Informe Código Dañino
CCN-CERT ID-28/20

Avaddon

Septiembre 2020

Edita:

GOBIERNO DE ESPAÑA MINISTERIO DE DEFENSA

CCN centro criptológico nacional

Fecha de Edición: septiembre de 2020

LIMITACIÓN DE RESPONSABILIDAD

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

AVISO LEGAL

Quedan rigurosamente prohibidas, sin la autorización escrita del Centro Criptológico Nacional, bajo las sanciones establecidas en las leyes, la reproducción parcial o total de este documento por cualquier medio o procedimiento, comprendidos la reprografía y el tratamiento informático, y la distribución de ejemplares del mismo mediante alquiler o préstamo públicos.

ÍNDICE

1. SOBRE CCN-CERT, CERT GUBERNAMENTAL NACIONAL

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

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

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

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

2. RESUMEN EJECUTIVO

La muestra analizada se corresponde con Avaddon, un *ransomware* involucrado en múltiples campañas orquestadas por diversos atacantes desde junio de este año que emplea un modelo de negocio conocido como RaaS (*Ransomware-as-a-Service*) [REF.- 1]. Según este modelo, por un lado, los autores del *malware* se encargan de su desarrollo y la operativa del servicio de pago vía Tor. Por otro lado, los afiliados que se unan al programa se encargarán de su distribución, por ejemplo, mediante *spear-phishing, exploit-kits*, etc., a cambio de un porcentaje del dinero recaudado en los rescates.

Entre el conjunto de condiciones impuesto por los autores de Avaddon a sus afiliados se encuentra la prohibición de atacar víctimas que formen parte del CIS (*Commonwealth of Independent State*) [REF.- 2].

Desde junio de este año se han identificado múltiples campañas a través de las cuales se ha distribuido Avaddon. Por ejemplo, el 4 de julio se identificaron diversas campañas de *spam* en las que se utilizaba la *botnet* Phorpiex/Trik [REF.-3] como vía de entrada para distribuir Avaddon entre otros códigos dañinos.

El *ransomware* está desarrollado en C++, emplea AES256 junto con RSA2028 para cifrar los ficheros de las víctimas y no requiere de conectividad a Internet para ejecutar sus acciones dañinas. Antes de comenzar a cifrar los ficheros desactiva las *shadow copies* y elimina las copias de seguridad del sistema. El código dañino tiene capacidad para infectar unidades de red, dispositivos extraíbles, así como otras unidades montadas en el sistema de la víctima.

3. CARACTERÍSTICAS DEL CÓDIGO DAÑINO

El código dañino realiza las siguientes acciones:

- Desactiva las *shadow copies* y elimina las copias de seguridad.
- Termina procesos y servicios relacionados con almacenamiento, escaneo y recuperación de ficheros.
- Cifra un gran número de ficheros del sistema y de las unidades/discos montados (incluidas unidades de red) haciendo uso de RSA y AES.

4. DETALLES GENERALES

La firma del código dañino es la siguiente:

MD5	275E4A63FC63C995B3E0D464919F211B
SHA1	51D85210C2F621CA14D92A8375EE24D62F9D7F44
SHA256	CC95A8D100F70D0FBF4AF14E852AA108BDB0E36DB4054C3F60B3515818A71F46

La muestra ha sido vista por primera vez el 31 de agosto de 2020 desde España y, a fecha de realización del presente informe, presenta un *ratio* de detección alto en soluciones antivirus.

El fichero tiene un tamaño de 736608 bytes, se ha compilado para arquitecturas de 32 bits el 29 de agosto de 2020 según su *compiler-stamp* (dicha información puede haber sido intencionadamente modificada) y está desarrollado en C++. Sus características estáticas se muestran en la siguiente imagen:

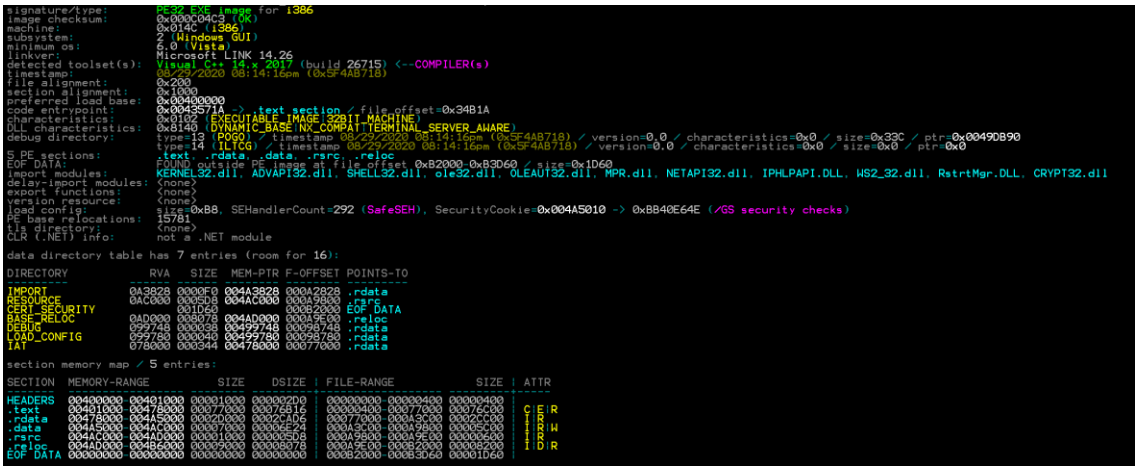


Figura 1. Características estáticas del binario

El binario se encuentra firmado (hora y fecha: 10:02 30/08/2020) por el *publisher* "*Gold Stroy SP Z 0 0*" y aparenta ser una aplicación vinculada con Microsoft, concretamente imita los atributos de la aplicación legítima *taskhost.exe.* Cabe destacar que otros códigos dañinos, por ejemplo, el *ransomware* Maze o ParallaxRAT están firmados también por dicho *Publisher* [REF.- 4].

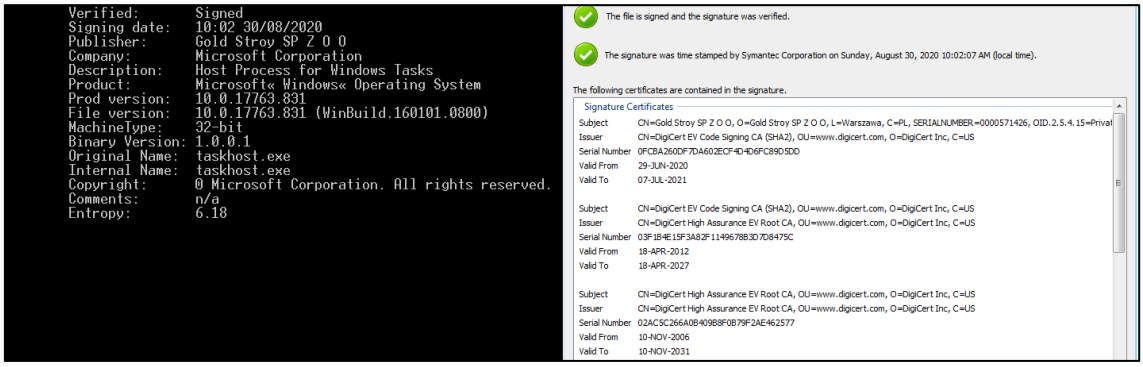


Figura 2. Firma: Gold Stroy SP Z 0 0

Pese a que el binario no contiene *strings* en claro de relevancia se identifican cadenas de interés cuando se aplican operaciones XOR junto con determinadas claves a su contenido. En la siguiente salida se muestran *strings* vinculados con algunos de los procesos que intentará finalizar antes de comenzar con el cifrado de los ficheros, así como la URL oficial del proyecto Tor.

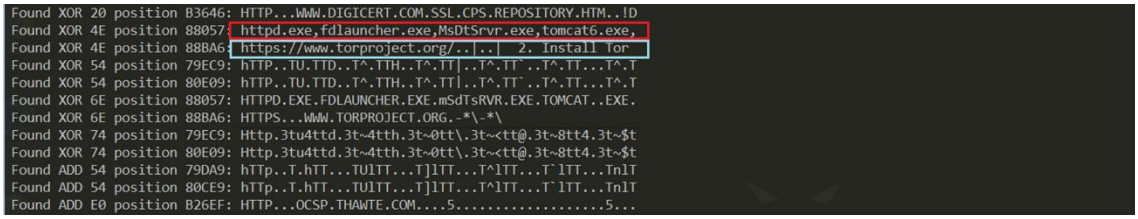


Figura 3. Cadenas ofuscadas

Dentro del fichero de manifiesto del binario se observa la directiva "*requireAdministrator*" para ejecutar el binario con permisos de administrador.

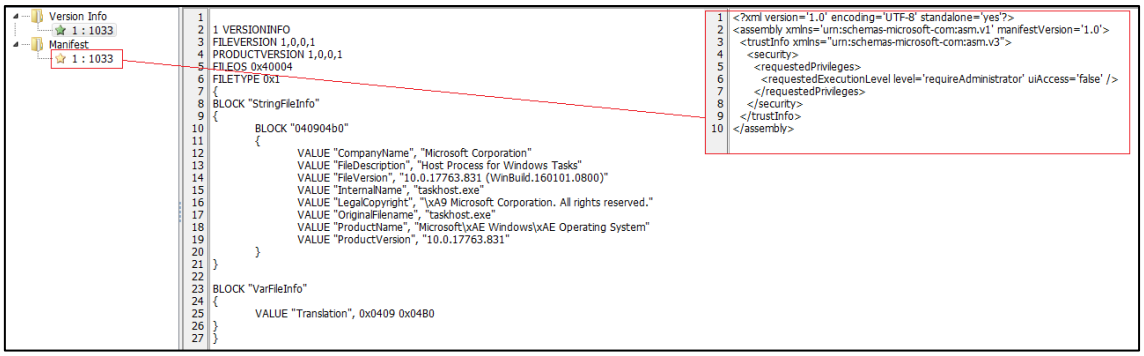


Figura 4. Fichero de manifiesto

El investigador David Montenegro publicó en su Twitter [REF.-5] imágenes sobre los anuncios de reclutamiento de afiliados vinculados con Avaddon en determinados foros *underground* de habla rusa en donde los autores del mismo publicitan su servicio "*Ransomware-as-a-Service*".

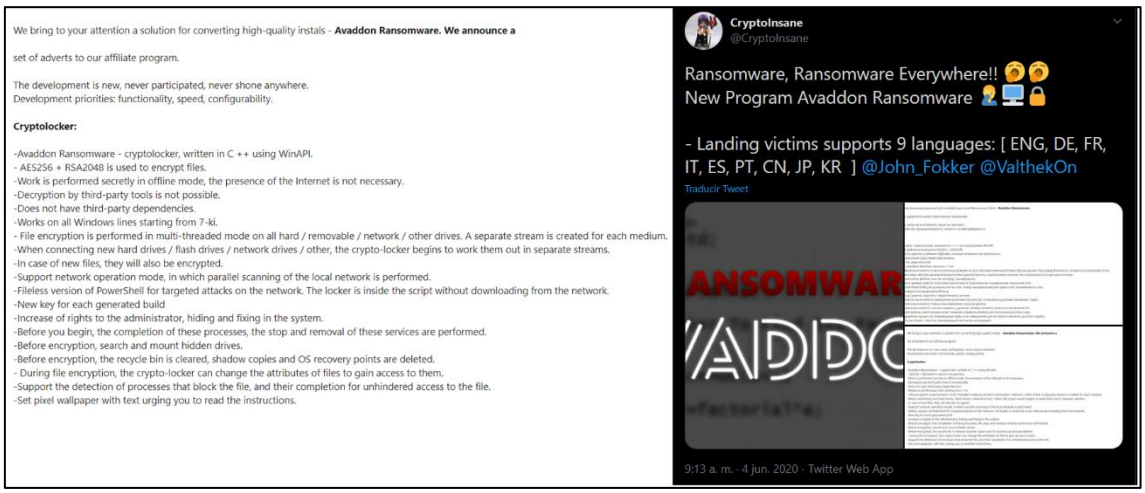


Figura 5. RaaS (Avaddon). Fuente de la imagen: Twitter @CryptoInsane

Asimismo, la empresa de ciber inteligencia israelí Kela [REF.- 6] informaba también de algunos de los anuncios realizados por los operadores de Avaddon en donde se publicaban portales de acceso a los datos extraídos de determinadas víctimas.

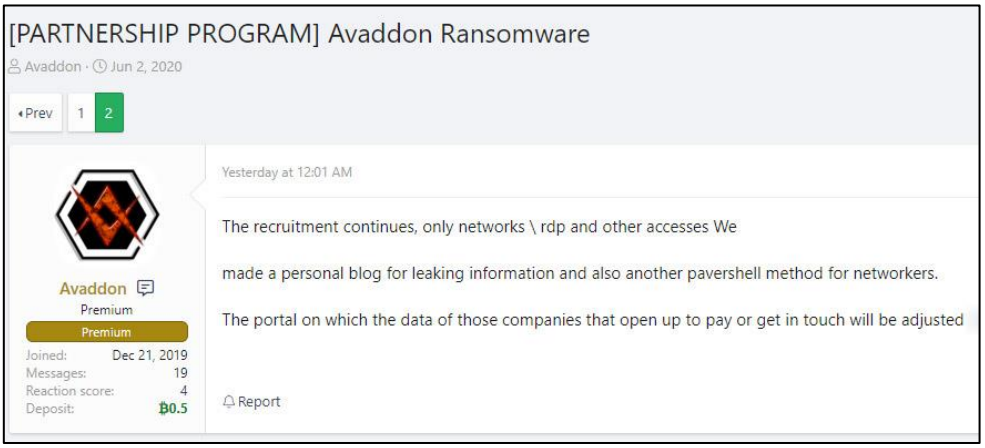


Figura 6. RaaS (Avaddon). Fuente: https://www.bleepingcomputer.com/

5. PROCEDIMIENTO DE INFECCIÓN

Se conoce, a raíz de las comunicaciones publicadas en determinados foros *underground* por los autores de Avaddon, que éstos no proporcionan ninguna vía de distribución del *ransomware*. No obstante, recomiendan a sus clientes hacer uso de *dediks* (término derivado de la palabra "*dedicated*" y que hace referencia a equipos bajo el control de un atacante y cuyo acceso está en venta) o similares para obtener fácilmente acceso a equipos en los que pueda desplegar el código dañino.

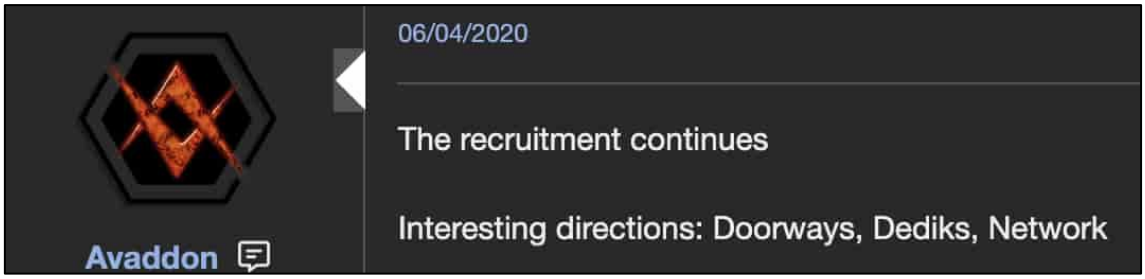


Figura 7. *Recruitment* Avaddon. Fuente: https://www.domaintools.com/

Desde junio de este año se han observado diversas campañas de *malspam* en donde los atacantes envían a las víctimas un *loader* de Avaddon en Javascript comprimido en un .zip. Dicho Javascript aparenta ser un fichero jpg, por ejemplo: *PIC124102.jpg.js*. Los atacantes, en sus correos, animan a las víctimas a abrir la imagen adjunta. Si éstos tienen habilitada la opción de "*Ocultar las extensiones de archivo para tipos de archivo conocido*"(opción por defecto) pueden pensar que el fichero adjunto se trata de una imagen al observar únicamente la extensión .jpg, dando pie a su ejecución.

En la siguiente imagen puede verse un email remitido a determinada víctima junto al *loader* en Javascript. Éste emplea *powershell* así como *bitsadmin* para descargar y ejecutar Avaddon en el sistema.

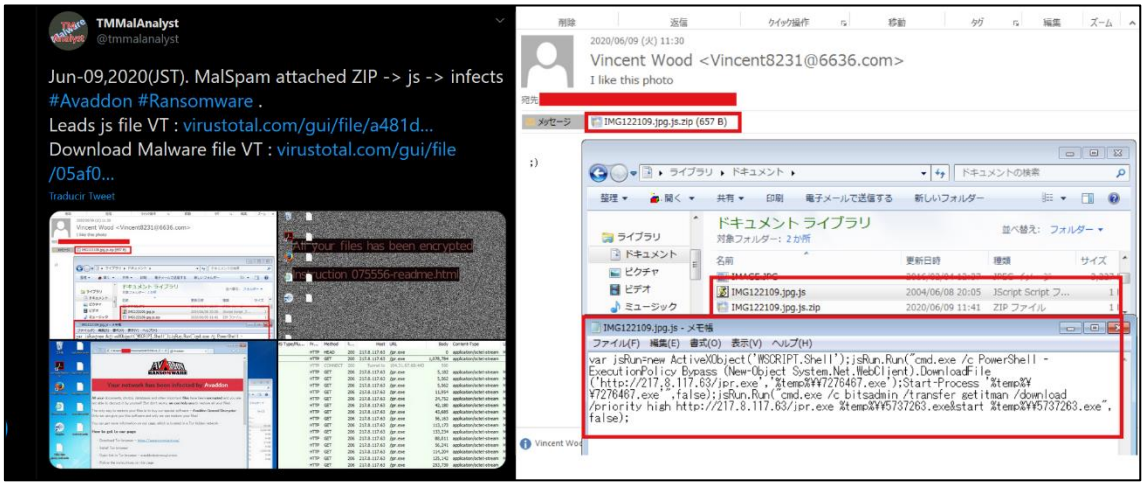


Figura 8. Malspam Avaddon. Fuente: Twitter @tmmalanalyst

Otro vector de infección utilizado por los atacantes es el uso de ficheros ofimáticos dañinos. Microsoft anunciaba [REF.- 7] en su Twitter a principios de junio el uso de documentos Excel con macros dañinas para descargar y ejecutar Avaddon en determinadas campañas contra ciudadanos italianos.

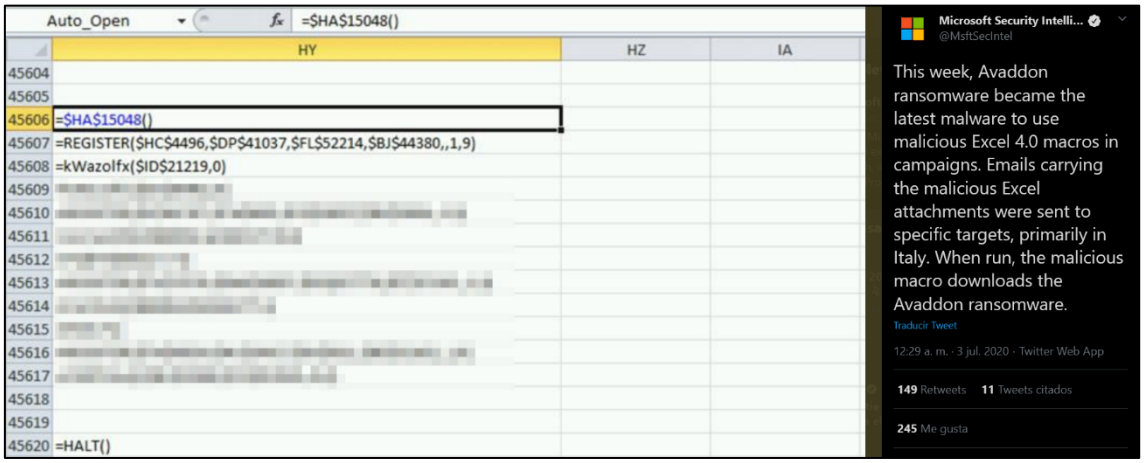


Figura 9. Macro Excel Avaddon. Fuente: Twitter @MsftSecIntel

6. CARACTERÍSTICAS TÉCNICAS

Con la ayuda de Radare puede identificarse fácilmente el *main()* del binario a partir de la dirección 0x435691.

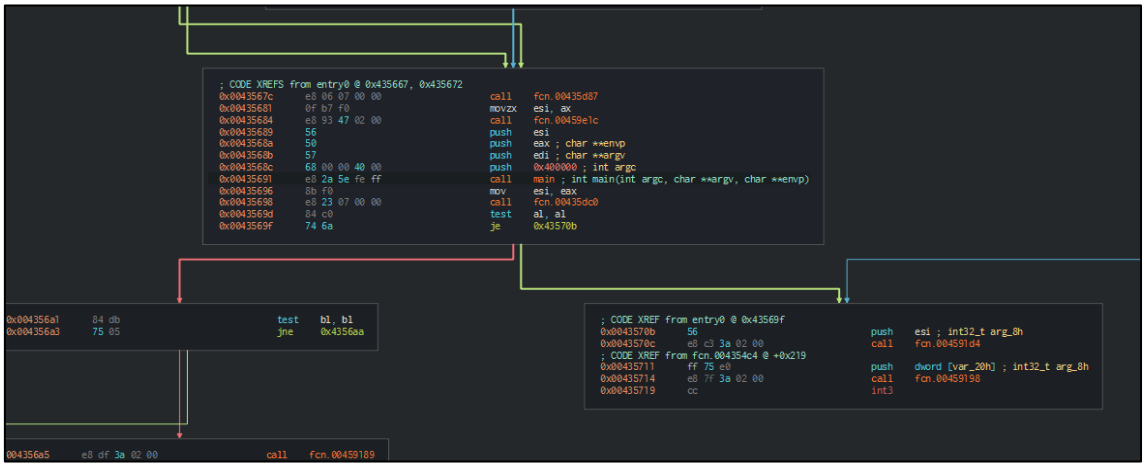


Figura 10. *main*()

El código dañino comienza resolviendo, vía *GetProcAddress*, APIs vinculados con la configuración regional del equipo: *EnumSystemLocalesEx*, *GetLocaleInfoEx*, *GetTimeFormatEx*, *GetUserDefaultLocaleName*, *IsValidLocaleName*, *LCIDToLocaleName*, *LocaleNameToLCID*, etc. Dichas APIs serán utilizadas posteriormente para extraer aspectos como el idioma, región y codificación del sistema.

```
1 int __cdecl resolve_API(int a1, LPCSTR lpProcName, int a3, int a4)
2 {
3   volatile signed __int32 *v4; // ebx
4   int v5; // edx
5   int result; // eax
6   HMODULE v7; // eax
7   FARPROC v8; // eax
8   int v9; // esi
9
10  v4 = (volatile signed __int32 *)((char *)&unk_4AB958 + 4 * a1);
11  v5 = __ROR4__(*v4 ^ __security_cookie, __security_cookie & 0x1F);
12  if ( v5 == -1 )
13    return 0;
14  if ( v5 )
15    return __ROR4__(*v4 ^ __security_cookie, __security_cookie & 0x1F);
16  v7 = sub_45C4BF((_DWORD *)a3, (_DWORD *)a4);
17  if ( v7 && (v8 = GetProcAddress(v7, lpProcName), (v9 = (int)v8) != 0) )
18  {
19    _InterlockedExchange(v4, sub_458F66((int)v8));
20    result = v9;
21  }
22  else
23  {
24    _InterlockedExchange(v4, __security_cookie ^ __ROR4__(-1, 32 - (__security_cookie & 0x1F)));
25    result = 0;
26  }
27  return result;
28}
```

Figura 11. Resolución de nombres (*GetProcAddress*)

Pese a que el binario no se encuentra empaquetado sí que dispone de múltiples *strings* ofuscados utilizando un algoritmo personalizado. En la siguiente imagen se muestran algunas de las cadenas en claro tras establecer un *breakpoint* condicional en el *return address* de la función encargada del descifrado de las mismas.

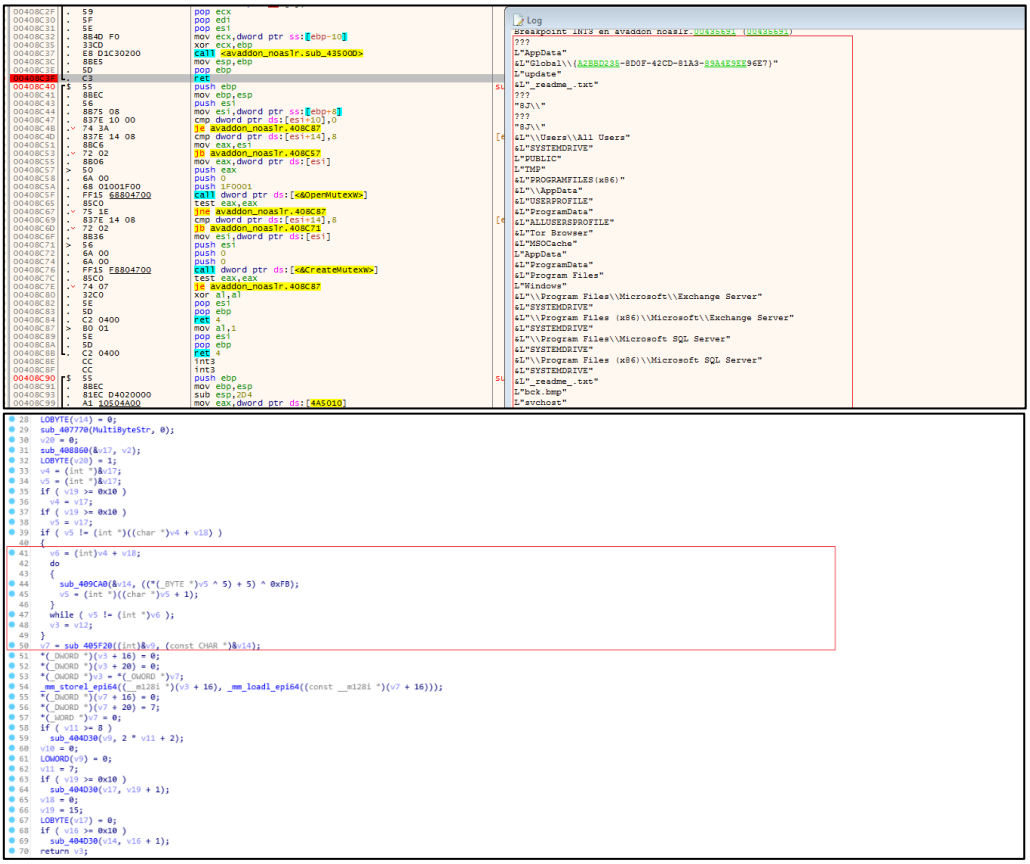


Figura 12. *Strings* cifrados / Descifrado

Entre los *strings* recuperados se puede observar el nombre de un objeto *mutex* ("**Global\\{A2BBD235-8D0F-42CD-81A3-89A4E9EE96E7}**") que será utilizado para conocer si ya existe una muestra de Avaddon en ejecución. En la siguiente imagen se muestra la lógica empleada. Se invoca en primer lugar a *OpenMutexW* y si se obtiene NULL (lo que implicaría que no existe una instancia de Avaddon ejecutándose) se crearía dicho objeto *mutex* mediante *CreateMutexW* y continuarían las acciones dañinas.

Figura 13. Gestión objeto mutex

Si, por el contrario, *OpenMutexW* devuelve un *handle* válido al objeto mutex el código dañino finalizará su ejecución (asumiendo que ya existe una instancia ejecutándose).

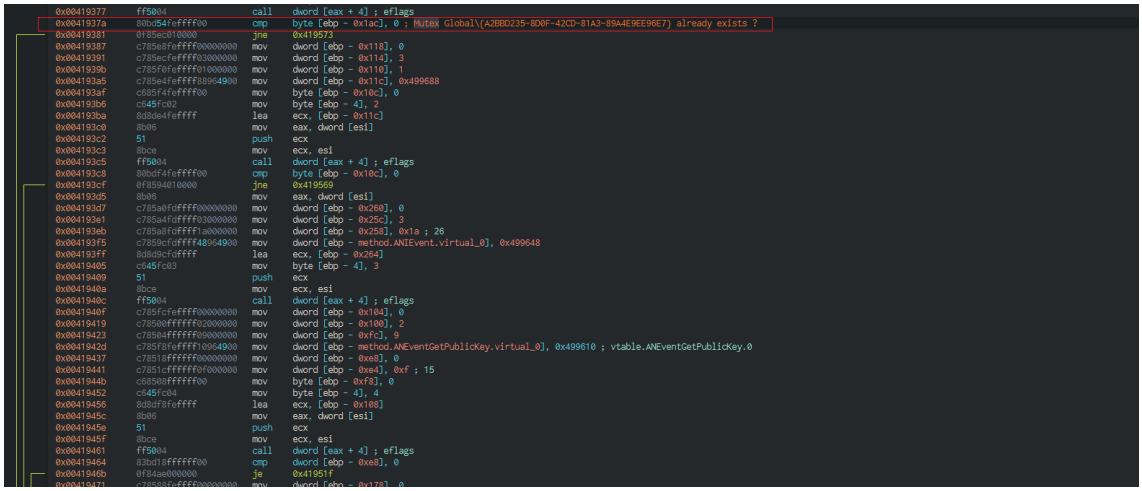


Figura 14. Existencia del objeto mutex

Otros *strings* ofuscados reflejan los procesos que posteriormente ejecutará el espécimen vía *CreateProcessW* antes de comenzar con el cifrado de ficheros con el objetivo de dificultar la recuperación de los mismos:

```
"wmic.exe SHADOWCOPY /nointeractive"
"wbadmin DELETE SYSTEMSTATEBACKUP"
"wbadmin DELETE SYSTEMSTATEBACKUP -deleteOldest"
"bcdedit.exe /set {default} recoveryenabled No"
"bcdedit.exe /set {default} bootstatuspolicy ignoreallfailures"
"vssadmin.exe Delete Shadows /All /Quiet"
```

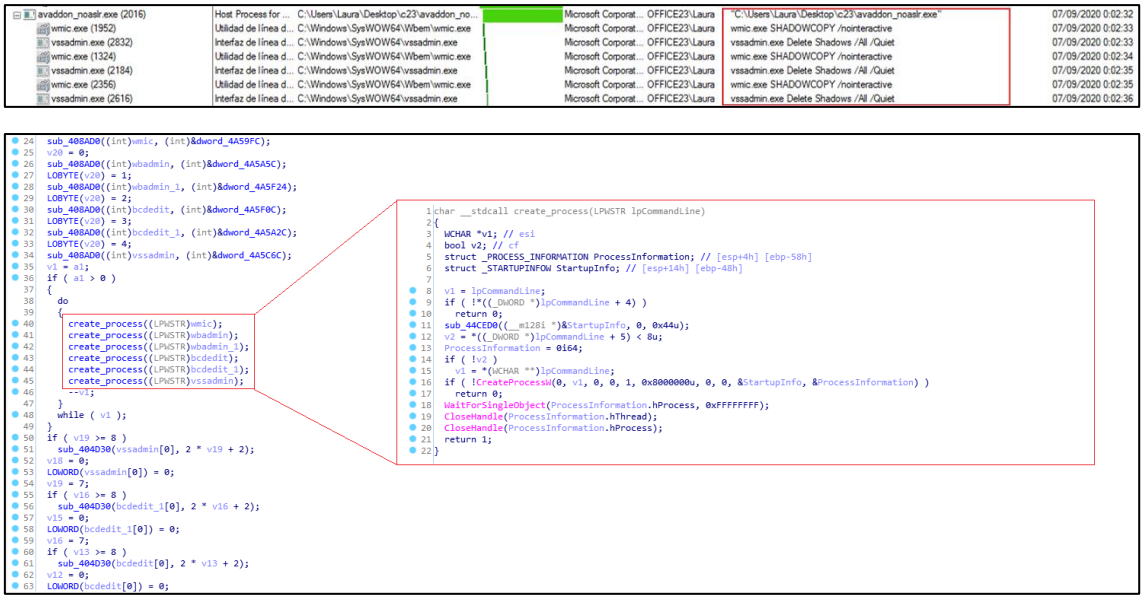


Figura 15. Procesos hijos generados

Mediante dichos procesos se deshabilitan las *shadow copy,* se eliminan las copias de seguridad del sistema, se deshabilita la reparación automática y se modifica la política de arranque (*boot status policy*) para ignorar diversos tipos de errores.

El *malware* implementa diversas técnicas *anti-debug;* por ejemplo, hace uso de APIs como *IsDebuggerPresent* y *CheckRemoteDebuggerPresent* (al cual le pasa como argumento un *handler* al propio proceso: *0xFFFFFFFF*) y comprueba el contenido de los

registros de *debug* (DR0-DR3). Si se identifica la presencia de un *debugger* finaliza la ejecución del proceso sin llevar a cabo las acciones de cifrado.

```
bool check_debugger()
{
  HANDLE v0; // eax
  HANDLE v1; // eax
  bool result; // al
  CONTEXT Context; // [esp+0h] [ebp-2D4h]
  BOOL pbDebuggerPresent; // [esp+2CCh] [ebp-8h]

  result = 1;
  if ( IsDebuggerPresent() != 1 )
  {
    pbDebuggerPresent = 1;
    v0 = GetCurrentProcess();
    if ( CheckRemoteDebuggerPresent(v0, &pbDebuggerPresent) )
    {
      if ( pbDebuggerPresent != 1 )
      {
        sub_44CED0((__m128i *)&Context.Dr0, 0, 0x2C8u);
        Context.ContextFlags = 65552;
        v1 = GetCurrentThread();
        if ( !GetThreadContext(v1, &Context) || !Context.Dr0 && !Context.Dr1 && !Context.Dr2 && !Context.Dr3 )
          result = 0;
      }
    }
  }
  return result;
}
```

Figura 16. Técnicas Anti-Debug

Posteriormente, Avaddon intenta identificar determinado tipo de servicios haciendo uso de APIs vinculadas con el *Service Control Manager* (*OpenSCManagerW*, *OpenServiceW*, *QueryServiceStatusEx*, etc.). Algunos de estos procesos se listan a continuación.

DefWatch, ccEvtMgr, ccSetMgr, ccSetMgr, SavRoam, dbsrv12, sqlservr, sqlagent, Intuit.QuickBooks.FCS, dbeng8, sqladhlp, QBIDPService, Culserver, RTVscan, vmware-usbarbitator64, vmware-converter, VMAuthdService, VMnetDHCP, VMUSBArbService, VMwareHostd, sqlbrowser, SQLADHLP, sqlwriter, msmdsrv, tomcat6, QBCFMonitorService.

Si alguno de estos servicios existe trata de finalizarlo y eliminarlo vía *DeleteService*.

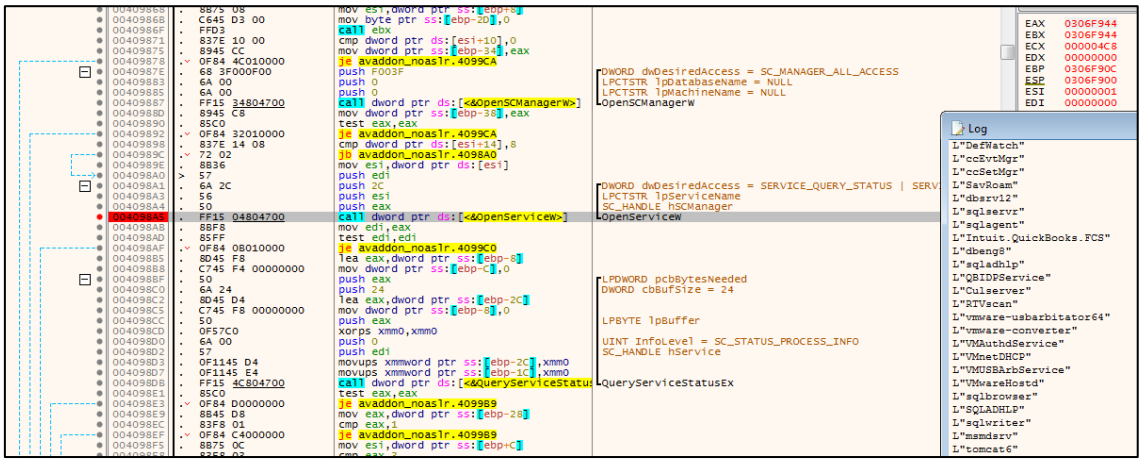


Figura 17. Listado de servicios

El código dañino también recorrerá el listado de procesos en ejecución para intentar localizar y finalizar el siguiente listado de procesos, seguramente para evitar problemas derivados del bloqueo de ficheros (*locks*) por parte de dichos programas a la hora de cifrar los mismos.

sqlservr.exe, sqlmangr.exe, RAgui.exe, QBCFMonitorService.exe, supervise.exe, fdhost.exe, Culture.exe, RTVscan.exe, Defwatch.exe, wxServerView.exe, qlbrowser.exe, winword.exe, GDscan.exe, QBW32.exe,

QBDBMgr.exe, qbupdate.exe, axlbridge.exe, 360se.exe, 360doctor.exe, QBIDPService.exe, wxServer.exe, httpd.exe, fdlauncher.exe, MsDtSrvr.exe, tomcat6.exe, java.exe, wdswfsafe.exe.

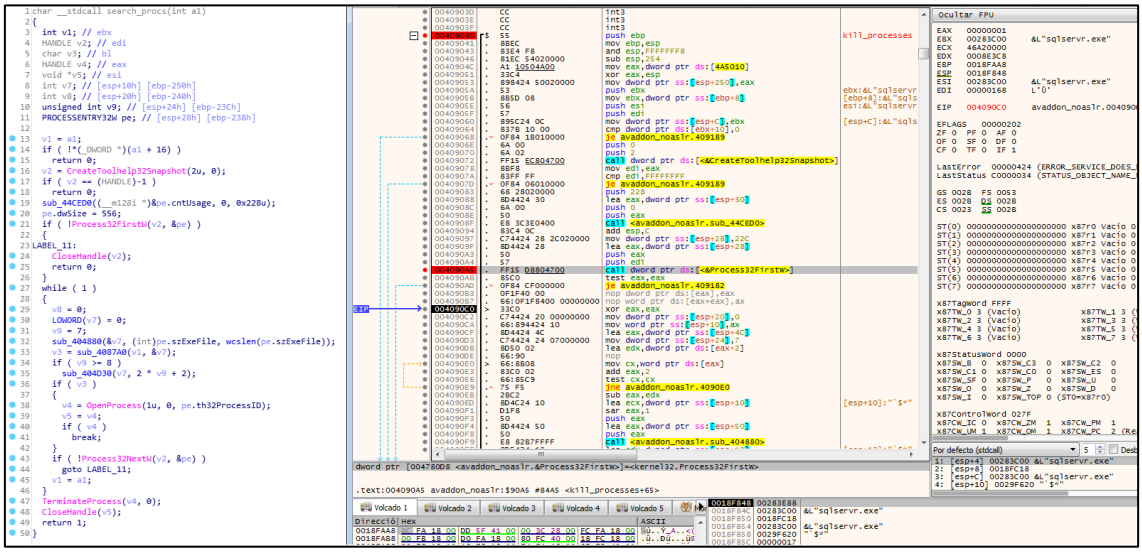


Figura 18. Búsqueda y finalización de procesos

Avaddon establece a 1 el valor de "*EnableLinkedConnections*" de la clave "*SOFTWARE*\\Microsoft*Windows**CurrentVersion**Policies**System*". Dicho valor permite modificar el comportamiento del servicio *LanmanWorkstation* y del subsistema de seguridad LSA para permitir que las unidades mapeadas sean accesibles por un proceso con un *restricted/elevated token* (en este caso por el propio proceso Avaddon ejecutándose con permisos de administrador).

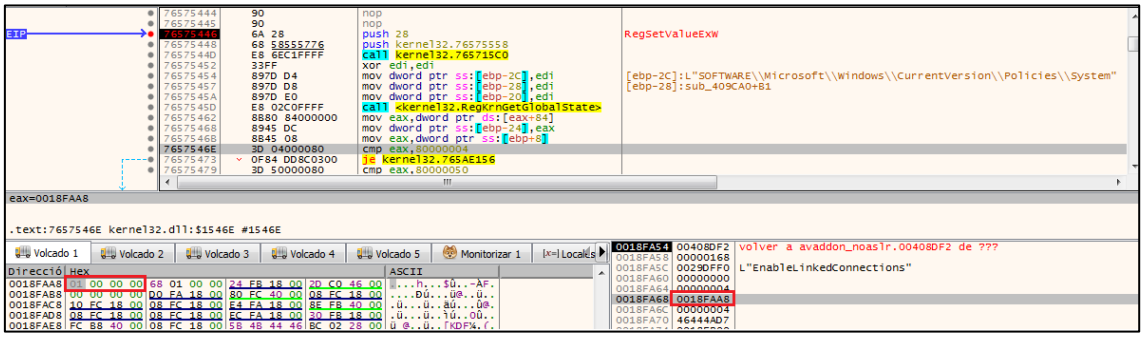
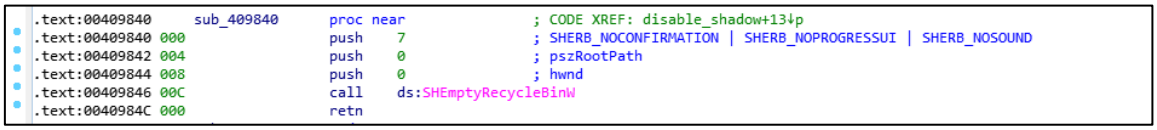


Figura 19. EnableLinkedConnections

De forma "silenciosa" (mediante los *flags* mostrados en la siguiente imagen) *también* eliminará el contenido de la papelera de reciclaje haciendo uso de *SHEmptyRecycleBinW* para dificultar su recuperación después del cifrado de ficheros.



```
.text:00409840     sub_409840      proc near               ; CODE XREF: disable_shadow+134↓p
.text:00409840 000                 push    7               ; SHERB_NOCONFIRMATION | SHERB_NOPROGRESSUI | SHERB_NOSOUND
.text:00409842 004                 push    0               ; pszRootPath
.text:00409844 008                 push    0               ; hwnd
.text:00409846 00C                 call    ds:SHEmptyRecycleBinW
.text:0040984C 000                 retn
```

Figura 20. Eliminación de ficheros de la papelera de reciclaje

El código dañino se apoya en el *Restart Manager* de Windows [REF.- 8] para forzar la finalización de procesos/servicios que pudieran estar haciendo uso de los ficheros de la víctima. Dichas acciones facilitan el "desbloqueo" de ficheros en uso por

aplicaciones legítimas para permitir al *ransomware* acceder y cifrar los mismos. La siguiente función muestra las APIs involucradas para interactuar con dicho componente de Windows.

```
22   if ( !RmStartSession(&pSessionHandle, 0, (WCHAR *)v13) )
23   {
24     if ( *(_DWORD *)(a1 + 20) >= 8u )
25       v1 = *(const WCHAR **)a1;
26     rgsFileNames = v1;
27     if ( !RmRegisterResources(pSessionHandle, 1u, &rgsFileNames, 0, 0, 0, 0) )
28     {
29       v2 = 0;
30       v3 = 0;
31       pnProcInfo = 0;
32       pnProcInfoNeeded = 0;
33       dwRebootReasons = 0;
34       while ( 1 )
35       {
36         v4 = RmGetList(pSessionHandle, &pnProcInfoNeeded, &pnProcInfo, v3, &dwRebootReasons);
37         if ( !v4 )
38           break;
39         if ( v4 != 234 )
40           return;
41         v5 = pnProcInfoNeeded;
42         pnProcInfo = pnProcInfoNeeded;
43         if ( v3 )
44         {
45           sub_43527A(v3);
46           v5 = pnProcInfo;
47         }
48         v6 = sub_43527F(668 * v5 | -(668 * (unsigned __int64)v5 >> 32 != 0));
49         v7 = v2++;
50         v3 = (RM_PROCESS_INFO *)v6;
51         if ( v7 >= 3 )
52           goto LABEL_15;
53       }
54       if ( !dwRebootReasons )
55         RmShutdown(pSessionHandle, 0, 0);
56 LABEL_15:
57       if ( v3 )
58         sub_43527A(v3);
59       if ( pSessionHandle != -1 )
60         RmEndSession(pSessionHandle);
```

Figura 21. Restart Manager (*unlock* de ficheros)

7. CIFRADO

El código dañino hace uso de la CryptoAPI de Windows con la que generará una clave AES256 vía *CryptGenKey* que utilizará para cifrar el contenido de los ficheros.

```
31   v1 = this;
32   v2 = this + 24;
33   if ( !CryptGenKey_wrapper(this, this + 24) )
34     return 0;
35   v3 = CryptExport_wrapper(*v2);
36   v4 = v3;
37   v5 = CryptEncrypt_wrapper(v1[22], 1, v3);
38   dwBufLen = v5;
39   if ( !v4 || !v5 )
40     return 0;
```

```
1 bool __thiscall CryptGenKey_wrapper(HCRYPTPROV *this, HCRYPTKEY *phKey)
2 {
3   return CryptGenKey(this[7], CALG_AES_256, 1u, phKey) != 0;
4 }
```

Figura 22. Generación y exportación AES256

Dicha *key* será cifrada con una clave pública RSA de 2048 bits importada mediante *CryptImportkey* (embebida en el propio binario).

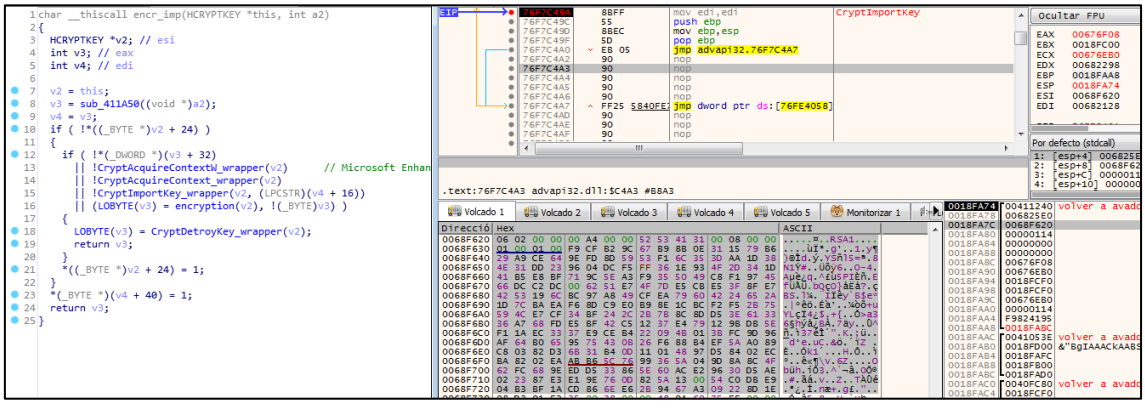


Figura 23. Importación de la key (RSA)

El cifrado de la clave AES con la clave RSA se aprecia en la siguiente imagen.

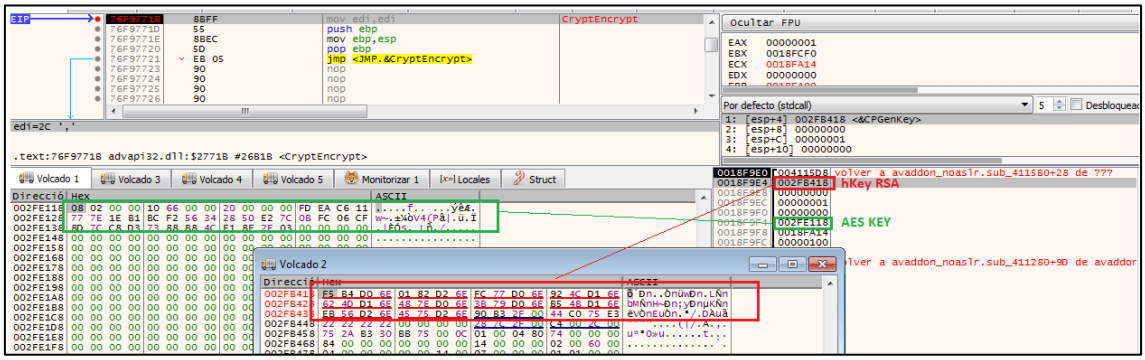


Figura 24. Cifrado AES con la clave pública

El código dañino recorrerá el sistema de ficheros cifrando el contenido de los mismos mediante la clave AES generada previamente.

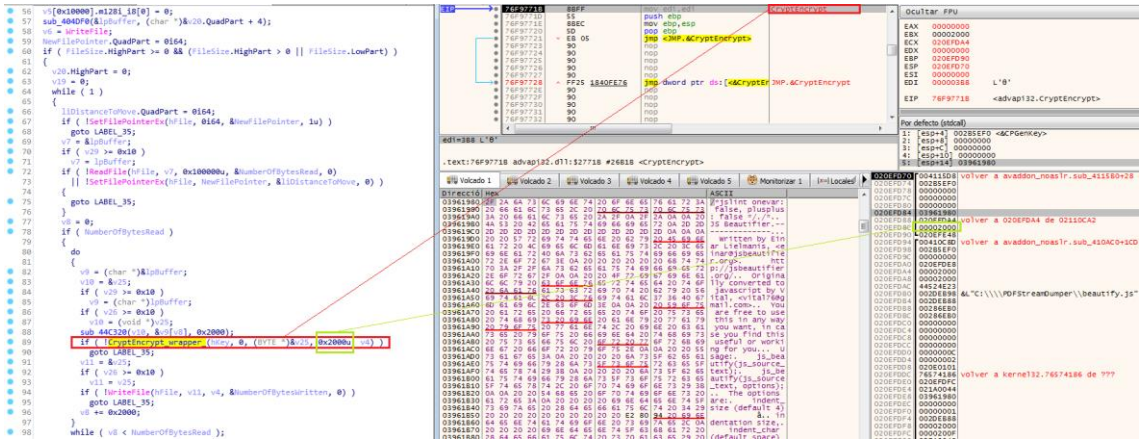
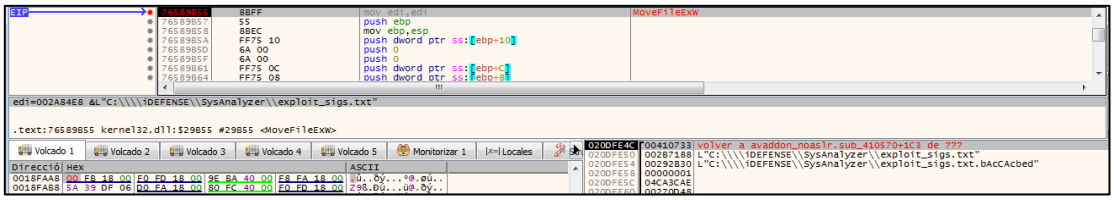


Figura 25. Ejemplo de cifrado de fichero

El fichero cifrado será posteriormente renombrado (en la siguiente imagen añadiendo .bAcCAcbed a la extensión) y dentro de cada directorio se creará un fichero de texto (*xxxxx_readme_.txt*) informando de la infección y cómo recuperar los ficheros afectados.



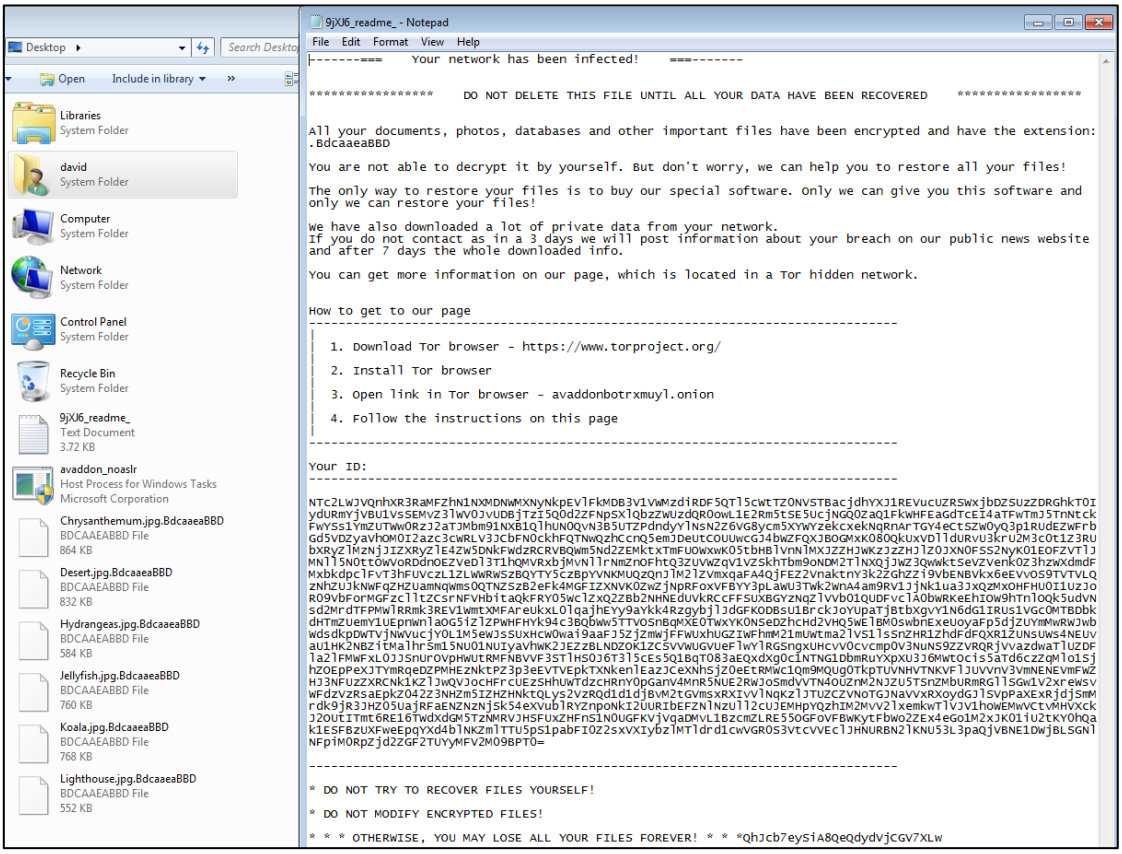


Figura 26. Renombre del fichero / Readme

El dominio referenciado en dicho fichero se corresponde con **avaddonbotrxmuyl.onion** (actualmente en funcionamiento). Ingresando el ID generado para cada cliente (incluido en el *readme*) se accede al recurso que informa de cómo recuperar los ficheros si se paga un rescate de 1500000 USD a determinada cartera BTC.

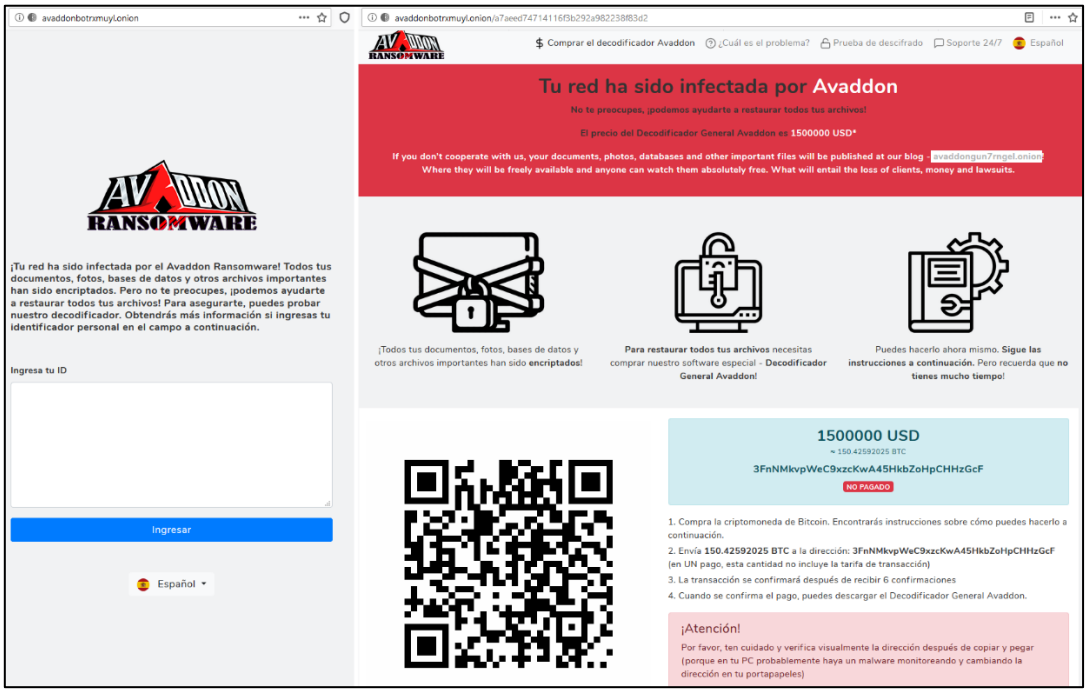


Figura 27. Dominio avaddonbotrxmuyl.onion

En el texto de dichas instrucciones se referencia un blog (accesible también en la red Tor) en donde se exponen datos de clientes, en concreto un fichero de descarga de 558 MB (*Personalne-PROKES-DUMP.zip*) correspondiente a una empresa eslovaca.

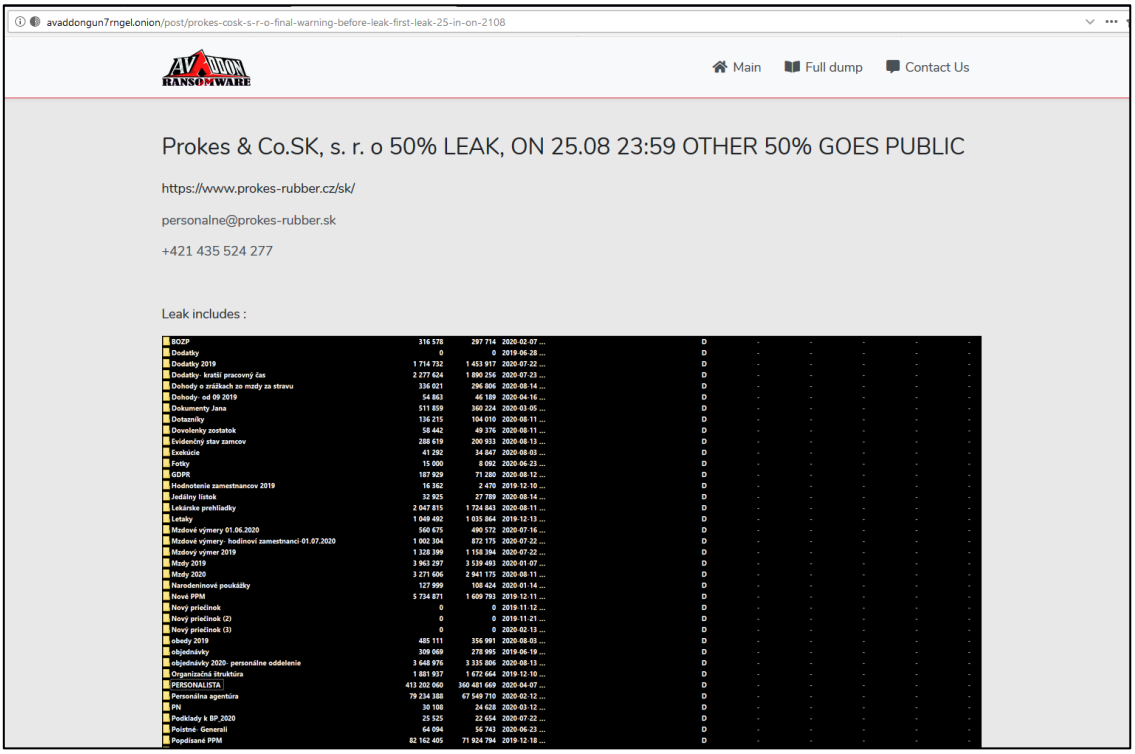


Figura 28. Leak (Prokes & Co.SK)

8. REGLAS DE DETECCIÓN

8.1 REGLA DE YARA

```
import "pe"
rule avaddon_ransom{
meta:
        description = "Avaddon ransomware rule based on some XOR-obfuscated strings"
        author = "CCN-CERT"
        version = "1.0"
        date = "2020-09-08"
        hash1 = "51d85210c2f621ca14d92a8375ee24d62f9d7f44"
strings:

        $xor_str_01 = "www.torproject.org" xor(0x01-0xff)
        $xor_str_02 = "httpd.exe" xor(0x01-0xff)
        $xor_str_03 = "fdlauncher.exe" xor(0x01-0xff)
        $xor_str_04 = "MsDtSrvr.exe" xor(0x01-0xff)
        $xor_str_05 = "tomcat6.exe" xor(0x01-0xff)
        $xor_str_06 = "RTVscan.exe" xor(0x01-0xff)
        $xor_str_07 = "RAgui.exe" xor(0x01-0xff)
      $xor_str_08 = "sqlmangr.exe" xor(0x01-0xff)
        $xor_str_09 = "qbupdate.exe" xor(0x01-0xff)
        $xor_str_10 = "QBW32.exe" xor(0x01-0xff)

condition:
        uint16(0) == 0x5A4D and 6 of ($xor_*) and
        filesize < 1MB and pe.number_of_signatures == 1
}
```

9. REFERENCIAS

[REF.- 1] New Avaddon Ransomware launches in massive smiley spam campaign:

https://www.bleepingcomputer.com/news/security/new-avaddon-ransomware-launches-in-massive-smiley-spam-campaign/

[REF.- 2] Commonwealth of Independent States:

https://en.wikipedia.org/wiki/Commonwealth_of_Independent_States

[REF.- 3] Phorphiex/Trik Botnet Delivers Avaddon Ransomware:

https://appriver.com/resources/blog/june-2020/phorphiextrik-botnet-delivers-avaddon-ransomware

[REF.- 4] Malware bazaar:

https://bazaar.abuse.ch/browse/tag/Gold%20Stroy%20SP%20Z%20O%20O/

[REF.- 5] Twitter @CryptoInsane:

https://twitter.com/CryptoInsane/status/1268440645770805248

[REF.- 6] Avaddon ransomware launches data leak site to extort victims:

https://www.bleepingcomputer.com/news/security/avaddon-ransomware-launches-data-leak-site-to-extort-victims/

[REF.- 7] Twitter @MsftSecIntel:

https://twitter.com/MsftSecIntel/status/1278817988867575808

[REF.- 8] Using Restart Manager:

https://docs.microsoft.com/en-us/windows/win32/rstmgr/using-restart-manager