# Informe Código Dañino
# CCN-CERT ID-01/21

## Egregor

Enero 2021

Edita:

GOBIERNO DE ESPAÑA MINISTERIO DE DEFENSA



Fecha de Edición: enero 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, CERT GUBERNAMENTAL NACIONAL

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 **C4735B8F2A4DE4C9640CCC605602B6C6**, perteneciente a la familia de ransomware **Egregor**. El principal objetivo de esta muestra es cifrar los ficheros del sistema afectado, para posteriormente, solicitar el pago de un rescate, en bitcoins, a cambio de la herramienta de descifrado.

## 3. CAPA I

### 3.1 DETALLES GENERALES

La muestra analizada en este apartado es una Dll de 32 bits, sin firma digital y con el siguiente hash MD5:

| NOMBRE DEL FICHERO | MD5 |
|---|---|
| qq.dll | C4735B8F2A4DE4C9640CCC605602B6C6 |

La fecha de compilación es el 20 de noviembre de 2020, 21:58:17 (UTC), sin embargo, esta información no es del todo fiable, ya que se puede alterar fácilmente:

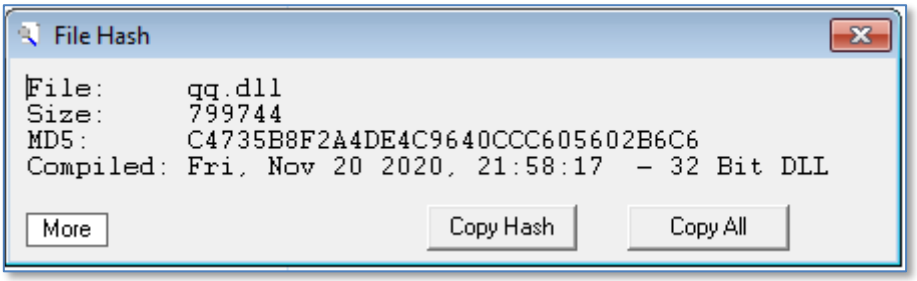


**Figura 1.** Fechas de compilación de la muestra.

La muestra no presenta propiedades del fichero.

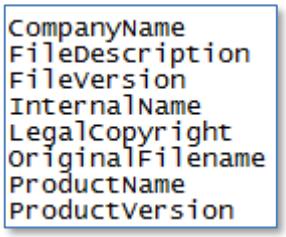


**Figura 2.** Las propiedades de los ficheros están vacías.

El malware contiene el nombre de un archivo de depuración: "**G:\Intel\Logs\qqqqq.pdb**"

## 3.2 ANÁLISIS TÉCNICO

El código dañino se encuentra altamente ofuscado, lo que dificulta el análisis estático de la muestra. Además, dispone de diferentes capas de ofuscación y cifrado. Cada capa utiliza a su vez técnicas anti-reversing y anti-debugging, incluyendo el cifrado del payload final mediante una clave que se tiene que pasar por la línea de comandos, lo que dificulta su ejecución en entornos de análisis automatizados (sandbox).

El malware se encuentra empaquetado, por lo que la tarea del código original es realizar el desempaquetado de la segunda capa. Inicialmente, el código dañino comprueba si existe el fichero "**C:\Heil Egregor\1488\ficker.py**". En caso afirmativo el malware detendrá su ejecución.

```
.text:6D0B7344 014                nop
.text:6D0B7345
.text:6D0B7345     loc_6D0B7345:                          ; CODE XREF: _DllMain@12_0:loc_6D0B7341↑j
.text:6D0B7345 014                push    80000000h       ; dwDesiredAccess
.text:6D0B734A 018                push    offset FileName ; "C:\\Heil Egregor\\1488\\ficker.py"
.text:6D0B734F 01C                jmp     loc_6D0B72B3
```

**Figura 3.** Fichero killswitch.

La Dll exporta 3 funciones, de las cuales **DllRegisterServer** es la utilizada por el malware para extraer la siguiente capa.

| Name | Address | Ordinal |
|---|---|---|
| DllInstall | 6D0B2DCE | 1 |
| DllRegisterServer | 6D0B1573 | 2 |
| DllUnregisterServer | 6D0B2162 | 3 |
| DllEntryPoint | 6D0B1460 | 268440672 [main entry] |

**Figura 4.** Funciones exportadas.

La función **DllRegisterServer** comienza comprobando los parámetros de la línea de comandos. Si el parámetro "**--del**" existe, el malware muestra el mensaje "**This is dummy messagebox**" mediante MessageBox() y terminará su ejecución. Esta capa acepta otro parámetro "**--dubisteinmutterficker**", que no tiene ninguna influencia en la ejecución del malware.

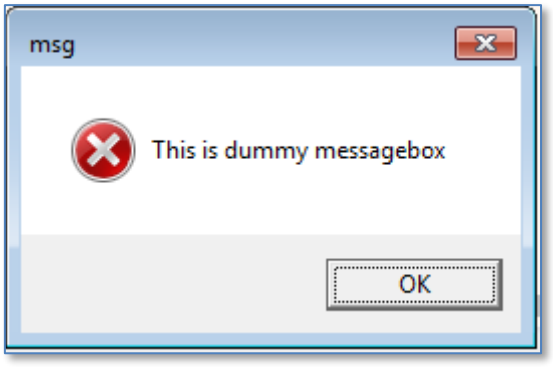


**Figura 5.** Mensaje mostrado usando la opción "**--del**".

Durante el siguiente paso, el código descifrará los datos que se encuentran en la sección "**.data**". Para ello realiza los siguientes pasos:

- Descifrar la sección "**.data**" mediante **XOR y la clave 0x06**. El resultado es un bloque de datos codificado en **Base64**.
- Decodificar el **Base64** mediante la función **CryptStringToBinaryA**. Como resultado se obtiene un bloque de datos cifrados mediante **ChaCha8**.
- Descifrar el bloque de datos mediante el algoritmo de cifrado **ChaCha8**, utilizando la siguiente clave e IV:

| CLAVE CHACHA8 |
| --- |
| Elon Musk 2024! To the future!!! |
| **IV CHACHA8** |
| SpaceX!! |

Una vez que los datos han sido descifrados, una Dll de 32 bits, es cargada y ejecutada en memoria mediante la técnica "**Reflective Load**".

# 4. CAPA II

## 4.1 DETALLES GENERALES

La muestra analizada en este apartado es una Dll de 32 bits, sin firma digital y con el siguiente hash MD5:

| NOMBRE DEL FICHERO | MD5 |
|---|---|
| - | 4FAFE7F3E3FF6E672DEEEC5AF3351EA7 |

La fecha de compilación es el 20 de noviembre de 2020, 21:53:05 (UTC), sin embargo, esta información no es del todo fiable, ya que se puede alterar fácilmente:

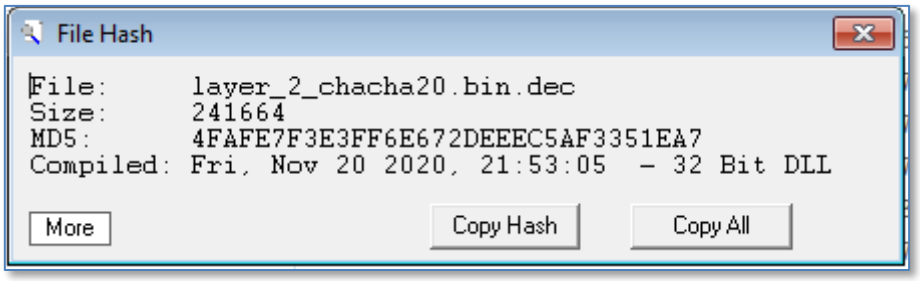


**Figura 6.** Fechas de compilación de la muestra.

La muestra no presenta propiedades del fichero.

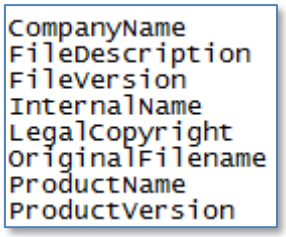


**Figura 7.** Las propiedades de los ficheros están vacías.

## 4.2 ANÁLISIS TÉCNICO

Al inicio de la capa II, el malware ejecuta un nuevo hilo, que contiene el payload principal de esta capa.

```
.text:00321960 ; BOOL __stdcall DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)
.text:00321960 _DllMain@12     proc near               ; CODE XREF: dllmain_dispatch(HINSTANCE__ * const,ulong,void * const)+6B↓p
.text:00321960                                         ; dllmain_dispatch(HINSTANCE__ * const,ulong,void * const)+83↓p
.text:00321960
.text:00321960 hinstDLL        = dword ptr  8
.text:00321960 fdwReason       = dword ptr  0Ch
.text:00321960 lpvReserved     = dword ptr  10h
.text:00321960
.text:00321960                 push    ebp
.text:00321961                 mov     ebp, esp
.text:00321963                 cmp     [ebp+fdwReason], 1
.text:00321967                 jnz     short loc_321985
.text:00321969                 push    0               ; lpThreadId
.text:0032196B                 push    0               ; dwCreationFlags
.text:0032196D                 push    0               ; lpParameter
.text:0032196F                 push    offset StartAddress ; lpStartAddress
.text:00321974                 push    0               ; dwStackSize
.text:00321976                 push    0               ; lpThreadAttributes
.text:00321978                 call    ds:CreateThread
.text:0032197E                 push    eax             ; hObject
.text:0032197F                 call    ds:CloseHandle
.text:00321985
.text:00321985 loc_321985:                             ; CODE XREF: DllMain(x,x,x)+7↑j
.text:00321985                 mov     eax, 1
.text:0032198A                 pop     ebp
.text:0032198B                 retn    0Ch
.text:0032198B _DllMain@12     endp
.text:0032198B
```

**Figura 8.** Punto de entrada con CreateThread.

La finalidad principal de esta capa es la de descifrar la capa III, utilizando para ello la contraseña que se pasa por la línea de comandos, mediante la opción "**-p**". Esta contraseña se utiliza como clave para el algoritmo de hash **HMAC-SHA256** y los datos de entrada para **HMAC-SHA256** se encuentran embebidos en el malware. En esta muestra en concreto los datos son "**kFJfhSj**". A continuación, el malware realiza 10.000 iteraciones del algoritmo **HMAC-SHA256**, combinándolo con operaciones XOR, para conseguir finalmente una clave para el algoritmo de cifrado **Rabbit**, que será utilizado para descifrar la capa III.

```
do
{
  v31[3] = v7;
  v31[0] = HIBYTE(v7);
  v31[1] = BYTE2(v7);
  v31[2] = BYTE1(v7);
  hmac_sha256_init((char *)password, ctx, password_unicode_size);
  sha256_sha224_update(data, ctx, data_unicode_size);
  sha256_sha224_update((int)v31, ctx, 4u);
  v8 = v23;
  sha256_sha224_final((int)ctx, hmac_sha256_result);
  sha256_sha224_init(ctx, v8);
  sha256_sha224_update((int)v22, v9, 0x40u);
  sha256_sha224_update((int)hmac_sha256_result, ctx, 4 * (v8 == 0) + 28);
  sha256_sha224_final((int)ctx, temp_txt);
  memmove(&rabbit_key, temp_txt, 0x20u);
  v10 = v25;
  v11 = v28;
  rabbit_key_ = rabbit_key;
  iter = 9999;
  do
  {
    hmac_sha256_init(v27, ctx, v11);
    sha256_sha224_update((int)temp_txt, ctx, 0x20u);
    v14 = v23;
    sha256_sha224_final((int)ctx, hmac_sha256_result);
    sha256_sha224_init(ctx, v14);
    sha256_sha224_update((int)v22, v15, 0x40u);
    sha256_sha224_update((int)hmac_sha256_result, ctx, 4 * (v14 == 0) + 28);
    sha256_sha224_final((int)ctx, temp_txt);
    rabbit_key_ = _mm_xor_si128(rabbit_key_, temp_txt[0]);
    rabbit_key = rabbit_key_;
    v10 = _mm_xor_si128(v10, temp_txt[1]);
    v25 = v10;
    --iter;
  }
  while ( iter );
```

**Figura 9.** Generación de la clave de descifrado del algoritmo Rabbit.

Una vez que los datos han sido descifrados, una Dll de 32 bits, es cargada y ejecutada en memoria mediante la técnica "**Reflective Load**".

# 5. CAPA III

## 5.1 DETALLES GENERALES

La muestra analizada en este apartado es una Dll de 32 bits, sin firma digital y con el siguiente hash MD5:

| NOMBRE DEL FICHERO | MD5 |
|---|---|
| - | 9876AC167941C67C0B114422689F7766 |

La fecha de compilación es 20 de noviembre de 2020, 21:50:40 (UTC), sin embargo, esta información no es del todo fiable, ya que se puede alterar fácilmente:

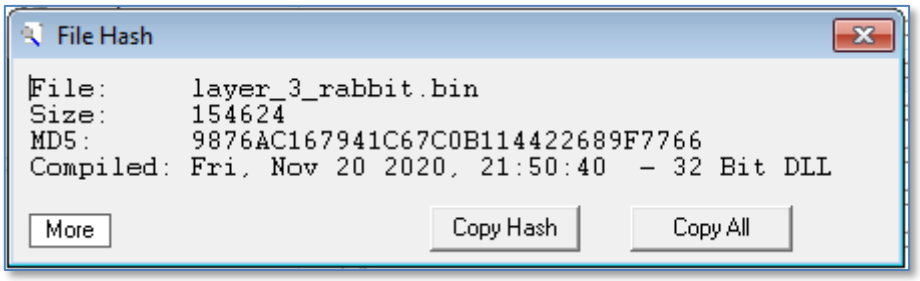


**Figura 10.** Fechas de compilación de la muestra.

La muestra no presenta propiedades del fichero.

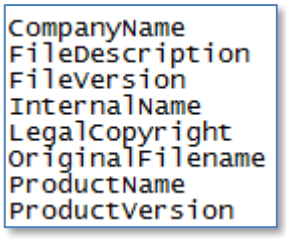


**Figura 11.** Las propiedades de los ficheros están vacías.

## 5.2 ANÁLISIS TÉCNICO

Al inicio de la capa III, el malware ejecuta un nuevo hilo, que contiene el payload principal de esta capa.

```
BOOL __stdcall DllEntryPoint(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpReserved)
{
  HANDLE v4; // eax

  if ( fdwReason == DLL_PROCESS_ATTACH )
  {
    v4 = CreateThread(0, 0, StartAddress, 0, 0, 0);
    CloseHandle(v4);
  }
  return 1;
}
```

**Figura 12.** Punto de entrada con CreateThread.

El código de la capa III está altamente ofuscado, mediante gran cantidad de saltos (jumps) y llamadas (calls) ofuscadas.

```
.text:01F725A0                  push    ebx
.text:01F725A1                  push    edi
.text:01F725A2                  push    esi
.text:01F725A3                  push    offset loc_1F729AC
.text:01F725A8                  retn
```

**Figura 13.** Ejemplo de calls ofuscadas.

Durante el inicio de la ejecución de esta capa, el malware obtiene la configuración local del equipo, con la finalidad de evitar su ejecución si el equipo está configurado en ruso o pertenece a algún país de la **Commonwealth of Independent States (CIS)**. La lista de códigos de países encontrados es la siguiente:

- 0x843 – Uzbek (Cyrillic, Uzbekistan)
- 0x443 – Uzbek (Cyrillic, Uzbekistan)
- 0x42B - Armenian
- 0x42C - Azerbaijani
- 0x437 - Georgian
- 0x419 - Russian
- 0x422 - Ukrainian
- 0x423 - Belarusian

- 0x428 – Tajik
- 0x43F – Kazakh
- 0x440 – Kyrgyz
- 0x442 – Turkmen
- 0x444 - Tatar (Russia)
- 0x818 - Romanian (Moldova)
- 0x819 - Russian (Moldova)
- 0x82C - Azerbaijani (Cyrillic, Azerbaijan)

```
.text:01F723B7 loc_1F723B7:                            ; CODE XREF: check_locales+11D↓j
.text:01F723B7                                         ; DATA XREF: check_locales+118↓o
.text:01F723B7                 cmp     eax, 422h
.text:01F723BC                 jz      short loc_1F7236D
.text:01F723BE                 push    offset loc_1F723DF
.text:01F723C3                 retn
.text:01F723C4 ; ---------------------------------------------------------------------------
.text:01F723C4
.text:01F723C4 loc_1F723C4:                            ; CODE XREF: check_locales:loc_1F71FC1↑j
.text:01F723C4                                         ; check_locales-3E1↑j
.text:01F723C4                 cmp     eax, 442h
.text:01F723C9                 jz      short loc_1F7236D
.text:01F723CB                 push    offset loc_1F724C6
.text:01F723D0                 retn
.text:01F723D1 ; ---------------------------------------------------------------------------
.text:01F723D1
.text:01F723D1 loc_1F723D1:                            ; CODE XREF: sub_1F72226+10↑j
.text:01F723D1                                         ; DATA XREF: sub_1F72226+B↑o
.text:01F723D1                 cmp     ecx, 43Fh
.text:01F723D7                 jz      short loc_1F7236D
.text:01F723D9                 push    offset sub_1F72015
.text:01F723DE                 retn
```

**Figura 14.** Comprobación del idioma por defecto.

Una vez que la configuración local es comprobada, el malware descifrará sus datos de configuración. Estos datos se encuentran localizados en la sección de datos del ejecutable (**0x10022044** en este caso). Los datos cifrados han sido ofuscados como si se tratase de un fichero "**PNG**", donde los 8 primeros bytes se corresponden con una cabecera PNG estándar. Los siguientes 8 bytes contienen el tamaño de los datos a descifrar (**0x01AB0 bytes**) y a partir del offset 0x0C se encuentran los datos cifrados. El algoritmo de cifrado es ChaCha, utilizando la siguiente clave e IV:

| CLAVE CHACHA |
| --- |
| 123223381918AE4F811114144849AA010019023201FF33048108AA0308020105 |
| **IV CHACHA** |
| 1021AA811911AA14 |

```
.data:01F92043                  db 0BCh ; ¼
.data:01F92044 Config_pointer   db  89h ; ‰          ; DATA XREF: StartAddress+912↑o
.data:01F92044                                       ; Fake PNG
.data:01F92045                  db  50h ; P
.data:01F92046                  db  4Eh ; N
.data:01F92047                  db  47h ; G
.data:01F92048                  db  0Dh
.data:01F92049                  db  0Ah
.data:01F9204A                  db  1Ah
.data:01F9204B                  db  0Ah
.data:01F9204C                  dd 1AB0h             ; Size of the configuration
.data:01F92050                  db 0A7h ; §          ; Encrypted configuration
.data:01F92051                  db 0EBh ; ë
.data:01F92052                  db 0BDh ; ½
.data:01F92053                  db  19h
.data:01F92054                  db  63h ; c
.data:01F92055                  db  72h ; r
.data:01F92056                  db 0A7h ; §
.data:01F92057                  db 0C4h ; Ä
.data:01F92058                  db 0DAh ; Ú
.data:01F92059                  db  33h ; 3
.data:01F9205A                  db  0Fh
.data:01F9205B                  db 0CCh ; Ì
.data:01F9205C                  db 0DFh ; ß
.data:01F9205D                  db  46h ; F
.data:01F9205E                  db 0D6h ; Ö
.data:01F9205F                  db    8
.data:01F92060                  db  9Ch ; œ
.data:01F92061                  db 0E2h ; â
```

**Figura 15.** Datos de configuración cifrados.

El bloque de configuración contiene los siguientes datos:

- La plantilla de la nota de rescate.
- Lista de procesos a terminar.
- Palabras clave para la búsqueda de servicios a terminar.
- Una clave pública RSA de 2048-bits, utilizada durante el proceso de cifrado de los ficheros.
- Flag para indicar si el fichero de configuración contiene alguna lista de servidores de C2 (en esta muestra no hay ninguno).

| CLAVE RSA PÚBLICA |
| --- |
| -----BEGIN PUBLIC KEY-----<br>MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA2HC4F1u4kVqVaiV1gJOS<br>Y5+As7J61rscUtGVW2wmCmrf9S84GNkjlihVdxYfvgYM/68aXEGis3pzBSIOqfzI<br>grCttIexPXh2BOIMBk4ZRjOK7RXdUhYu/rqKbQhtACXTnEAcg5xPot9wkqOA0/KR<br>eSNfUpsQkvkw18yGNd5CBQLeOMOI6KoQJZHHE2OoAJ6rGD8wd4ggGukZppHttZ3A<br>mK5tBK9vHSfnj2kiCsVacPD0C7d2AStMn+l/XksuH4dxm58zp9pl6k+O3P8x6q6M<br>FGQNmUJwhc6vjlC+EouK/J4woE1+w3Yg278GlpdOX64jbcs3oag6AtUEYdTHHaKn<br>/wIDAQAB<br>-----END PUBLIC KEY----- |

Esta capa acepta una lista amplia de parámetros, que se encuentran cifrados mediante “rolling XOR”. La lista de parámetros es la siguiente:

- **--fast** –Limitar el tamaño de los ficheros a cifrar. El valor aceptado debe de ser expresado en megabytes.
- **--full** – Cifrar todos los ficheros locales y remotos (carpetas compartidas de red).
- **--multiproc** – Evita la creación del mutex.
- **--nonet** – No cifrar ficheros en carpetas compartidas de red.
- **--nomimikatz** – Relacionado con mimikatz.
- **--path** – Permite especificar la ruta o carpeta que se quiere cifrar.
- **--target** – Permite especificar la extensión de los ficheros que se quieren cifrar.
- **--append** – Permite especificar la extensión que se añadirá a los ficheros cifrados. En caso contrario se utilizará un valor aleatorio de entre 4 y 6 caracteres.
- **--norename** – No renombrar los ficheros cifrados.
- **--greetings** – Permite añadir un nombre al mensaje de rescate, lo que permitiría personalizar la nota de rescate para la víctima del ataque.
- **--samba** – Desconocido.
- **--killrdp** – Terminar el servicio asociado a terminal remoto (RDP).

```
if ( a1 || !a1 )
{
  if ( ptr_to_decrypted_config )
  {
    decrypt_data(&cli_fast, (int)&switch_fast);
    decrypt_data(&cli_full, (int)&switch_full);
    decrypt_data(&cli_multiproc, (int)&switch_multiproc);
    decrypt_data(&cli_nonet, (int)&switch_nonet);
    decrypt_data(&cli_path, (int)switch_path_);
    switch_path_ = &switch_path;
    decrypt_data(&cli_nomimikatz, (int)&switch_nomimikatz);
    decrypt_data(&cli_target, (int)&switch_target);
    decrypt_data(&cli_append, (int)&switch_append);
    decrypt_data(&cli_norename, (int)&switch_norename);
    JUMPOUT(0x1F75129);
```

**Figura 16.** Descifrado de parámetros de línea de comandos.

El malware contiene una lista de ficheros y directorios que serán excluidos del proceso de cifrado:

- Los siguientes ficheros: **autorun.inf, boot.ini, desktop.ini, ntuser.dat, iconcache.db, bootsect.bak, ntuser.dat.log, thumbs.db, Bootfont.bin** y **RECOVER-FILES.txt**
- Los siguientes directorios: **Windows, Program Files, Tor Browser, ProgramData, \cache2\entries, \Low\Content.IE5\, \User Data\Default\Cache** y **All Users**.
- Las siguientes extensiones: **lnk, exe, sys** y **dll**.

El malware obtiene diferente información del equipo (nombre del equipo, nombre de usuario, nombre de dominio, nombre del sistema operativo, volume serial number) que se utilizarán para generar identificadores únicos mediante CRC32 utilizados posteriormente para crear el mutex, los ficheros LNK, etc… A su vez, el malware utiliza WMI para identificar los programas antivirus mediante la llamada "**SELECT * FROM AntiVirusProducts**", que se encuentran registrados en **root\SecurityCenter2**.

```
if ( v5 )
{
  v10 = v5;
  GetComputerNameW(computer_name[1], a4);
}
v12 = &sub_1F7916F;
```

**Figura 17.** Obtención del nombre del equipo.

El malware utiliza las siguientes funciones del API de Windows: **GetLogicalDriveStrings** y **GetDiskFreeSpace** para identificar los dispositivos locales conectados al equipo, así como el espacio libre disponible en cada uno de ellos. Seguidamente, crea el mutex con el siguiente formato: Global\[UNIQUEID], donde UNIQUEID es la unión del "**Volume Serial Number**" y del **CRC32** del nombre del equipo en formato unicode. Por ejemplo: "Global\427DA44969A98A3A", donde 427DA449 es el "Volume Serial Number" y 69A98A3A es el CRC32 del nombre del equipo en unicode.

```
BOOL __stdcall create_mutex(int a1)
{
  BOOL v1; // esi
  DWORD v2; // eax
  const WCHAR *v3; // eax
  int mutex_format_Global_x_; // [esp-4h] [ebp-114h] BYREF
  _DWORD v6[2]; // [esp+0h] [ebp-110h] BYREF
  WCHAR unique_id[132]; // [esp+8h] [ebp-108h] BYREF

  v1 = 1;
  if ( !*((_DWORD *)dword_1F94CC4 + 4) )
  {
    decrypt_data(&mutex_format_Global_x_, (int)&mutex_format_Global_x);
    if ( sub_1F74100(&mutex_format_Global_x_) )
    {
      v3 = (const WCHAR *)sub_1F74100(v6);
      wsprintfW(unique_id, v3, a1);
      CreateMutexW(0, 0, unique_id);
      v2 = GetLastError();
      v1 = v2 != 5 && v2 != 183;
    }
    else
    {
      v1 = 1;
    }
    clean_memory(v6);
  }
  return v1;
}
```

**Figura 18.** Creación del mutex.

El malware creará un fichero LNK en cada directorio donde se han cifrado datos. El nombre del fichero LNK es el CRC32 generado a partir del nombre de dominio del equipo en unicode. Por ejemplo, en el caso de un equipo aislado, el dominio será "**WORKGROUP**", cuyo CRC32, usando "**WORKGROUP**" en unicode, será **e6189640**. Estos ficheros serán eliminados una vez que el proceso de cifrado finaliza.

Durante su ejecución, el malware obtiene el directorio **%ALLUSERSPROFILE%** (generalmente "**C:\ProgramData**"), donde creará el fichero "**dtb.dat**". Este fichero es

utilizado para almacenar las claves RSA de sesión utilizadas durante el proceso de cifrado de los ficheros. Antes de iniciar el proceso de cifrado, el malware genera una clave RSA pública y privada, que será utilizada para cifrar la clave simétrica usada para cifrar los ficheros del equipo. El proceso de generación y almacenamiento de estas claves RSA es el siguiente:

- Utiliza **CryptGenKey** para generar una clave de sesión RSA de **2048 bits** (pública y privada)
- La clave pública y privada son exportadas mediante la llamada del API **CryptExportKey**.
- La clave privada exportada es cifrada mediante ChaCha, con una clave de 0x20 bytes y un IV de 0x08 bytes, generados aleatoriamente mediante **CryptGenRandom**.
- La clave ChaCha es cifrada mediante **CryptEncrypt**, utilizando la clave pública extraída de los datos de configuración del malware.
- La clave ChaCha e IV cifrados y la clave privada de sesión cifrada son guardadas en el fichero **%ProgramData%\dtb.dat**, junto con la clave pública de sesión (sin cifrar)

```
v32 = (*(int (**)(void))(v30 + 4))();       // Call to crypt_import_key.
                                            // Import public key from config data
if ( v31 || !v31 )
{
  if ( !v32 )
  {
    ((void (__thiscall *)(BYTE *, int, _DWORD *, int *))nullsub_71)(pub_key_config, priv_key, a6, a7);
    return;
  }
  while ( 1 )
  {
    v26 = *(_DWORD *)pub_key_config;
    v46 = pub_key_config;
    v24 = (DWORD *)(*(int (__thiscall **)(BYTE *, unsigned int))(v26 + 12))(pub_key_config, v43);
    if ( v23 || !v23 )
    {
      pub_key_config = v47;
      v48 = (int)(v47 + 1172);
      v42 = (int)(v47 + 1172);
      v41 = 32;
      pub_key_config_ = v24;
    }
    v37 = CryptGenRandom((HCRYPTPROV)v24, v41, (BYTE *)v42);
    v20 = !v37;
    if ( !v37 )
      break;
    v29 = pub_key_config + 1428 == 0;
    pub_key_config += 1428;
    if ( pub_key_config )
    {
      ((void (__stdcall *)(int, _DWORD *, int *))nullsub_74)(priv_key, a6, a7);
      return;
    }
    v42 = 0;
    if ( v29 || !v29 )
    {
      v41 = 8;
      if ( v29 || !v29 )
      {
        buffer_0x694 = (_DWORD *)CryptGenRandom((HCRYPTPROV)pub_key_config_, v41, (BYTE *)v42);
        goto LABEL_40;
      }
      a3_buffer_0x694 = a6;
      goto LABEL_1;
```

**Figura 19.** Creación de clave e IV ChaCha aleatorias.

```
  }
  else
  {
    if ( v31 )
    {
      ((void (__stdcall *)(int, _DWORD *, int *))sub_1F76EB9)(priv_key, a6, a7);
      return;
    }
    if ( !v31 )
    {
      if ( *(_BYTE *)(v32 + 248621) )
      {
        *v33 = *(_DWORD *)pub_key_config;
        JUMPOUT(0x1F7679F);
      }
      JUMPOUT(0x1F76C5A);
    }
    chacha20_enc_dec((int)ctx, priv_key_1, chacha_key_iv, v43);
    v36 = *(_DWORD *)v28;
    v48 = 32;
    v43 = 0;
    priv_key_1 = 256;
    v40 = &v48;
    if ( ctx && !ctx )
    {
      if ( chacha_key_iv )
        ((void (__stdcall *)(int, _DWORD *, int *))nullsub_70)(priv_key, a6, a7);
      return;
    }
    if ( (*(int (__cdecl **)(DWORD *, int *, DWORD))(v36 + 16))(pub_key_config_, v40, priv_key_1) )
    {
```

**Figura 20.** Cifrado de clave privada mediante ChaCha.

La estructura del fichero **dtb.dat** es la siguiente:

| Offset | Descripción |
|---|---|
| **0x00-0x01** | 0xAAFF |
| **0x02-0x05** | Checksum de la clave RSA pública generada en el equipo (clave de sesión) |
| **0x06-0x499** | Clave RSA privada (clave de sesión), generada en el equipo y cifrada con ChaCha |
| **0x49A-0x599** | Clave ChaCha cifrada con la clave pública RSA embebida en la configuración del malware |
| **0x59A-0x699** | IV de ChaCha cifrada con la clave pública RSA embebida en la configuración del malware |
| **0x69A-0x7AE** | Clave RSA pública generada en el equipo (clave de sesión) |

```
int __usercall get_crypt_rsa_key@<eax>(HCRYPTKEY **a1@<ebx>, DWORD dwFlags, DWORD pdwDataLen, DW
{
  BOOL v11; // eax
  bool v12; // zf
  HCRYPTKEY *v13; // eax
  DWORD v15; // [esp+10h] [ebp-10h]
  DWORD v16; // [esp+14h] [ebp-Ch]
  ALG_ID v17; // [esp+14h] [ebp-Ch]
  BYTE *v18; // [esp+18h] [ebp-8h]
  HCRYPTKEY retaddr; // [esp+20h] [ebp+0h] BYREF

  if ( !a11 )
    goto LABEL_20;
  v11 = CryptAcquireContextW(
          &dwBlobType,
          0,
          L"Microsoft Enhanced Cryptographic Provider v1.0",
          PROV_RSA_FULL,
          0xF0000000);
  v12 = !v11;
  if ( !v11 )
LABEL_20:
    JUMPOUT(0x1F77677);
  v13 = &retaddr;
  retaddr = 0;
  if ( v12 || !v12 )
  {
    if ( !CryptGenKey(dwBlobType, v17, 0xA400u, (HCRYPTKEY *)0x8000001) )// CALG_RSA_KEYX
      goto LABEL_21;
    pdwDataLen = 0;
    if ( !CryptExportKey(retaddr, 0, v15, v16, v18, (DWORD *)6) )// PUBLICKEYBLOB
LABEL_21:
      JUMPOUT(0x1F7729A);
    a1[1] = 0;
    v13 = (HCRYPTKEY *)j_wrapper_VirtualAlloc(1u);
    v12 = v13 == 0;
  }
  *a1 = v13;
  if ( v12 )
    goto LABEL_21;
  return sub_1F775F3(dwFlags, pdwDataLen);
}
```

**Figura 21.** Creación de claves RSA.

El malware tiene que componer la nota de rescate, personalizándola para cada infección. La nota de rescate contiene 3 variables: **%Greetings2target%**, **%id%** y **%egregor_data%**, que serán sustituidas de la siguiente forma:

- **%id%** - es reemplazado por el checksum del nombre del equipo.
- **%egregor_data%** - es reemplazado por los datos, en base64, de los 3 primeros bloques de datos almacenados en el fichero dtb.dat: clave de sesión RSA privada cifrada, clave ChaCha cifrada e IV de ChaCha cifrada.
- **% Greetings2target%** - es reemplazado por los datos que se pasan por la línea de comandos, en el parámetro "**--greetings**".

En el apéndice I se ha incluido la plantilla de la nota de rescate utilizada por el malware, donde pueden observarse las variables explicadas anteriormente.

El siguiente paso es lanzar un nuevo hilo, que será utilizado para buscar y detener los procesos y servicios configurados en la configuración del malware. Para ello detendrá cualquier servicio que contenga alguna de las siguientes cadenas de texto: **sql, database** y **msexchange** y, además, detendrá cualquier proceso que coincida con los procesos de la configuración del código dañino (Apéndice II). El malware utiliza **NtQuerySystemInformation** para enumerar la lista de procesos en el sistema y **NtTerminateProcess** para matar los procesos. A su vez, el malware eliminará las "**shadow copies**" mediante **WMI**.

```
.data:01F94576                    db  2Ah ; *
.data:01F94577                    db  58h ; X
.data:01F94578                    db  85h ; …
.data:01F94579                    db  9Ch ; œ
.data:01F9457A select_from_win32_ShadowCopy db 0DBh ; Û
.data:01F9457A                                        ; DATA XREF: sub_1F838EE↑o
.data:01F9457B                    db 0D3h ; Ó
.data:01F9457C                    db 0CDh ; Í
.data:01F9457D                    db  84h ; „
.data:01F9457E                    db  5Bh ; [
```

**Figura 22.** Eliminación de shadow copies mediante WMI.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 73 | 71 | 6C | 3B | 64 | 61 | 74 | 61 | 62 | 61 | 73 | 65 | 3B | 6D | 73 | 65 | sql;database;mse |
| 78 | 63 | 68 | 61 | 6E | 67 | 65 | 3B | 6D | 73 | 66 | 74 | 65 | 73 | 71 | 6C | xchange;msftesql |
| 2E | 65 | 78 | 65 | 3B | 73 | 71 | 6C | 61 | 67 | 65 | 6E | 74 | 2E | 65 | 78 | .exe;sqlagent.ex |
| 65 | 3B | 73 | 71 | 6C | 62 | 72 | 6F | 77 | 73 | 65 | 72 | 2E | 65 | 78 | 65 | e;sqlbrowser.exe |
| 3B | 73 | 71 | 6C | 77 | 72 | 69 | 74 | 65 | 72 | 2E | 65 | 78 | 65 | 3B | 6F | ;sqlwriter.exe;o |
| 72 | 61 | 63 | 6C | 65 | 2E | 65 | 78 | 65 | 3B | 6F | 63 | 73 | 73 | 64 | 2E | racle.exe;ocssd. |
| 65 | 78 | 65 | 3B | 64 | 62 | 73 | 6E | 6D | 70 | 2E | 65 | 78 | 65 | 3B | 73 | exe;dbsnmp.exe;s |
| 79 | 6E | 63 | 74 | 69 | 6D | 65 | 2E | 65 | 78 | 65 | 3B | 61 | 67 | 6E | 74 | ynctime.exe;agnt |
| 73 | 76 | 63 | 2E | 65 | 78 | 65 | 3B | 69 | 73 | 71 | 6C | 70 | 6C | 75 | 73 | svc.exe;isqlplus |
| 73 | 76 | 63 | 2E | 65 | 78 | 65 | 3B | 78 | 66 | 73 | 73 | 76 | 63 | 63 | 6F | svc.exe;xfssvcco |
| 6E | 2E | 65 | 78 | 65 | 3B | 73 | 71 | 6C | 73 | 65 | 72 | 76 | 72 | 2E | 65 | n.exe;sqlservr.e |
| 78 | 65 | 3B | 6D | 79 | 64 | 65 | 73 | 6B | 74 | 6F | 70 | 73 | 65 | 72 | 76 | xe;mydesktopserv |
| 69 | 63 | 65 | 2E | 65 | 78 | 65 | 3B | 6F | 63 | 61 | 75 | 74 | 6F | 75 | 70 | ice.exe;ocautoup |
| 64 | 73 | 2E | 65 | 78 | 65 | 3B | 65 | 6E | 63 | 73 | 76 | 63 | 2E | 65 | 78 | ds.exe;encsvc.ex |
| 65 | 3B | 66 | 69 | 72 | 65 | 66 | 6F | 78 | 63 | 6F | 6E | 66 | 69 | 67 | 2E | e;firefoxconfig. |
| 65 | 78 | 65 | 3B | 74 | 62 | 69 | 72 | 64 | 63 | 6F | 6E | 66 | 69 | 67 | 2E | exe;tbirdconfig. |
| 65 | 78 | 65 | 3B | 6D | 79 | 64 | 65 | 73 | 6B | 74 | 6F | 70 | 71 | 6F | 73 | exe;mydesktopqos |
| 2E | 65 | 78 | 65 | 3B | 6F | 63 | 6F | 6D | 6D | 2E | 65 | 78 | 65 | 3B | 6D | .exe;ocomm.exe;m |
| 79 | 73 | 71 | 6C | 64 | 2E | 65 | 78 | 65 | 3B | 6D | 79 | 73 | 71 | 6C | 64 | ysqld.exe;mysqld |
| 2D | 6E | 74 | 2E | 65 | 78 | 65 | 3B | 6D | 79 | 73 | 71 | 6C | 64 | 2D | 6F | -nt.exe;mysqld-o |
| 70 | 74 | 2E | 65 | 78 | 65 | 3B | 64 | 62 | 65 | 6E | 67 | 35 | 30 | 2E | 65 | pt.exe;dbeng50.e |
| 78 | 65 | 3B | 73 | 71 | 62 | 63 | 6F | 72 | 65 | 73 | 65 | 72 | 76 | 69 | 63 | xe;sqbcoreservic |
| 65 | 2E | 65 | 78 | 65 | 3B | 65 | 78 | 63 | 65 | 6C | 2E | 65 | 78 | 65 | 3B | e.exe;excel.exe; |
| 69 | 6E | 66 | 6F | 70 | 61 | 74 | 68 | 2E | 65 | 78 | 65 | 3B | 6D | 73 | 61 | infopath.exe;msa |
| 63 | 63 | 65 | 73 | 73 | 2E | 65 | 78 | 65 | 3B | 6D | 73 | 70 | 75 | 62 | 2E | ccess.exe;mspub. |
| 65 | 78 | 65 | 3B | 6F | 6E | 65 | 6E | 6F | 74 | 65 | 2E | 65 | 78 | 65 | 3B | exe;onenote.exe; |
| 6F | 75 | 74 | 6C | 6F | 6F | 6B | 2E | 65 | 78 | 65 | 3B | 70 | 6F | 77 | 65 | outlook.exe;powe |
| 72 | 70 | 6E | 74 | 2E | 65 | 78 | 65 | 3B | 73 | 71 | 6C | 73 | 65 | 72 | 76 | rpnt.exe;sqlserv |
| 72 | 2E | 65 | 78 | 65 | 3B | 74 | 68 | 65 | 62 | 61 | 74 | 2E | 65 | 78 | 65 | r.exe;thebat.exe |
| 3B | 73 | 74 | 65 | 61 | 6D | 2E | 65 | 78 | 65 | 3B | 74 | 68 | 65 | 62 | 61 | ;steam.exe;theba |
| 74 | 36 | 34 | 2E | 65 | 78 | 65 | 3B | 74 | 68 | 75 | 6E | 64 | 65 | 72 | 62 | t64.exe;thunderb |
| 69 | 72 | 64 | 2E | 65 | 78 | 65 | 3B | 76 | 69 | 73 | 69 | 6F | 2E | 65 | 78 | ird.exe;visio.ex |
| 65 | 3B | 77 | 69 | 6E | 77 | 6F | 72 | 64 | 2E | 65 | 78 | 65 | 3B | 77 | 6F | e;winword.exe;wo |
| 72 | 64 | 70 | 61 | 64 | 2E | 65 | 78 | 65 | 3B | 51 | 42 | 57 | 33 | 32 | 2E | rdpad.exe;QBW32. |
| 65 | 78 | 65 | 3B | 51 | 42 | 57 | 36 | 34 | 2E | 65 | 78 | 65 | 3B | 69 | 70 | exe;QBW64.exe;ip |
| 79 | 74 | 68 | 6F | 6E | 2E | 65 | 78 | 65 | 3B | 77 | 70 | 79 | 74 | 68 | 6F | ython.exe;wpytho |
| 6E | 2E | 65 | 78 | 65 | 3B | 70 | 79 | 74 | 68 | 6F | 6E | 2E | 65 | 78 | 65 | n.exe;python.exe |
| 3B | 64 | 75 | 6D | 70 | 63 | 61 | 70 | 2E | 65 | 78 | 65 | 3B | 70 | 72 | 6F | ;dumpcap.exe;pro |
| 63 | 6D | 6F | 6E | 2E | 65 | 78 | 65 | 3B | 70 | 72 | 6F | 63 | 6D | 6F | 6E | cmon.exe;procmon |
| 36 | 34 | 2E | 65 | 78 | 65 | 3B | 70 | 72 | 6F | 63 | 65 | 78 | 70 | 2E | 65 | 64.exe;procexp.e |
| 78 | 65 | 3B | 70 | 72 | 6F | 63 | 65 | 78 | 70 | 36 | 34 | 2E | 65 | 78 | 65 | xe;procexp64.exe |

**Figura 23.** Lista de procesos y servicios a detener.

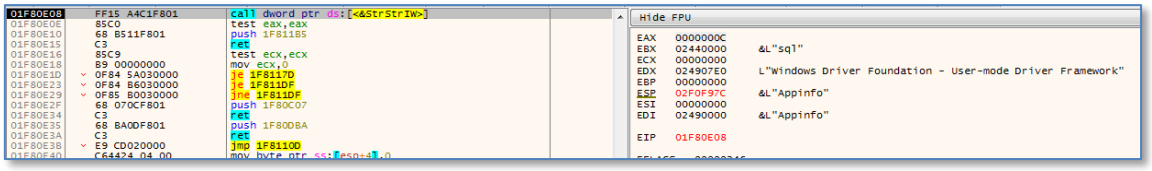


**Figura 24.** Búsqueda de servicios de interés.

Por cada fichero que es cifrado, el malware genera una nueva clave ChaCha de 256 bits y un IV de 64 bits. El proceso de cifrado es el siguiente:

- Generar una clave ChaCha e IV aleatorios, mediante **CrypGenRandom**.
- Cifrar el contenido del fichero mediante ChaCha y la clave e IV generados en el punto anterior.
- Cifrar la clave ChaCha y el IV mediante la clave pública RSA de sesión (guardad en el fichero **dtb.dat** sin cifrar).
- Añadir la clave ChaCha e IV cifrados al final del fichero cifrado.
- Añadir 16 bytes al final del fichero cifrado:
  - 4 bytes aleatorios.
  - 4 bytes aleatorios, los mismos que antes, pero cifrados mediante XOR y la clave **0xB16B00B5**.
  - Renombrar el fichero, añadiendo una extensión aleatoria, de entre 4 y 6 caracteres.

```
v12 = (*(int (__thiscall **)(int, char, char, char, char))(*(_DWORD *)a4 + 12))(a4, v31, v32, v33, v34);
if ( !CryptGenRandom(v12, 0x20u, pbBuffer) )
  goto LABEL_28;
if ( !CryptGenRandom(v12, 8u, v99) )
  goto LABEL_28;
chacha20_Init_Key(&v45, pbBuffer, 256);
chacha20_Init_IV(&v45, v99);
Buffer.LowPart = 40;
if ( !(*(int (__thiscall **)(int, BYTE *, LARGE_INTEGER *, int, _DWORD, _DWORD))(*(_DWORD *)a4 + 16))(
        a4,
        pbBuffer,
        &Buffer,
        256,
        0,
        0) )
{
LABEL_28:
  v10 = 0;
  v9 = 0;
  goto LABEL_52;
}
v44 = GetTickCount();
v102 = sub_1F73AA0(&v44);
v103 = v102 ^ 0xB16B00B5;
v101 = *((_DWORD *)dword_1F94CC4 + 7);
HIDWORD(v100) = 0;
if ( v35 && __SPAIR64__(v35 >> 12, v35 << 20) < v8->QuadPart )
  HIDWORD(v100) = v35;
LOBYTE(v96[0]) = 0;
Buffer.QuadPart = 0i64;
if ( SetFilePointerEx(hFile, 0i64, 0, 2u) )
  v36 = !WriteFile(hFile, pbBuffer, 0x110u, NumberOfBytesWritten, 0);
else
  v36 = 1;
*(_QWORD *)NumberOfBytesWritten = LOBYTE(v96[0]);
SetFilePointerEx(hFile, (LARGE_INTEGER)LOBYTE(v96[0]), 0, 0);
v9 = 0;
if ( !v36 )
{
```

**Figura 25.** Función de cifrado de ficheros.

| CLAVE RSA PÚBLICA |
| --- |
| -----BEGIN PUBLIC KEY-----<br>MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA2HC4F1u4kVqVaiV1gJOS<br>Y5+As7J61rscUtGVW2wmCmrf9S84GNkjlihVdxYfvgYM/68aXEGis3pzBSIOqfzI<br>grCttIexPXh2BOIMBk4ZRjOK7RXdUhYu/rqKbQhtACXTnEAcg5xPot9wkqOA0/KR<br>eSNfUpsQkvkw18yGNd5CBQLeOMOI6KoQJZHHE2OoAJ6rGD8wd4ggGukZppHttZ3A<br>mK5tBK9vHSfnj2kiCsVacPD0C7d2AStMn+l/XksuH4dxm58zp9pl6k+O3P8x6q6M<br>FGQNmUJwhc6vjlC+EouK/J4woE1+w3Yg278GlpdOX64jbcs3oag6AtUEYdTHHaKn<br>/wIDAQAB<br>-----END PUBLIC KEY----- |

## 6. PERSISTENCIA

Esta muestra de malware no está configurada para instalar persistencia en el sistema, ya que una vez que finaliza el cifrado de todos los ficheros, no necesita volver a ejecutarse.

## 7. YARA

La siguiente regla de YARA puede utilizarse para detectar el malware en un equipo infectado.

**RAGNAR**

```
import "pe"
import "hash"

rule ransomware_egregor {

   meta:
      description = "Egregor"
      author = "CCN-CERT"
      date = "2021-01-09"
      hash = "C4735B8F2A4DE4C9640CCC605602B6C6"

   strings:
      $p1 = "qqqqq.pdb" fullword ascii

      $s1 = "C:\\Heil Egregor\\1488\\ficker.py" fullword wide
      $s2 = "dubisteinmutterficker" fullword wide

   condition:
      uint16(0) == 0x5a4d and filesize < 1000KB and
      hash.sha256(pe.rich_signature.clear_data) ==
"38b155b6546db882189cc79bcac0b0284d3f858e0feb1e5dbc24b22f78cdf
b68" or
      (pe.sections[4].name == ".gfids") and
      ($p1 or all of ($s*))
}
```

## 8. IOCS

Los siguientes IOCs pueden ser utilizados para detectar equipos infectados con este malware.

| | IOCs |
|---|---|
| **MD5** | C4735B8F2A4DE4C9640CCC605602B6C6 |
| **MD5** | 4FAFE7F3E3FF6E672DEEEC5AF3351EA7 |
| **MD5** | 9876AC167941C67C0B114422689F7766 |
| **Nombre de fichero** | RECOVER-FILES.txt |

# 9. APÉNDICE I

El malware presenta el siguiente mensaje de rescate.

**MENSAJE DE RESCATE**

```
%Greetings2target%
------------------
| What happened? |
------------------

Your network was ATTACKED, your computers and servers were LOCKED,
Your private data was DOWNLOADED.

----------------------
| What does it mean? |
----------------------

It means that soon mass media, your partners and clients WILL KNOW about your
PROBLEM.

--------------------------
| How it can be avoided? |
--------------------------

In order to avoid this issue,
you are to COME IN TOUCH WITH US no later than within 3 DAYS and conclude the
data recovery and breach fixing AGREEMENT.

-------------------------------------------
| What if I do not contact you in 3 days? |
-------------------------------------------

If you do not contact us in the next 3 DAYS we will begin DATA publication.

----------------------------
| I can handle it by myself |
----------------------------

It is your RIGHT, but in this case all your data will be published for public
USAGE.

-------------------------------
| I do not fear your threats! |
-------------------------------

That is not the threat, but the algorithm of our actions.
If you have hundreds of millions of UNWANTED dollars, there is nothing to
FEAR for you.
That is the EXACT AMOUNT of money you will spend for recovery and payouts
because of PUBLICATION.

--------------------------
| You have convinced me! |
--------------------------

Then you need to CONTACT US, there is few ways to DO that.

I. Recommended (the most secure method)

   a) Download a special TOR browser: https://www.torproject.org/
   b) Install the TOR browser
   c) Open our website with LIVE CHAT in the TOR browser:
http://egregor4u5ipdzhv.onion/%id%
   d) Follow the instructions on this page.

II. If the first method is not suitable for you
```

```
   a) Open our website with LIVE CHAT: https://egregor-support.com/%id%
   b) Follow the instructions on this page.

Our LIVE SUPPORT is ready to ASSIST YOU on this website.

----------------------------------------
| What will I get in case of agreement |
----------------------------------------

You WILL GET full DECRYPTION of your machines in the network, FULL FILE
LISTING of downloaded data,
confirmation of downloaded data DELETION from our servers, RECOMMENDATIONS
for securing your network perimeter.

And the FULL CONFIDENTIALITY ABOUT INCIDENT.

------------------------------------------------------------------------------
-----
Do not redact this special technical block, we need this to authorize you.
---EGREGOR---
%egregor_data%
---EGREGOR---
```

# 10. APÉNDICE II

Lista de procesos a detener.

| PROCESOS A TERMINAR |
| --- |
| msftesql.exe;sqlagent.exe;sqlbrowser.exe;sqlwriter.exe;oracle.exe;ocssd.exe;dbsnmp.exe;synctime.exe;agntsvc.exe;isqlplussvc.exe;xfssvccon.exe;sqlservr.exe;mydesktopservice.exe;ocautoupds.exe;encsvc.exe;firefoxconfig.exe;tbirdconfig.exe;mydesktopqos.exe;ocomm.exe;mysqld.exe;mysqld-nt.exe;mysqld-opt.exe;dbeng50.exe;sqbcoreservice.exe;excel.exe;infopath.exe;msaccess.exe;mspub.exe;onenote.exe;outlook.exe;powerpnt.exe;sqlservr.exe;thebat.exe;steam.exe;thebat64.exe;thunderbird.exe;visio.exe;winword.exe;wordpad.exe;QBW32.exe;QBW64.exe;ipython.exe;wpython.exe;python.exe;dumpcap.exe;procmon.exe;procmon64.exe;procexp.exe;procexp64.exe |