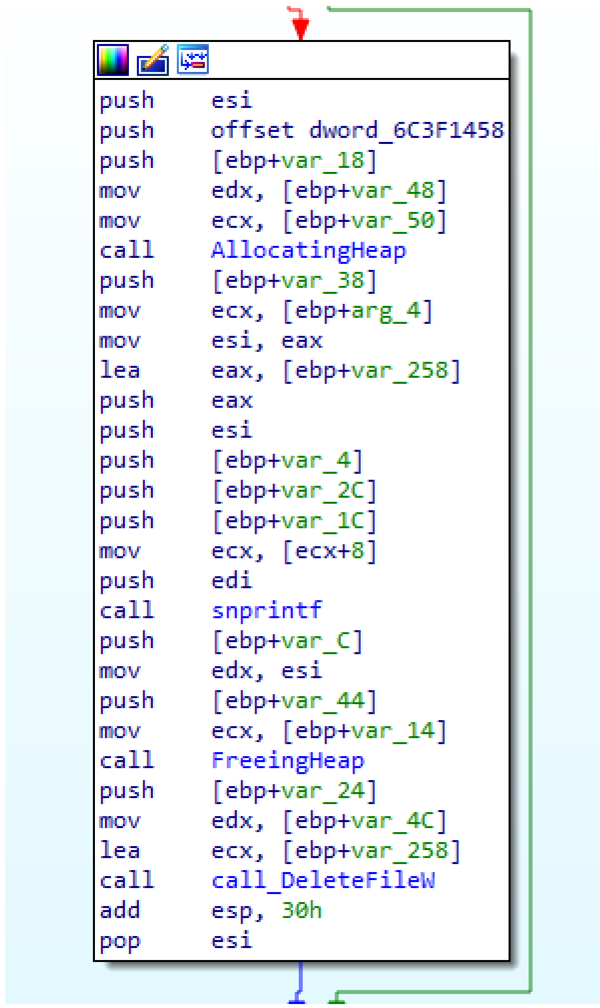


Figure 9 – Sample from November 2021 using direct function calls instead of sub-functions.

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 IDA database and Python script used in this article.

Tags

code analysis emotet