Informe Código Dañino
CCN-CERT ID-11/21

RansomEXX_Win ransomware

Agosto 2021

Edita:

GOBIERNO DE ESPAÑA MINISTERIO DE DEFENSA

CCN centro criptológico nacional

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

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 *txd0t*). 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	480af18104198ad3db1518501ee58f9c4aecd19dbbf2c5dd7694d1d87e9aeac7

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 dañino:

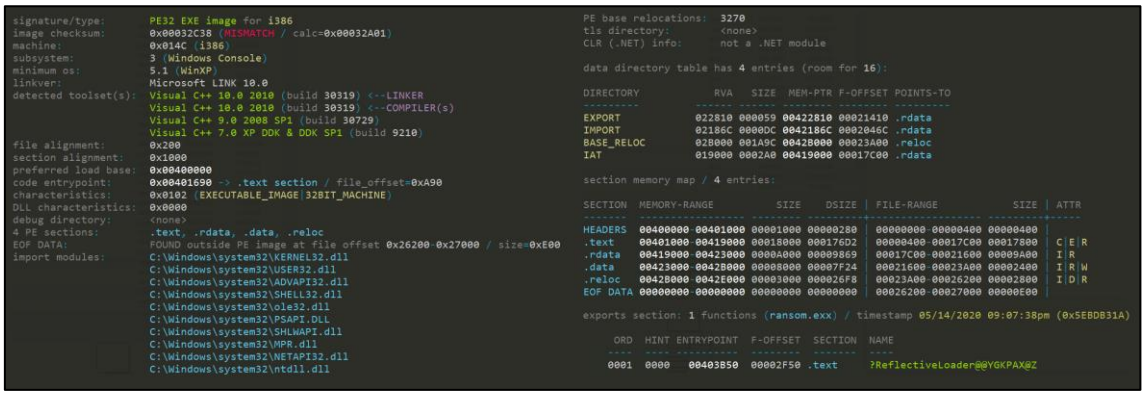


Figura 1. Características estáticas del código dañino

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.

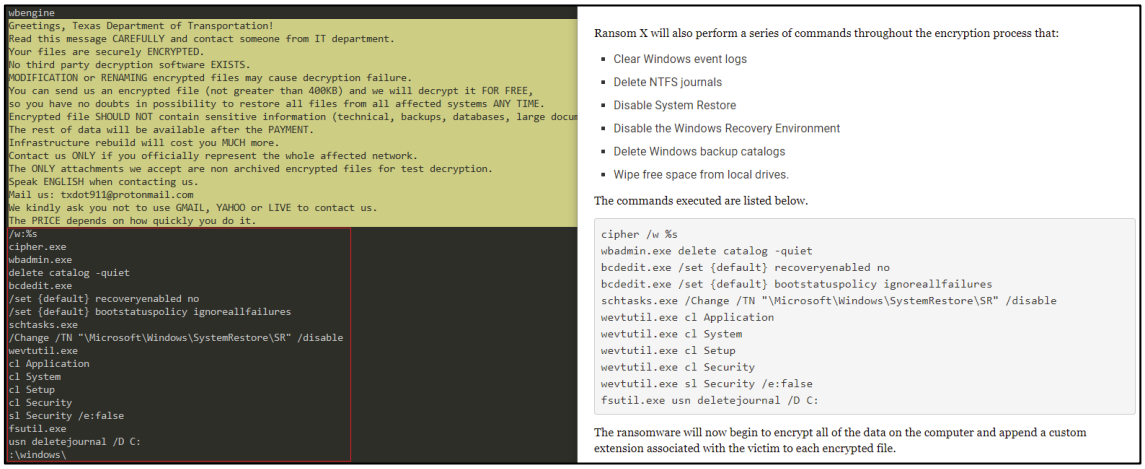


Figura 2. Strings: mensaje TXT, servicios Windows

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.

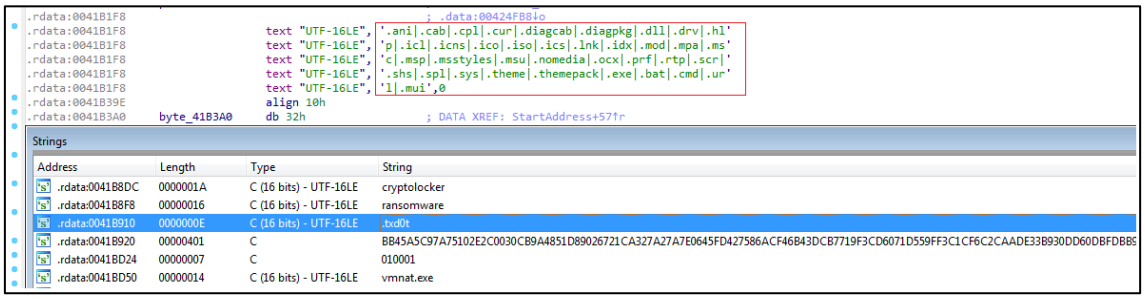


Figura 3. Strings: extensiones

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.

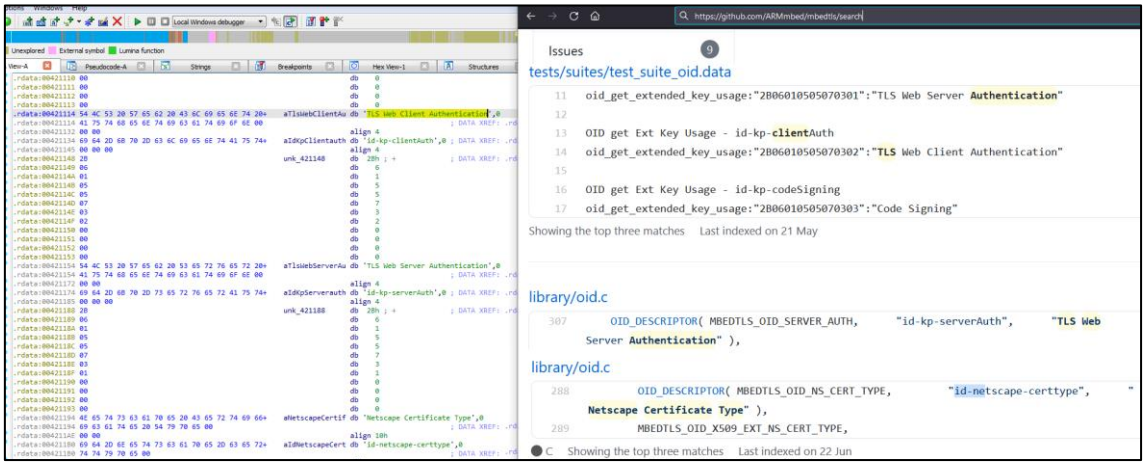


Figura 4. Strings: mbedtls

El código dañino 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).

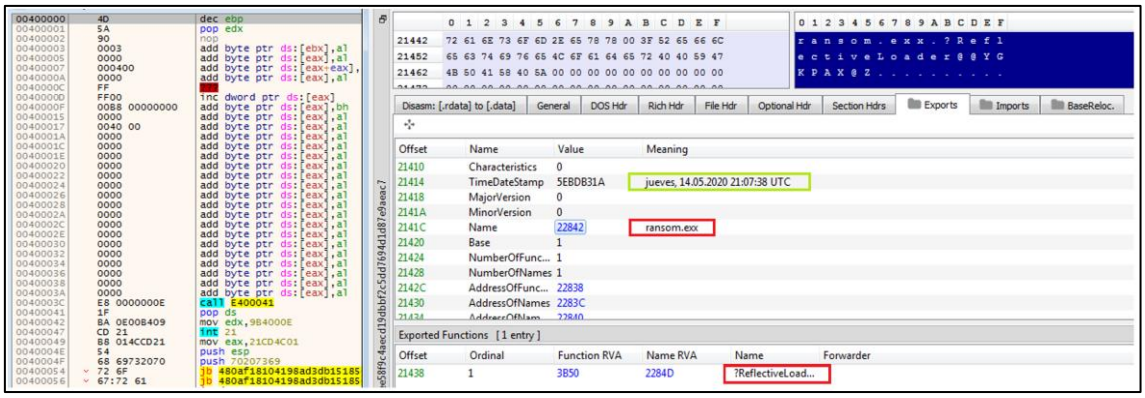


Figura 5. Símbolo exportado: ReflectiveLoader

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).

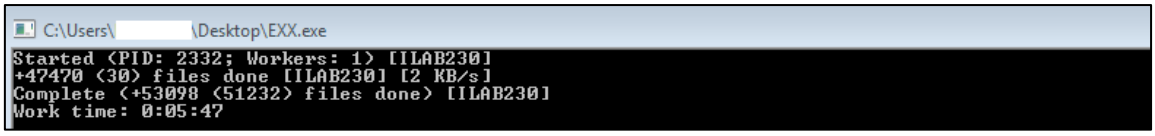


Figura 6. Información sobre el proceso de cifrado

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***.

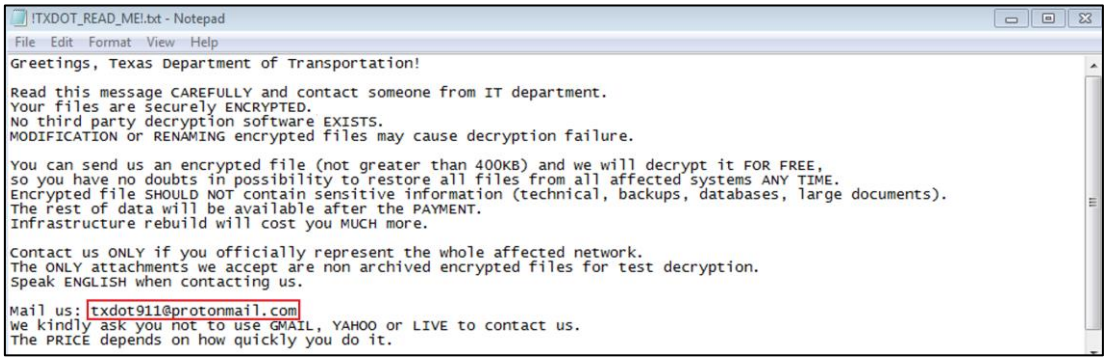


Figura 7. Cuenta atacantes: txdot911@protonmail.com

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 dañino exporta el símbolo "*ReflectiveLoader*" comúnmente utilizado por herramientas que tienen capacidad de cargarse de forma *reflectiva* en memoria.

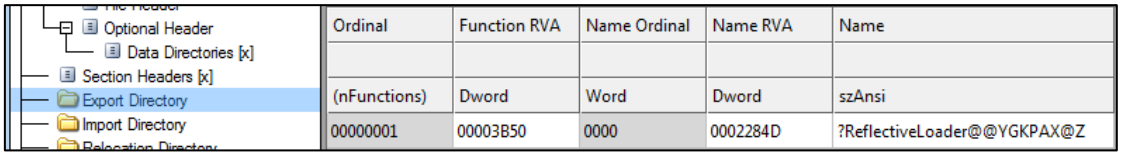


Figura 8. Export ReflectiveLoader

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 dañino 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.

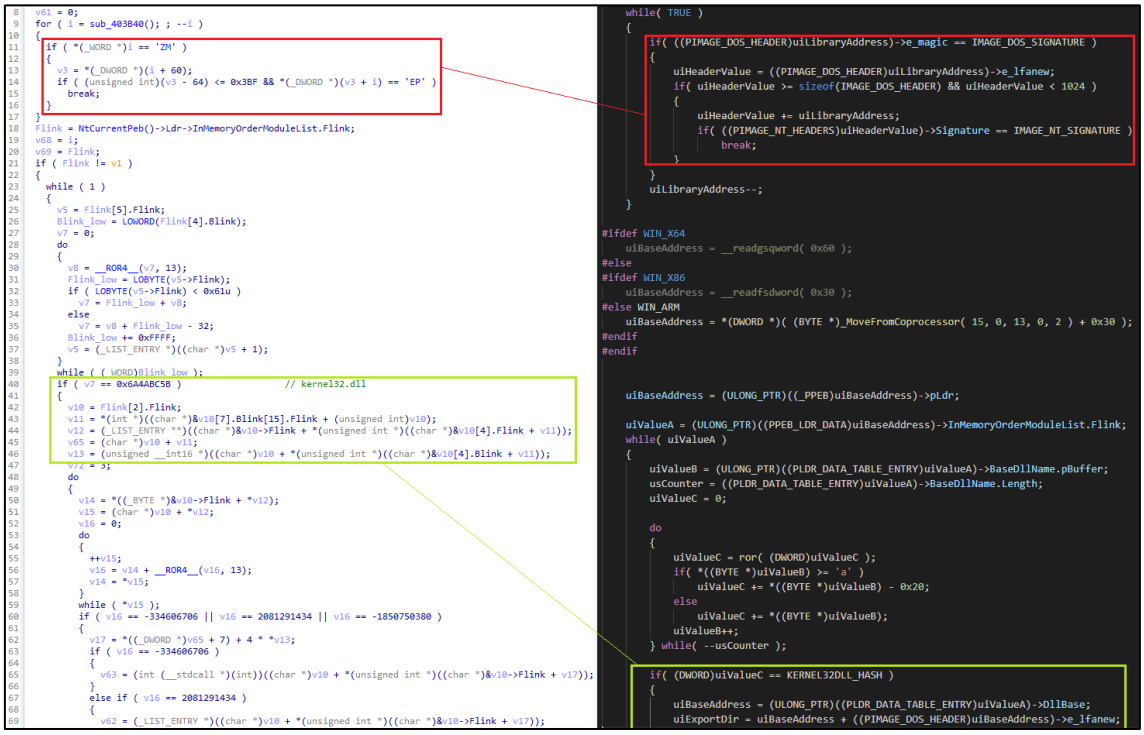


Figura 9. Export ReflectiveLoader / ReflectiveLoader.c (https://github.com/stephenfewer)

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.

Figura 10. flag SE_DACL_PROTECTED

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

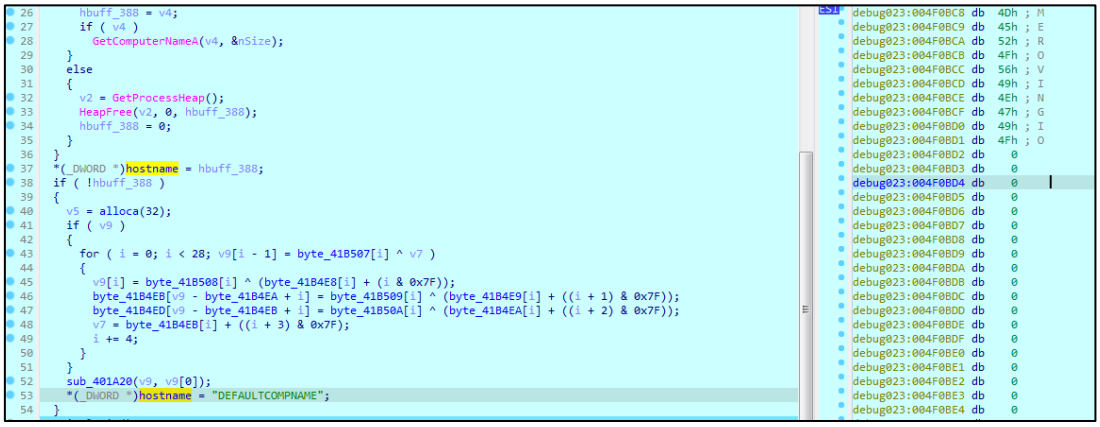


Figura 11. Hostname

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 dañino 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 dañino 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).

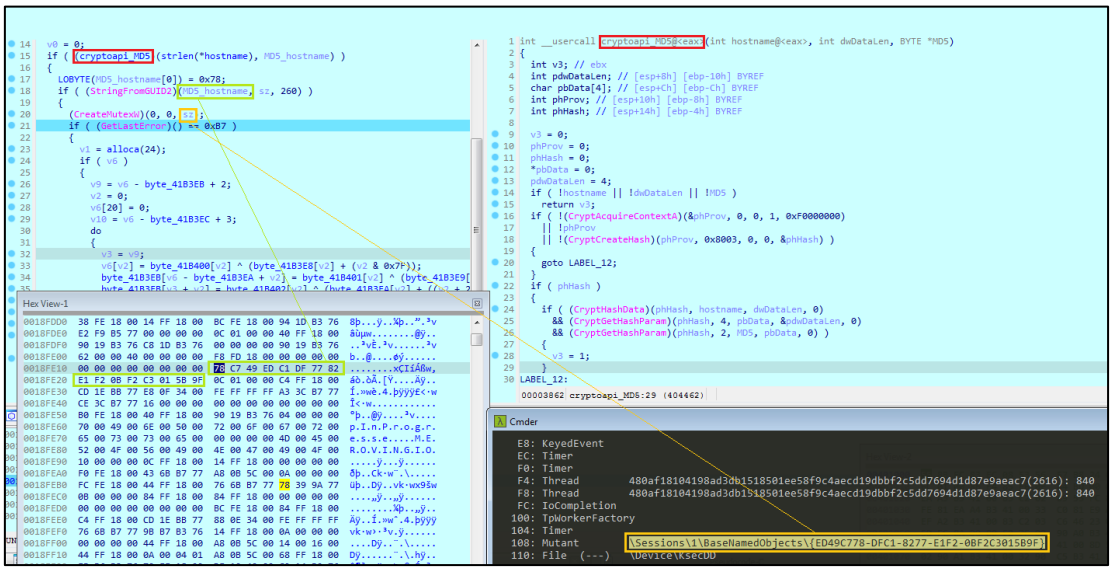


Figura 12. Creación objeto Mutex

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()*.

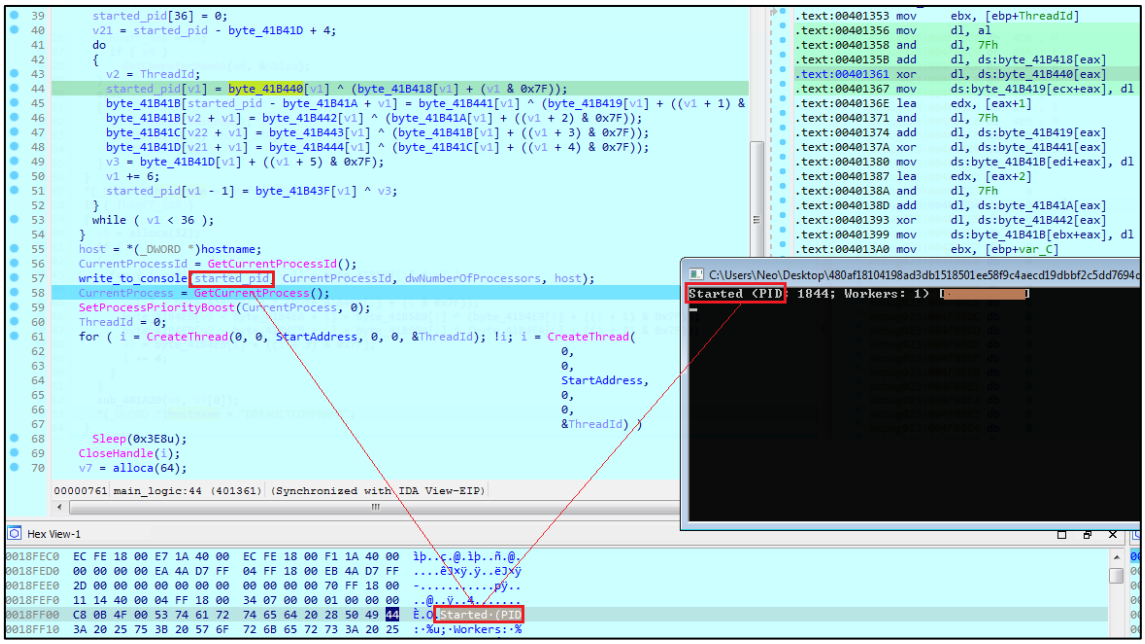


Figura 13. Consola / Boost threads

El código dañino 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 = OpenServiceW(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);
  }
}
result = L"ARSM";
v2[0] = (int)L"AVP";
v2[1] = (int)L"AcrSch2Svc";
v2[2] = (int)L"Acronis VSS Provider";
v2[3] = (int)L"AcronisAgent";
v2[4] = (int)L"AcronixAgent";
v2[5] = (int)L"Antivirus";
v2[6] = (int)L"BackupExecAgentAccelerator";
v2[7] = (int)L"BackupExecAgentBrowser";
v2[8] = (int)L"BackupExecDeviceMediaService";
v2[9] = (int)L"BackupExecJobEngine";
v2[10] = (int)L"BackupExecManagementService";
v2[11] = (int)L"BackupExecRPCService";
v2[12] = (int)L"BackupExecVSSProvider";
v2[13] = (int)L"DCAgent";
v2[14] = (int)L"DbxSvc";
v2[15] = (int)L"EPSecurityService";
v2[16] = (int)L"EPUpdateService";
v2[17] = (int)L"ESHASRV";
v2[18] = (int)L"EhttpSrv";
v2[19] = (int)L"Enterprise Client Service";
v2[20] = (int)L"EraserSvc11710";
v2[21] = (int)L"EsgShKernel";
v2[22] = (int)L"FA_Scheduler";
v2[23] = (int)L"IISAdmin";
v2[24] = (int)L"IMAP4Svc";
v2[25] = (int)L"KAVFS";
v2[26] = (int)L"KAVFSGT";
v2[27] = (int)&off_41C12C;
v2[28] = (int)&off_41C144;
v2[29] = (int)L"MMS";
v2[30] = (int)L"MSExchangeAB";
v2[31] = (int)L"MSExchangeADTopology";
v2[32] = (int)L"MSExchangeAntispamUpdate";
v2[33] = (int)L"MSExchangeES";
v2[34] = (int)L"MSExchangeEdgeSync";
v2[35] = (int)L"MSExchangeFBA";
v2[36] = (int)L"MSExchangeFDS";
v2[37] = (int)L"MSExchangeIS";
v2[38] = (int)L"MSExchangeMGMT";
v2[39] = (int)L"MSExchangeMTA";
v2[40] = (int)L"MSExchangeMailSubmission";
v2[41] = (int)L"MSExchangeMailboxAssistants";
v2[42] = (int)L"MSExchangeMailboxReplication";
v2[43] = (int)L"MSExchangeProtectedServiceHost";
v2[44] = (int)L"MSExchangeRPC";
v2[45] = (int)L"MSExchangeRepl";
v2[46] = (int)L"MSExchangeSA";
v2[47] = (int)L"MSExchangeSRS";
v2[48] = (int)L"MSExchangeSearch";
v2[49] = (int)L"MSExchangeServiceHost";
v2[50] = (int)L"MSExchangeThrottling";
v2[51] = (int)L"MSExchangeTransport";
v2[52] = (int)L"MSExchangeTransportLogSearch";
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 posexplotación.

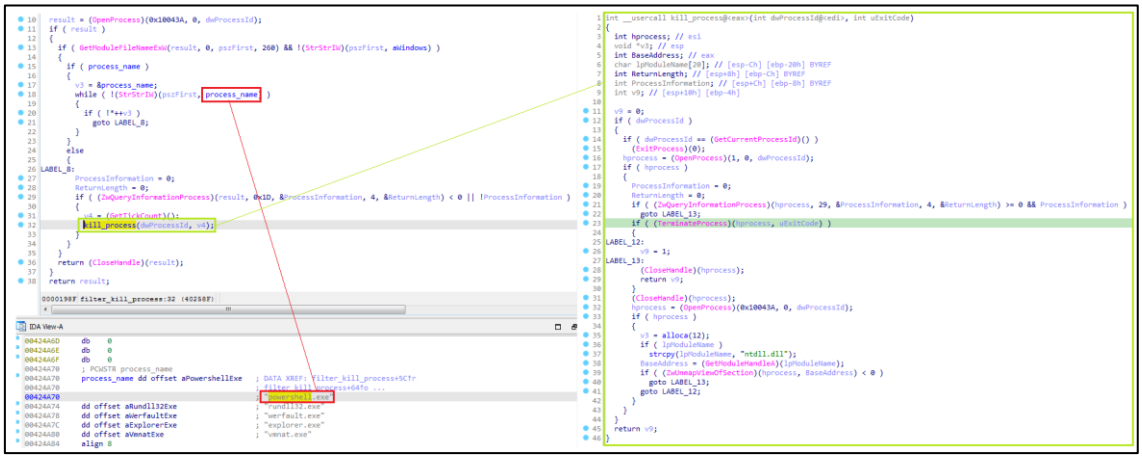


Figura 15. Finalización de procesos

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.

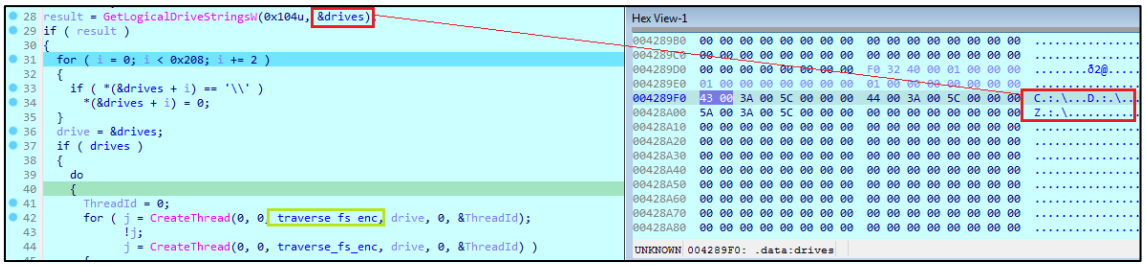


Figura 16. Obtención de unidades del sistema

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.

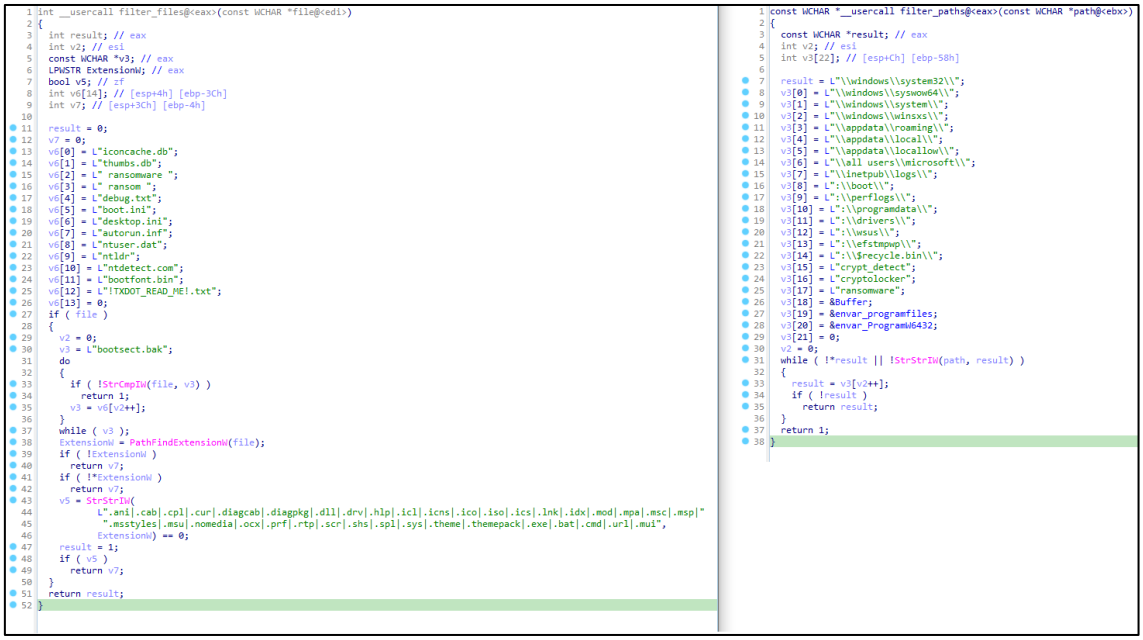


Figura 17. Exclusión ficheros, directorios y extensiones

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 jounal* (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.

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.

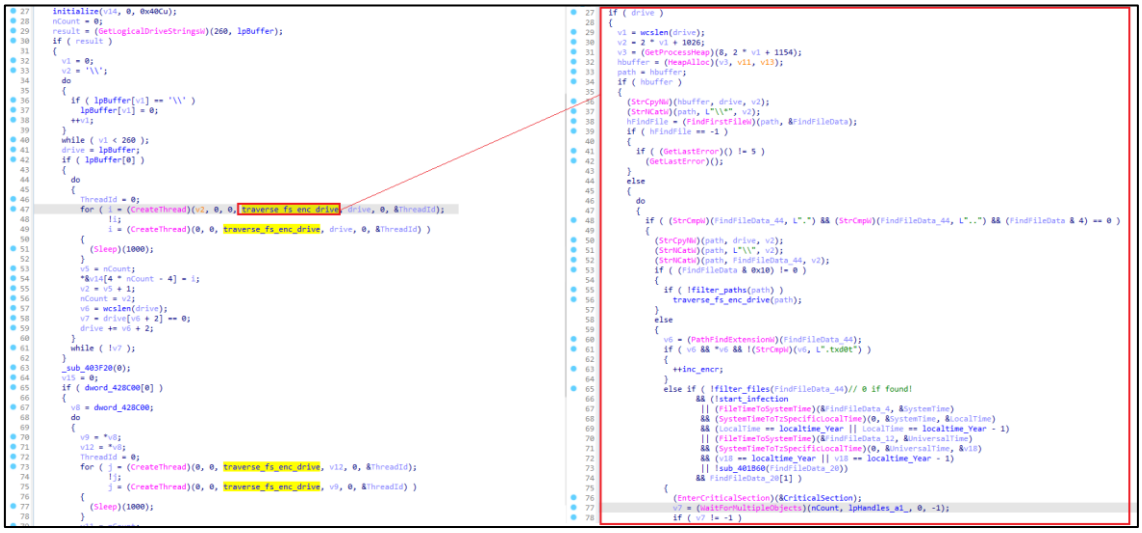


Figura 19. Recorrer ficheros de cada unidad

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.

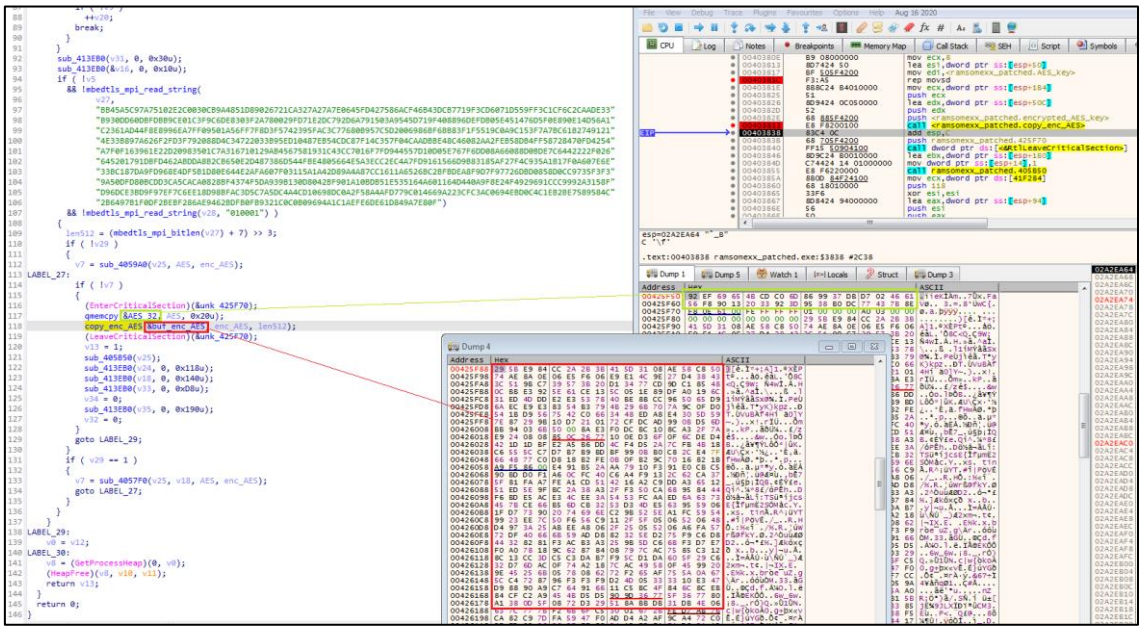


Figura 20. Generación clave AES / Cifrado clave AES con RSA (pub key)

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.

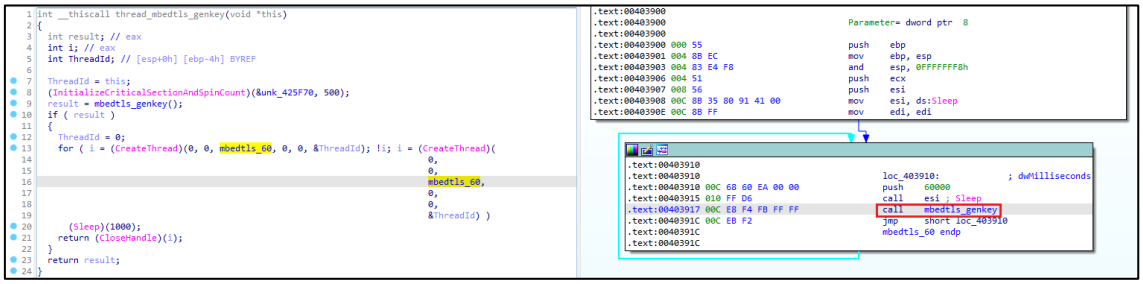


Figura 21. Generación de claves (60 segundos)

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

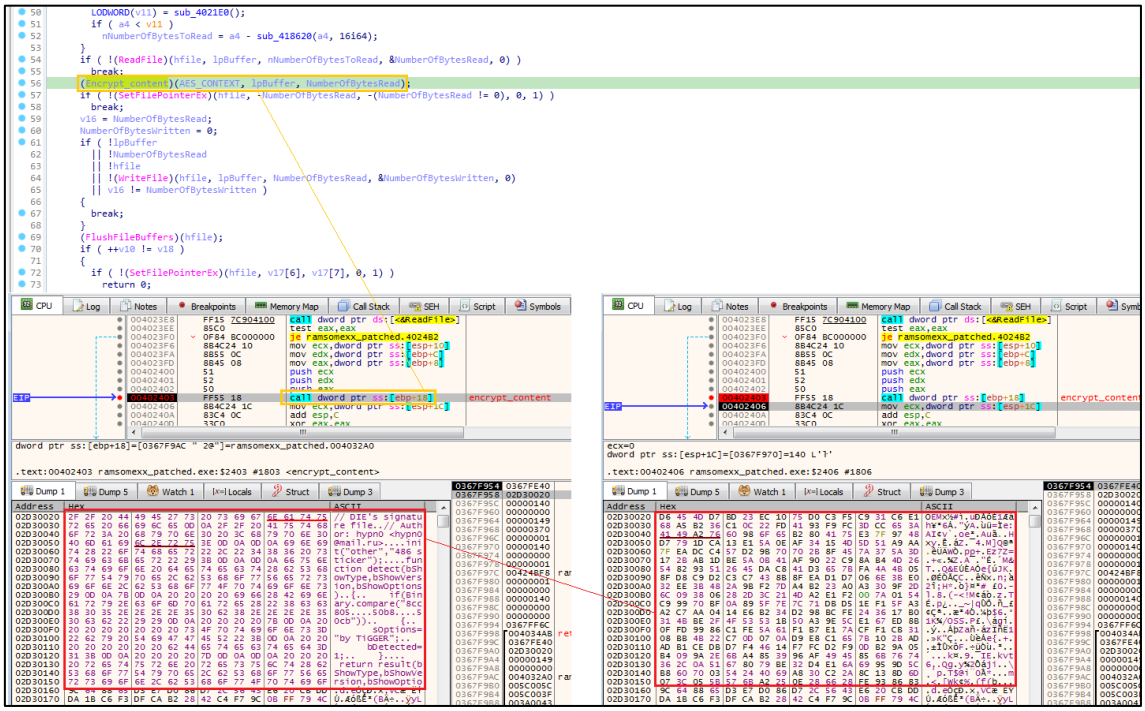
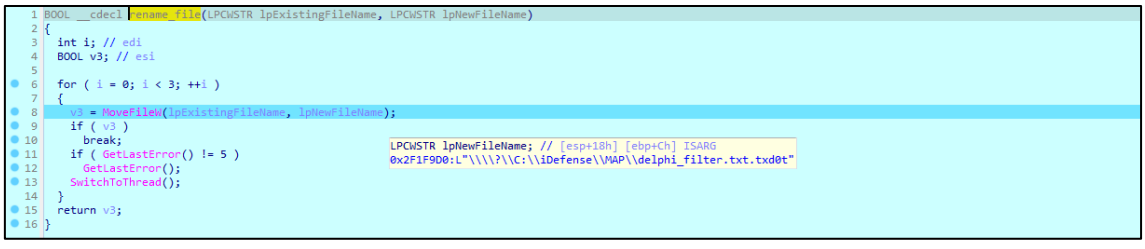


Figura 22. Cifrado fichero

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.



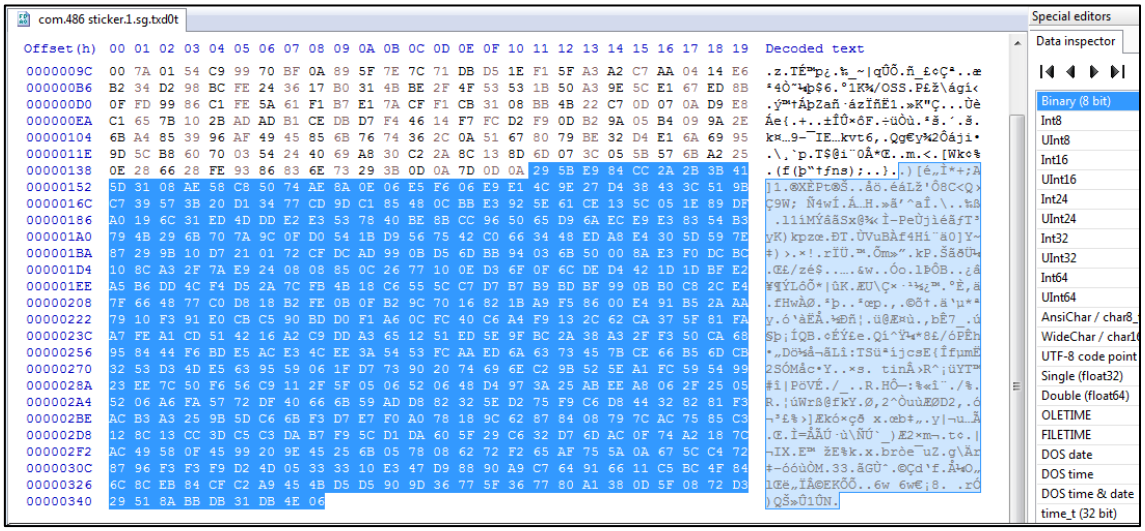


Figura 23. Fichero cifrado (clave AES cifrada con RSA)

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"*.

```
v1 = (CreateFileW)(pszPath, 0x40000000, 3, 0, 2, 128, 0);
v2 = v1;
if ( v1 != -1 )
{
  Buffer = -257;
  NumberOfBytesWritten = 0;
  if ( v1 )
  {
    if ( (WriteFile)(v1, &Buffer, 2, &NumberOfBytesWritten, 0) )
    {
      if ( NumberOfBytesWritten == 2 )
      {
        (FlushFileBuffers)(v2);
        NumberOfBytesWritten = 0;
        if ( (WriteFile)(
               v2,
               L"Greetings, Texas Department of Transportation!\r\n"
                "\r\n"
                "Read this message CAREFULLY and contact someone from IT department.\r\n"
                "Your files are securely ENCRYPTED.\r\n"
                "No third party decryption software EXISTS.\r\n"
                "MODIFICATION or RENAMING encrypted files may cause decryption failure.\r\n"
                "\r\n"
                "You can send us an encrypted file (not greater than 400KB) and we will decrypt it FOR FREE,\r\n"
                "so you have no doubts in possibility to restore all files from all affected systems ANY TIME.\r"
                "\n"
                "Encrypted file SHOULD NOT contain sensitive information (technical, backups, databases, large "
                "documents).\r\n"
                "The rest of data will be available after the PAYMENT.\r\n"
                "Infrastructure rebuild will cost you MUCH more.\r\n"
                "\r\n"
                "Contact us ONLY if you officially represent the whole affected network.\r\n"
                "The ONLY attachments we accept are non archived encrypted files for test decryption.\r\n"
                "Speak ENGLISH when contacting us.\r\n"
                "\r\n"
                "Mail us: txdot911@protonmail.com\r\n"
                "We kindly ask you not to use GMAIL, YAHOO or LIVE to contact us.\r\n"
                "The PRICE depends on how quickly you do it.\r\n",
               2032,
               &NumberOfBytesWritten,
               0) )
        {
          if ( NumberOfBytesWritten == 2032 )
            (FlushFileBuffers)(v2);
```

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 = ".txd0t" 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	MSExchangeTransport
AVP	MSExchangeTransportLogSearch
AcrSch2Svc	MSOLAP$SQL_2008
Acronis VSS Provider	MSOLAP$SYSTEM_BGC
AcronisAgent	MSOLAP$TPS
AcronixAgent	MSOLAP$TPSAMA
Antivirus	MSSQL$BKUPEXEC
BackupExecAgentAccelerator	MSSQL$ECWDB2
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$SQL_2008
EPUpdateService	MSSQL$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
MSExchangeAB	MSSQLFDLauncher$TPSAMA
MSExchangeADTopology	MSSQLSERVER
MSExchangeAntispamUpdate	MSSQLServerADHelper
MSExchangeES	MSSQLServerADHelper100
MSExchangeEdgeSync	MSSQLServerOLAPService
MSExchangeFBA	McAfeeEngineService
MSExchangeFDS	McAfeeFramework
MSExchangeIS	McAfeeFrameworkMcAfeeFramework
MSExchangeMGMT	McShield
MSExchangeMTA	McTaskManager
MSExchangeMailSubmission	MongoDB
MSExchangeMailboxAssistants	MsDtsServer
MSExchangeMailboxReplication	MsDtsServer100
MSExchangeProtectedServiceHost	MsDtsServer110
MSExchangeRPC	MySQL57
MSExchangeRepl	MySQL80
MSExchangeSA	nginx
MSExchangeSRS	NetMsmqActivator
MSExchangeSearch	OracleClientCache80
MSExchangeServiceHost	OracleServiceXE
MSExchangeThrottling	OracleXETNSListener
PDVFSService	Sophos Health Service
POP3Svc	Sophos MCS Agent
RESvc	Sophos MCS Client
ReportServer	Sophos Message Router
ReportServer$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	klnagent
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\\locallow\\
\\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
```