Informe Código Dañino
CCN-CERT ID-09/22

Vice Society - Neshta

Agosto 2022

Edita:

GOBIERNO DE ESPAÑA MINISTERIO DE DEFENSA

CCN centro criptológico nacional

CENTRO CRIPTOLOGICO NACIONAL
cn=CENTRO CRIPTOLOGICO NACIONAL, 2.5.4.97=VATES-S2800155J, ou=CENTRO CRIPTOLOGICO NACIONAL, o=CENTRO CRIPTOLOGICO NACIONAL, c=ES
2022.08.30 16:16:19 +02'00'

© Centro Criptológico Nacional, 2022

Fecha de Edición: agosto de 2022

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 4
2. INFORMACIÓN DEL CÓDIGO DAÑINO 5
3. CARACTERÍSTICAS DEL CÓDIGO DAÑINO 5
4. DETALLES GENERALES 6
4.1 FLUJO DE EJECUCIÓN 6
4.2 PERSISTENCIA 9
5. CARACTERÍSTICAS TÉCNICAS 10
5.1 FUNCIONAMIENTO GENERAL 10
5.2 TÉCNICAS ANTI ANÁLISIS 13
5.2.1 CIFRADO DE CADENAS 13
5.2.2 RESOLUCIÓN DE APIS 13
5.3 ESQUEMA DE CIFRADO 14
5.3.1 CIFRADO DE FICHEROS 15
6. MENSAJE DE RESCATE 16
7. INDICADORES DE COMPROMISO 17
8. DESINFECCIÓN 18
9. MITIGACIÓN 18
10. REGLAS DE DETECCIÓN 19
10.1 REGLAS YARA 19
ANEXO 20
LISTA DE EXTENSIONES EXCLUIDAS DE SER CIFRADAS 20
LISTA DE CARPETAS EXCLUIDAS DE SER CIFRADAS 20
LISTA DE APIS RESULTAS DINÁMICAMENTE 21
MENSAJE DE RESCATE 21

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 ciber amenazas, 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 ciber incidentes que afecten a cualquier organismo o empresa pública. En el caso de operadores críticos del sector público la gestión de ciber incidentes se realizará por el CCN-CERT en coordinación con el CNPIC.

2. INFORMACIÓN DEL CÓDIGO DAÑINO

El presente documento recoge el análisis de una muestra de ransomware que pertenece al grupo ***VICE SOCIETY*** infectada por el virus ***Neshta***.

Muestra	Hash SHA-1
Ransomware infectado por Neshta	AFF4E088DE7E6630C1877B046362B01DBD549622
Ransomware	244B8F0593A6A01BDF608A33D94A0A6B092C958A

VICE SOCIETY, también conocido como ***VICE SPIDER,*** empezó a tener actividad en abril de 2021. Este actor utiliza diferentes familias de ransomware como Zeppelin, **Spider**, Death Kitty para versión de Linux y Hive como RaaS (Ransomware as a service) para cifrar los archivos de sus víctimas. El vector de entrada que suele utilizar este actor es la utilización de credenciales legítimas comprometidas de servicios de VPN o RDP y así acceder a la red de las víctimas.

Neshta es un código dañino del tipo *file infector*. Este tipo de código dañino tiene las capacidades de autorreplicarse en otros ejecutables (exe y dll) del sistema, es decir, inyecta su propio código en otros ejecutables, de tal forma que hace que su código se ejecute antes y después de ejecutar el código original. El código que se inyecta, por lo general, se ejecuta cada vez que se abre un archivo "exe" y se copia en otros ficheros del mismo tipo, replicándose a sí mismo en más ficheros. En otros casos, también pueden añadir algún código dañino que pueda ser utilizado para otros fines como por ejemplo el robo de datos o credenciales. En el caso de esta muestra, solo se ha observado el comportamiento de autorreplicado.

Por lo tanto, este documento explica los códigos dañinos de ambas familias.

3. CARACTERÍSTICAS DEL CÓDIGO DAÑINO

El código dañino examinado posee las siguientes características:

- Es compatible con sistemas Windows de 32 y 64 bits.
- Resuelve APIs de forma dinámica.
- Emplea técnicas para dificultar su detección de forma estática.
- Infecta los ficheros ejecutables del equipo afectado.
- Cifra los ficheros del equipo afectado.
- Cifra ficheros utilizando algoritmos de cifrado simétrico y asimétrico.
- Crea un mensaje de rescate en el escritorio del usuario.
- No requiere de conexión a Internet.
- Crea persistencia en el sistema.

4. DETALLES GENERALES

La muestra analizada es un fichero Portable Executable (PE) que se corresponde con un archivo de código ejecutable para sistemas operativos Windows. En este caso el código que contiene el fichero es de 32 bits, lo cual también lo hace compatible para sistemas de 32 y 64 bits.

La muestra está infectada con el virus ***Neshta***, escrito en el entorno de desarrollo Delphi. Este virus infecta archivos PE y los modifica de tal forma que hace que primero se ejecute el código del virus para después ejecutar el archivo PE original, en este caso el ransomware.

Es por eso que la cabecera PE de la muestra analizada tiene la siguiente fecha de compilación.

Sections Count	8	8
Time Date Stamp	2a425e19	viernes, 19.06.1992 22:22:17 UTC
Ptr to Symbol Table	0	0

Figura 1. Información del código dañino Neshta.

Esta fecha de compilación (TimeDateStamp) indica que la muestra fue creada el 19/06/1992. Sin embargo, esto es un dato que no hay que tener en cuenta ya que se trata de la fecha de compilación que pone por defecto el entorno de desarrollo Delphi que se corresponde con el virus Neshta y no la del ransomware.

Una vez que el código del virus se ejecuta, éste continúa con el fichero PE infectado. En este caso el ransomware, el cual ha sido escrito en lenguaje C/C++. Para esta muestra se observa la siguiente fecha de compilación.

Sections Count	1	1
Time Date Stamp	626a11f6	jueves, 28.04.2022 04:03:02 UTC
Ptr to Symbol Table	0	0

Figura 2. Información del ransomware.

Esta fecha de compilación (TimeDateStamp) corresponde al ransomware e indica que la muestra ha sido creada el 28/04/2022. No obstante, hay que tener en cuenta que esta información puede ser fácilmente alterada.

4.1 FLUJO DE EJECUCIÓN

La muestra analizada está infectada con el virus ***Neshta***. El primer paso es la ejecución del código del virus. A continuación, se muestra un diagrama con los pasos realizados por el código:

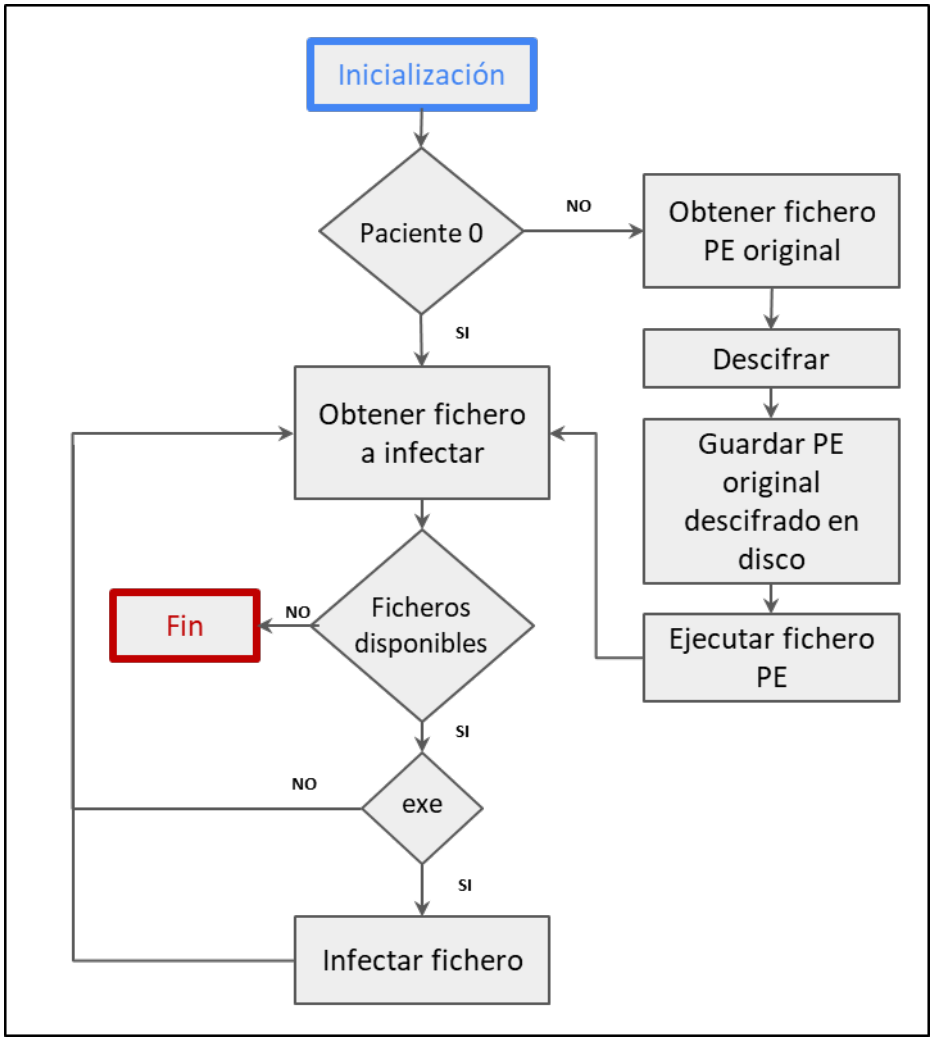


Figura 3. Diagrama de ejecución del código dañino Neshta.

1. Si es el paciente 0, es decir, el primer caso de Neshta (el cual aún no ha infectado ningún fichero PE) directamente salta a la infección de los ficheros.
2. Si no es el paciente 0, es decir, si ya es un fichero que ha sido infectado por Neshta, descifra el fichero PE original que se encuentra en el *overlay* de la muestra.
3. Guarda en la carpeta **"%TMP%\3582-490**" el fichero original. Este nombre de carpeta esta *hardcodeado* en el propio binario.

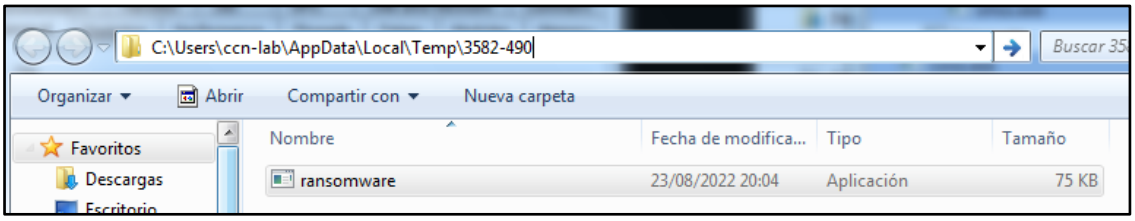


Figura 4. Fichero original del ransomware después de ser descifrado por el Neshta.El fichero es copiado en la carpeta temporal "3582-490"

4. Ejecuta el fichero PE original. En este caso se ejecuta el ransomware.
5. Enumera todos los ficheros del sistema.
6. Filtra aquellos ficheros que quiere infectar, en este caso infecta solo ficheros PE (EXE), por lo que las DLL no se ven afectadas. Y finalmente comprueba que el fichero no haya sido previamente infectado.
7. Infecta los ficheros que han pasado los filtros.

Se podría interpretar que el virus actúa como un *packer*. Cuando el virus infecta un nuevo PE, su comportamiento es copiar su propio código y añadirle en el *overlay* el fichero original infectado. Por lo que ahora el fichero infectado cuando se ejecuta lo primero que va a hacer es obtener el fichero original del *overlay*, ejecutarlo y después seguir con su rutina de infección. De esta forma es capaz de propagarse por todo el sistema.

La siguiente imagen muestra como se ve un fichero antes y después de haber sido infectado por el virus Neshta.

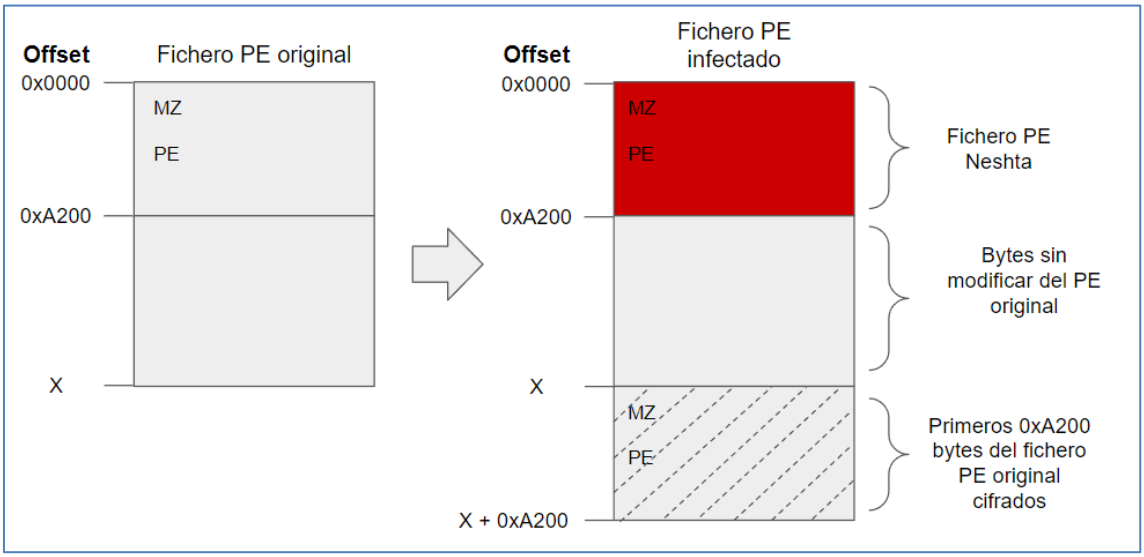


Figura 5. Esquema de un fichero infectado por Neshta.

Para este caso en concreto, el fichero PE original es el ransomware y el funcionamiento de éste se explica a continuación.

Por último, ***Neshta*** crea dos ficheros adicionales, el primero lo guarda en **%SystemRoot%** con el nombre **dyrectx.sys**. Se trata de un fichero en texto plano el cual contiene el último fichero infectado que se ha ejecutado en el sistema.

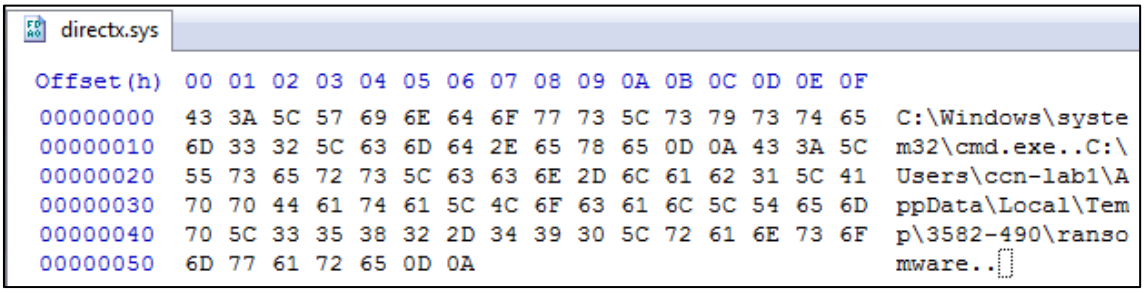


Figura 6. Contenido del fichero C:\Windows\directx.sys creado por Neshta.

El otro fichero lo crea en la carpeta temporal **%TMP%**, con el nombre "**tmp5023.tmp**". Aunque parece un nombre aleatorio, es un nombre que tiene *hardcodeado* ***Neshta***, por lo que siempre creará el mismo fichero el cual es creado una vez infectados todos los ficheros.

4.2 PERSISTENCIA

El código dañino ***Neshta*** se instala en la carpeta **%SystemRoot%** con el nombre svchost.com.

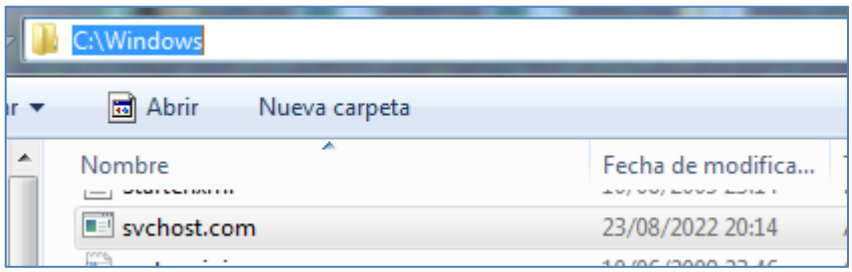


Figura 7. Fichero svchost.com copiado por Neshta.

A continuación, crea la siguiente clave de registro:

HKLM\SOFTWARE\Classes\exefile\shell\open\command

Esta clave de registro hace que cada vez que se ejecute un fichero ".exe" en el sistema, se ejecute automáticamente el fichero "C:\Windows\svchost.com", creado previamente.

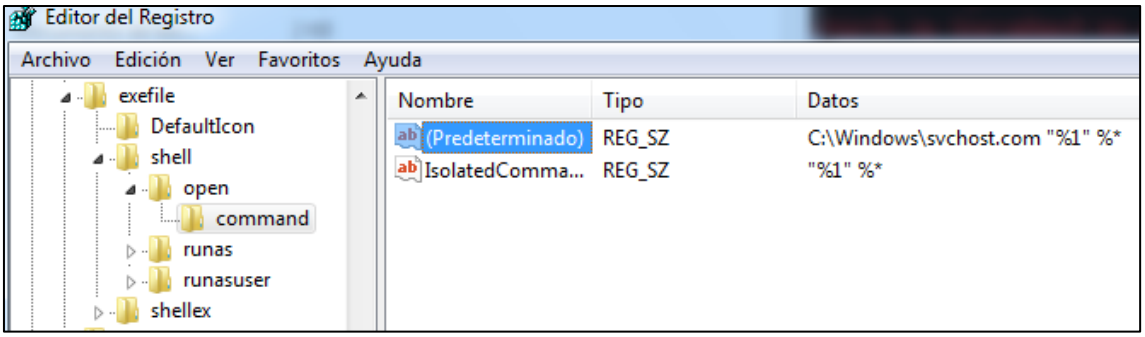


Figura 8. Clave de registro creada por Neshta con el nombre de fichero svchost.com.

5. CARACTERÍSTICAS TÉCNICAS

A continuación, se recogen las características técnicas del ransomware, ya que como se explica en el apartado anterior, el virus Neshta tiene como objetivo infectar todos los ficheros PE de un sistema, pero no se observa ninguna actividad que comprometiera al usuario o al correcto funcionamiento del sistema. No obstante, es recomendable detectar la actividad de este virus con las reglas que se proporcionan en Anexo y proceder a la eliminación del mismo.

5.1 FUNCIONAMIENTO GENERAL

Este ransomware acepta dos opciones como argumento en su ejecución, la opción /f y la opción /d.

- Con la opción "/f", se puede especificar la ruta de un único fichero que se quiere cifrar.
- La opción "/d", se puede especificar un único directorio que se quiere cifrar.
- Si no se le proporciona ninguna opción, todo el disco será cifrado.

La siguiente imagen resume el funcionamiento general del ransomware.

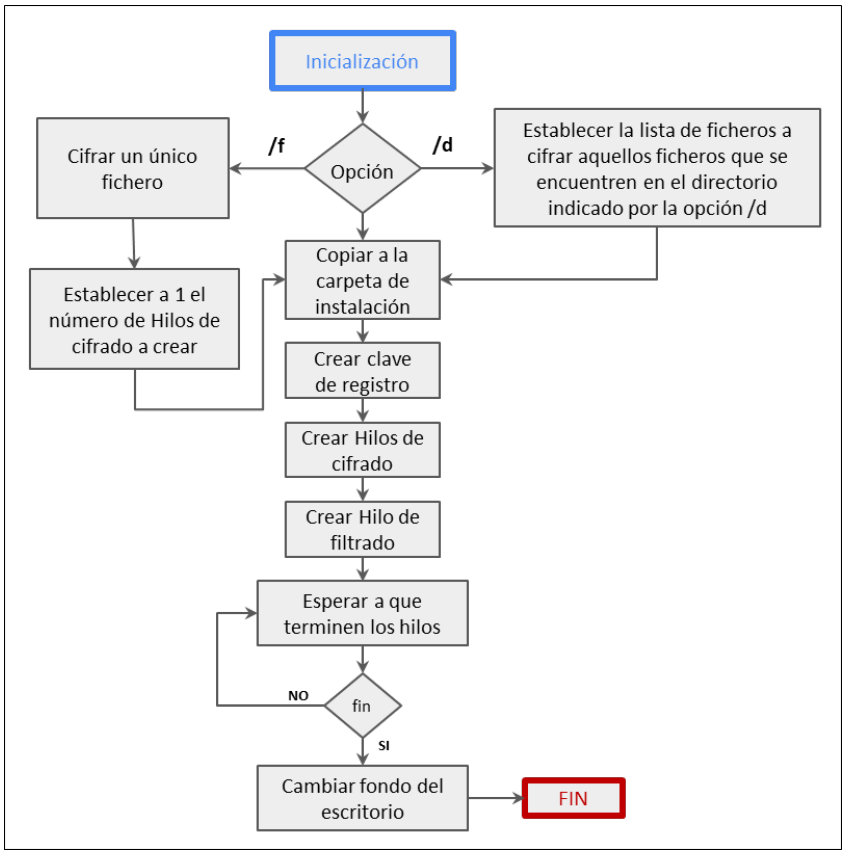


Figura 9. Flujo de ejecución del ransomware.

Cuando al ransomware no se le proporciona ningún argumento sucede lo siguiente:

1. Se crea persistencia en el sistema:
   a. Se copia a sí mismo en la carpeta:

      "%LocalAppData%" (C:\Users\{username}\AppData\Local)

   b. Crea la clave de registro apuntando al fichero que se ha copiado en la carpeta "%LocalAppData%":

      Software\Microsoft\Windows\CurrentVersion\RunOnce LicChk

```
if ( !result )
{
  v4 = 520;
  result = AllocateHeap(260, 2);
  v2 = result;
  if ( result )
  {
    RegQueryValueExW(v5, L"LicChk", 0, 0, result, &v4);
    if ( lstrcmpiW(v2, lpString2) )
    {
      if ( !RegOpenKeyExW(
              HKEY_LOCAL_MACHINE,
              L"Software\\Microsoft\\Windows\\CurrentVersion\\RunOnce",
              0,
              0x20006u,
              &v5) )
      {
        v3 = lstrlenW(lpString2);
        RegSetValueExW(v5, L"LicChk", 0, 1u, lpString2, 2 * v3 + 2);
      }
    }
    RegCloseKey(v5);
    result = FreeHeap(v2);
  }
}
```

Figura 10. Código para la creación de clave de registro para la persistencia en el sistema.

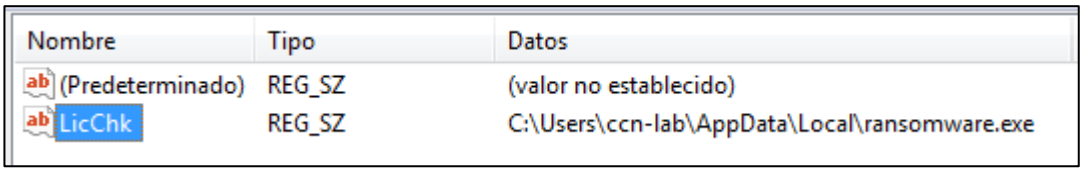


Figura 11. Clave de registro para la persistencia en el sistema.

2. Crea 3 hilos que se quedan a la espera de recibir nuevas órdenes. Estos hilos son los responsables de cifrar los ficheros.
3. Otro hilo es creado para encargase de mandar las órdenes a los 3 hilos creados posteriormente. Este hilo es el encargado de recorrer todos los ficheros del sistema e ir filtrando los ficheros que considere válidos para cifrar. Los criterios que sigue principalmente para elegir un fichero válido para cifrar son los siguientes:

a. La muestra tiene una lista de extensiones de ficheros que no pueden ser cifrados. Se puede encontrar la lista completa en el ANEXO en el apartado "LISTA DE EXTENSIONES EXCLUIDAS DE SER CIFRADAS".

b. El ransomware contiene también una lista de carpetas las cuales va a evitar cifrar. Se puede encontrar la lista completa en el ANEXO en el apartado "LISTA DE CARPETAS EXCLUIDAS DE SER CIFRADAS".

4. El hilo principal del programa se queda esperando a que todos los hilos terminen su trabajo.

5. Una vez todos los hilos han acabado, el código dañino cambia el fondo de pantalla, por un fondo con un mensaje que indica que todos los ficheros del sistema han sido cifrados tal y como se ve en la figura 22.

El código dañino usa un sistema de colas para sincronizar los hilos de cifrado y el hilo que filtra los ficheros a cifrar, de esta manera consigue que el cifrado se haga de forma sincronizada.

Para eso se hace con la ayuda del sistema I/O que proporciona Windows, usando las APIs:

- CreateIoCompletionPort.
- GetQueuedCompletionStatus.
- GetQueuedCompletionStatus.

Proporcionadas por el sistema operativo Windows. Lo cual les permite pasarse tareas entre los hilos.

En la siguiente imagen se ilustra la sincronización entre los hilos que se produce a la hora de cifrar los ficheros.

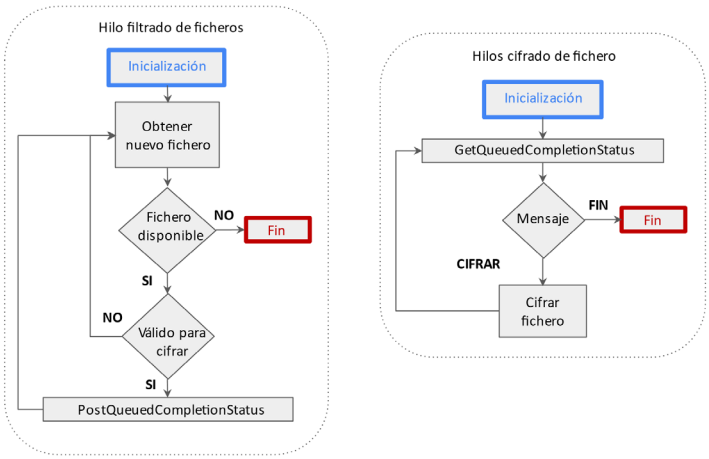


Figura 12. Diagrama de ejecución del hilo de filtrado y el hilo de cifrado creados por el ransomware.

Mientras que la función PostQueuedCompletionStatus, añade un elemento a la cola, en este caso el nombre del fichero a cifrar, GetQueuedCompletionStatus, obtiene un elemento de la cola. De esta forma el código dañino consigue sincronizar los hilos para cifrar todo el sistema de una forma más rápida.

5.2 TÉCNICAS ANTI ANÁLISIS

5.2.1 CIFRADO DE CADENAS

El código dañino mantiene cifradas las distintas cadenas que emplea, como el mensaje de rescate y la lista de extensiones de ficheros a cifrar. Las cadenas se descifran en tiempo de ejecución.

Para el cifrado de las cadenas utiliza un algoritmo AEAD (Authenticated Encryption with Associated Data) en este caso el ChaCha20-Poly1305 con una clave de 16 bytes y un nonce de 12 bytes.

```
1  char *DecryptArrayOfStrings(char *IN_key, char *IN_nonce, char *IN_encrypted_data, int IN_encrypted_data_size, i
2  {
3    // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]
4
5    v7 = 0;
6    OUT_decrypted_data = AllocateHeap(IN_encrypted_data_size, 1);
7    v9 = OUT_decrypted_data;
8    if ( OUT_decrypted_data )
9    {
10     if ( chacha20_poly1305(IN_key, IN_nonce, IN_encrypted_data, OUT_decrypted_data, IN_encrypted_data_size) )
11     {
12       v7 = 1;
13       *a6 = StringCopy_(a7 + a5, v9);
14     }
15     else
16     {
17       FreeHeap(v9);
18     }
19     OUT_decrypted_data = v7;
20   }
21   return OUT_decrypted_data;
22 }
```

Figura 13. Función utilizada para descifrar cadenas.

5.2.2 RESOLUCIÓN DE APIS

El ransomware resuelve dinámicamente las APIs de Windows necesarias para su funcionamiento. La lista completa de las APIs resultas se encuentra en el ANEXO de este documento.

Estas APIs son resueltas haciendo uso de las funciones "LoadLibraryA" y "GetProcAddress".

```
h_advapi32 = LoadLibraryW(L"advapi32.dll");
h_shell32 = LoadLibraryW(L"Shell32.dll");
h_ntdll = LoadLibraryW(L"ntdll.dll");
h_shlwapi = LoadLibraryW(L"Shlwapi.dll");
h_psapi = LoadLibraryW(L"Psapi.dll");
h_rstrtmgr = LoadLibraryW(L"Rstrtmgr.dll");
if ( h_advapi32 )
{
  SystemFunction036 = (int)GetProcAddress(h_advapi32, "SystemFunction036");
  A_SHAInit = (int)GetProcAddress(h_advapi32, "A_SHAInit");
  A_SHAUpdate = (int)GetProcAddress(h_advapi32, "A_SHAUpdate");
  A_SHAFinal = (int)GetProcAddress(h_advapi32, "A_SHAFinal");
  *(_DWORD *)CryptReleaseContext = GetProcAddress(h_advapi32, "CryptReleaseContext");
  *(_DWORD *)CryptAcquireContextW = GetProcAddress(h_advapi32, "CryptAcquireContextW");
  *(_DWORD *)CryptGenRandom = GetProcAddress(h_advapi32, "CryptGenRandom");
  *(_DWORD *)RegOpenKeyExW = GetProcAddress(h_advapi32, "RegOpenKeyExW");
  *(_DWORD *)RegQueryValueExW = GetProcAddress(h_advapi32, "RegQueryValueExW");
  *(_DWORD *)RegCreateKeyExW = GetProcAddress(h_advapi32, "RegCreateKeyExW");
  *(_DWORD *)RegSetValueExW = GetProcAddress(h_advapi32, "RegSetValueExW");
  *(_DWORD *)RegCloseKey = GetProcAddress(h_advapi32, "RegCloseKey");
  RegSetKeyValueW = (int)GetProcAddress(h_advapi32, "RegSetKeyValueW");
  if ( RegSetKeyValueW )
    dword_40D4E0 = 1;
}
if ( h_shell32 )
{
  *(_DWORD *)ShellExecuteExW = GetProcAddress(h_shell32, "ShellExecuteExW");
  *(_DWORD *)SHChangeNotify = GetProcAddress(h_shell32, "SHChangeNotify");
  *(_DWORD *)CommandLineToArgvW = GetProcAddress(h_shell32, "CommandLineToArgvW");
}
if ( h_ntdll )
{
  RtlComputeCrc32 = (int (__stdcall *)(_DWORD, _DWORD, _DWORD))GetProcAddress(h_ntdll, "RtlComputeCrc32");
  qsort = (int)GetProcAddress(h_ntdll, "qsort");
```

Figura 14. Función encargada de resolver las APIs de Windows de forma dinámica.

5.3 ESQUEMA DE CIFRADO

El código dañino después de filtrar los ficheros a cifrar, mediante el sistema de colas que se ha explicado previamente, envía a los hilos de cifrado los ficheros a cifrar.

En caso que el fichero tenga un tamaño superior a 2621440 bytes (2,6 MB) solo se cifrarán los primeros 2621440 bytes, mientras que el resto de datos permanecerán sin modificar.

Esto es una práctica muy común en el ransomware para acelerar el proceso de cifrado. Cifrando los primeros 2,6 MB es suficiente para asegurarse que el fichero no sea reconocido y por lo tanto la totalidad de sus datos.

```
02621376  DA DB E4 A5 76 24 7E 94 73 88 07 8F C1 D0 F3 7A  ÚÛä¥v$~"sˆ..ÁÐóz
02621392  7B 74 58 7A E8 40 F5 7E 50 18 63 23 54 EC CB C8  {tXzè@õ~P.c#TìËÈ
02621408  9E BA D5 34 9F E4 5B 08 12 5A 49 DA FB EC 1F 29  ž°Õ4Ÿä[..ZIÚûì.)
02621424  F5 2D 4E F0 05 2A EF 13 9C 78 59 51 46 25 62 5D  õ-Nð.*ï.œxYQF%b]
02621440  61 20 70 61 72 74 69 72 20 64 65 20 61 71 75 69  a partir de aqui
02621456  20 6C 6F 73 20 64 61 74 6F 73 20 6E 6F 20 73 65   los datos no se
02621472  6E 20 63 69 66 72 61 64 6F 73 20 00 00 00 00 00  n cifrados .....
02621488  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
02621504  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
02621520  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

Figura 15. Contenido de un fichero cifrado. Se observa como a partir del offset 2621440 los datos no son cifrados.

A los ficheros que han sido cifrados, se les cambia el nombre. Concretamente, se les añade la extensión "xnxxx" como se ve en la siguiente figura.

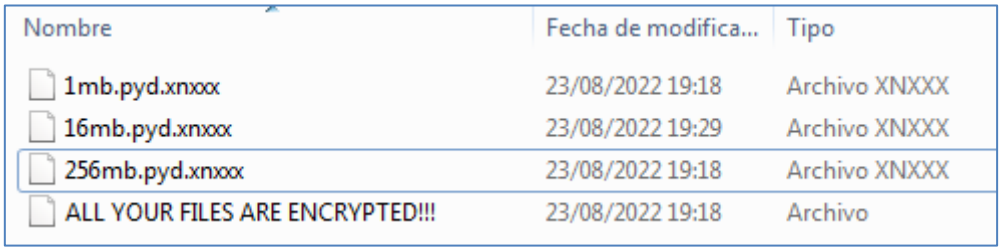


Figura 16. Nombre de los ficheros después de ser cifrados.

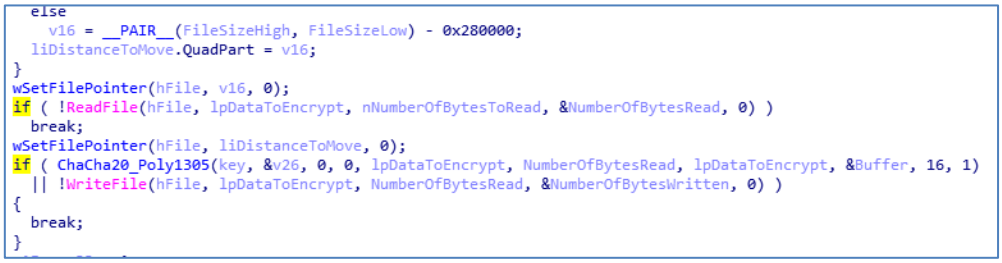
5.3.1 CIFRADO DE FICHEROS

Para el cifrado de los ficheros, utiliza una combinación de cifrado asimétrico y cifrado simétrico, concretamente el NTRUEncrypt para el cifrado asimétrico y el ChaCha20-Poly1305 para el cifrado simétrico.

Para cada fichero, genera una clave aleatoria de 32 bytes y un *nonce* de 12 bytes. Esta clave aleatoria es utilizada para cifrar el contenido del fichero. Es cifrada con una clave pública usando el algoritmo NTRUEncrypt. De esta forma solo quien tenga la clave privada, podrá descifrar la clave simétrica y, finalmente, descifrar el contenido del fichero.

```
// initialize keys
if ( GenerateRandomBytes(&RandomKey, 0x2C, tContext->RandomCls)
  || chacha20_spand_key(0, key, &RandomKey, 0x100)
  || EncryptChaCha20KeyAndNonce(
       0,
       lpFileName,
       hFile,
       &RandomKey,
       0x2Cu,
       tContext->GlobarVar1,
       &tContext->PublicKey,
       tContext->RandomCls,
       tContext->RandomKeyEncryptedWithPubKey) )
{
  goto LABEL_39;
}
```

Figura 17. Inicialización de la clave aleatoria y cifrado de la misma usando la clave pública.



```
  else
    v16 = __PAIR__(FileSizeHigh, FileSizeLow) - 0x280000;
  liDistanceToMove.QuadPart = v16;
}
wSetFilePointer(hFile, v16, 0);
if ( !ReadFile(hFile, lpDataToEncrypt, nNumberOfBytesToRead, &NumberOfBytesRead, 0) )
  break;
wSetFilePointer(hFile, liDistanceToMove, 0);
if ( ChaCha20_Poly1305(key, &v26, 0, 0, lpDataToEncrypt, NumberOfBytesRead, lpDataToEncrypt, &Buffer, 16, 1)
  || !WriteFile(hFile, lpDataToEncrypt, NumberOfBytesRead, &NumberOfBytesWritten, 0) )
{
  break;
}
```

Figura 18. Cifrado del contenido del fichero usando la clave generada previamente.

Una vez son cifrados los datos, la clave simétrica que ha sido cifrada con la clave pública se escribe al final del fichero.

```
if ( v15 == v32 )
{
  wSetFilePointer(hFile, 0i64, 2u);
  if ( WriteFile(
         hFile,
         tContext->RandomKeyEncryptedWithPubKey,
         tContext->RandomKeyEncryptedWithPubKeySize,
         &NumberOfBytesWritten,
         0) )
  {
    WriteFile(hFile, tContext->lpBuffer, tContext->lpBufferSize, &NumberOfBytesWritten, 0);
    WriteFile(hFile, &Buffer, 0x10u, &NumberOfBytesWritten, 0);
    SetEndOfFile(hFile);
    v17 = 1;
    goto LABEL_33;
  }
  break;
}
```

Figura 19. Escritura de la clave simétrica cifrada con la clave pública

6. MENSAJE DE RESCATE

Cuando termina el proceso de cifrado, en todos los directorios se puede encontrar un documento con el nombre "**ALL YOUR FILES ARE ENCRYPTED!!!**" el cual contiene el mensaje de rescate.

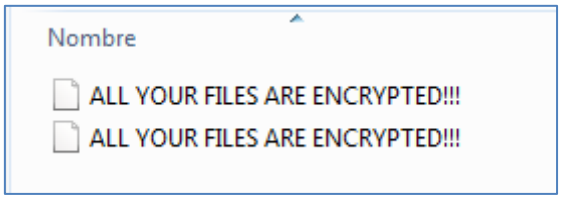


Figura 20. Fichero de rescate.

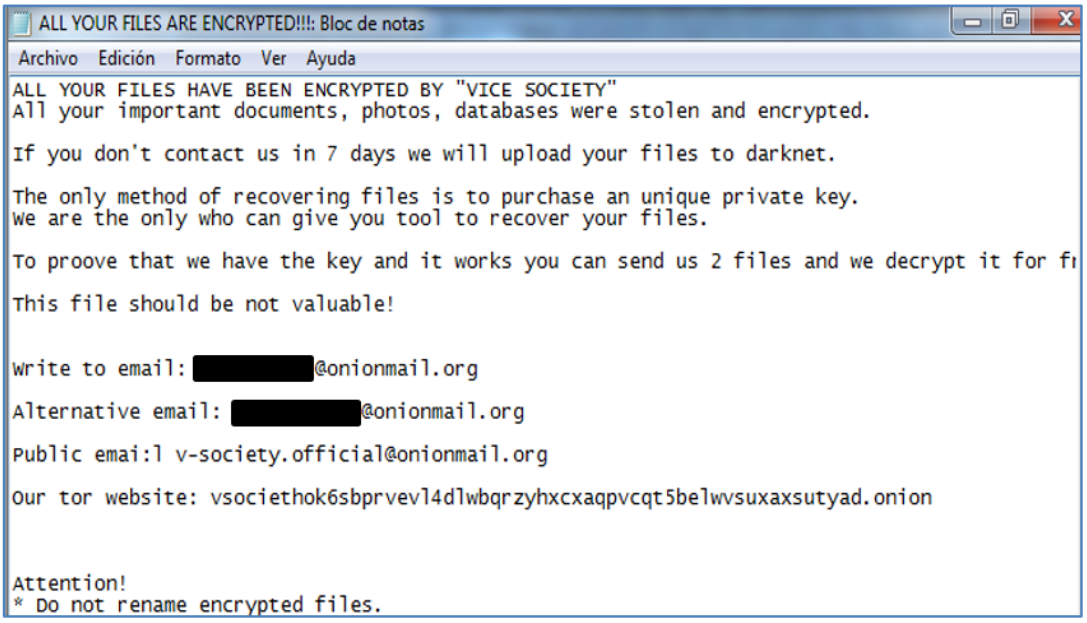


Figura 21. Contenido del fichero de rescate.

En este mensaje de rescate se pueden encontrar las indicaciones para la recuperación de los ficheros cifrados.

Así mismo, el fondo de pantalla del escritorio es cambiado por uno que genera el ransomware y el cual indica que todo el sistema ha sido cifrado.

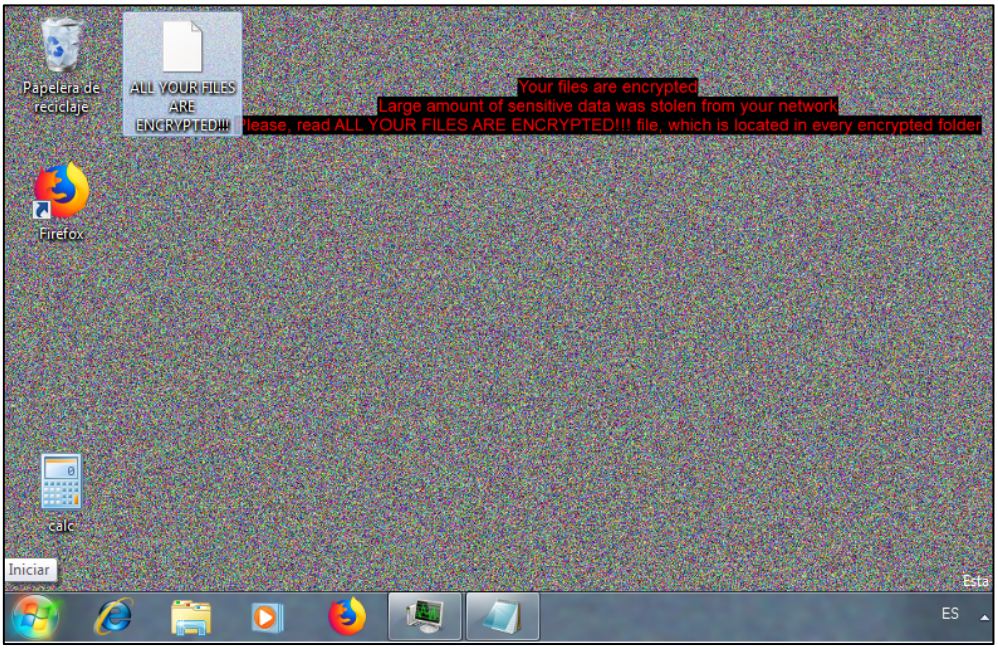


Figura 22. Fondo de pantalla después de que todos los ficheros del sistema sean cifrados.

7. INDICADORES DE COMPROMISO

Neshta		
Tipo	**Descripción**	**Valor**
Sistema de ficheros	Carpeta en la que los ficheros originales son guardados una vez se descifran. Previamente a su ejecución.	"%TEMP%\3582-490
Sistema de ficheros	Fichero que crea para hacer una copia de sí mismo.	%SystemRoot%\svchost.com
Sistema de ficheros	Indica el ultimo fichero infectado que ha sido ejecutado	%SystemRoot%\directx.sys
Sistema de ficheros	Creado una vez todos los ficheros han sido infectados	%TMP%\tmp5023.tmp
Clave de registro	Clave de registro creada para la persistencia	**Clave:** HKLM\SOFTWARE\Classes\exefile\shell\open\command **Valor de Registro:** (Default) **Valor:** %SystemRoot%\svchost.com "%1" %*
Mutex	Mutex creado para evitar que estén en ejecución múltiples instancias del virus	MutexPolesskayaGlush*.*

Ransomware		
Tipo	**Descripción**	**Valor**
Sistema de ficheros	Extensión de los ficheros cifrados	".xnxxx"
Sistema de ficheros	Nombre del fichero que contiene el mensaje de rescate	"ALL YOUR FILES ARE ENCRYPTED!!!
Sistema de ficheros	El nombre de este fichero es generado a raíz de la clave pública, y tiene un tamaño de 40 caracteres. Se utiliza para guardar información relacionada con el cifrado	**Caso genérico:** C:\Users\Public\[A-F0-9]{40} **Caso concreto de la muestra analizada:** C:\Users\Public\6F4B95343D2D76D10F87017F60B7235937F26A64
Clave de registro	Clave de registro creada para la persistencia	**Clave:** HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunOnce **Valor de Registro:** LicChk **Valor:** {path_al_fichero_ejecutable_del_ransomware}

8. DESINFECCIÓN

Para la correcta desinfección del ***Neshta*** es recomendable borrar todos los ficheros que aparecen en los indicadores de compromiso del apartado anterior, correspondientes a ***Neshta***, aun así, esto no hace que los ficheros ya infectados se desinfecten, por lo que es necesario usar una solución antivirus para desinfectar todos los ficheros infectados por ***Neshta***.

Por otro lado, para los ficheros cifrados por el ransomware, solo los operadores del ransomware son capaces de descifrar los ficheros.

Para evitar desencadenar sucesivas ejecuciones del código dañino, es conveniente revisar dos posibles mecanismos que pueden relanzar la muestra.

El primero consiste borrar la clave de registro "LicChk" tal y como aparece en el apartado de indicadores de compromiso. Así mismo, también es recomendable borrar el ransomware. En este caso el ransomware se encuentra en el valor indicado por la clave de registro "LicChk".

9. MITIGACIÓN

Una manera 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).

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.

10. REGLAS DE DETECCIÓN

10.1 REGLAS YARA

```
rule Vice_Society_ransomware
{
        meta:
                author      = "Centro Criptológico Nacional (CCN)"
                date        = "26/08/2022"
                description = "Ransomware utilizado por el grupo cibercriminal Vice Society"
        strings:
                $s1 = "{{IDENTIFER}}"
                $s2 = "Software\\Microsoft\\Windows\\CurrentVersion\\RunOnce" wide
                $s3 = "EES401EP2"
                $s4 = "EES587EP1"
                $s5 = "LicChk" wide

        condition:
                uint16(0) == 0x5A4D and
                uint32(uint32(0x3C)) == 0x00004550 and
        all of them
}
rule Neshta_fileinfector
{
        meta:
                author      = "Centro Criptológico Nacional (CCN)"
                date        = "23/08/2022"
                description = "Virus Neshta"
        strings:
                $s1 = "Delphi-the best. Fuck off all the rest. Neshta 1.0 Made in Belarus."
                $s2 = "Best regards 2 Tommy Salo. [Nov-2005] yours [Dziadulja
Apanas]\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
        condition:
                uint16(0) == 0x5A4D and
                uint32(uint32(0x3C)) == 0x00004550 and
        all of them
}
```

ANEXO

LISTA DE EXTENSIONES EXCLUIDAS DE SER CIFRADAS

.386	.lock
.adv	.mod
.ani	.mpa
.bat	.msc
.bin	.msi
.cab	.msp
.cmd	.msstyles
.com	.msu
.cpl	.nls
.cur	.nomedia
.deskthemepack	.ocx
.diagcab	.prf
.diagcfg	.ps1
.diagpkg	.rom
.dll	.rtp
.drv	.scr
.exe	.spl
.hlp	.sys
.hta	.theme
.icl	.themepack
.icns	.wpx
.ico	.shs
.ics	
.idx	
.lnk	

LISTA DE CARPETAS EXCLUIDAS DE SER CIFRADAS

$RECYCLE.BIN	Microsoft SDKs	Windows App Certification Kit
vscode	Microsoft Shared	Windows Defender
AVG	Microsoft VS Code	Windows Kits
Avast	Microsoft Visual Studio	Windows Mail
Avira	Microsoft.NET	Windows Media Player
COMODO	MovieMaker	Windows Multimedia Platform
Chrome	Mozilla	Windows NT
Common Files	Mozilla Firefox	Windows Phone Kits
Common7	NVIDIA Corporation	Windows Phone Silverlight Kits
Dr.Web	Opera	Windows Photo Viewer
ESET	Package Cache	Windows Portable Devices
Firefox	Packages	Windows Security
Install Shield Installation Information	Reference assemblies	Windows Sidebar
Intel	Spytechsoftware	WindowsApps
Internet Explorer	Symantec	WindowsPowerShell
KasperskyLab	Symantec Client Security	Wsus
McAfee	System Volume Information	Yandex Browser
Microsoft	Temp	sysconfig
Microsoft Help	Windows	

LISTA DE APIS RESULTAS DINÁMICAMENTE

advapi32.dll	Shell32.dll	Ntdll.dll
SystemFunction036 A_SHAInit A_SHAUpdate A_SHAFinal CryptReleaseContext CryptAcquireContextW CryptGenRandom RegOpenKeyExW RegQueryValueExW RegCreateKeyExW RegSetValueExW RegCloseKey RegSetKeyValueW	ShellExecuteExW SHChangeNotify CommandLineToArgvW	RtlComputeCrc32 qsort RtlMoveMemory
Shlwapi.dll	**Psapi.dll**	**Rstrmgr.dll**
PathFindFileNameW PathFindExtensionW	GetModuleFileNameExW	RmEndSession RmStartSession RmGetList RmShutdown RmRegisterResources

MENSAJE DE RESCATE

ALL YOUR FILES HAVE BEEN ENCRYPTED BY "VICE SOCIETY"

All your important documents, photos, databases were stolen and encrypted.

If you don't contact us in 7 days we will upload your files to darknet.

The only method of recovering files is to purchase an unique private key.

We are the only who can give you tool to recover your files.

To proove that we have the key and it works you can send us 2 files and we decrypt it for free (not more than 2 MB each).

This file should be not valuable!

Write to email: <editado>@onionmail.org

Alternative email: <editado>@onionmail.org

Public emai:l v-society.official@onionmail.org

Our tor website: vsociethok6sbprvevl4dlwbqrzyhxcxaqpvcqt5belwvsuxaxsutyad.onion

Attention!

* Do not rename encrypted files.

* Do not try to decrypt your data using third party software, it may cause permanent data loss.

* Decryption of your files with the help of third parties may cause increased price (they add their fee to ours) or you can become a victim of a scam