Basque CyberSecurity Centre

BianLian Ransomware

BCSC-MALWARE-BIANLIAN

TLP: CLEAR

www.ciberseguridad.eus

EUSKO JAURLARITZA GOBIERNO VASCO

GRUPO spri TALDEA

Índice

Cláusula de exención de responsabilidad

El presente documento se proporciona con el objeto de divulgar las alertas que el BCSC considera necesarias en favor de la seguridad de las organizaciones y de la ciudadanía interesada. En ningún caso el BCSC puede ser considerado responsable de posibles daños que, de forma directa o indirecta, de manera fortuita o extraordinaria pueda ocasionar el uso de la información revelada, así como de las tecnologías a las que se haga referencia tanto de la web de BCSC como de información externa a la que se acceda mediante enlaces a páginas webs externas, a redes sociales, a productos de software o a cualquier otra información que pueda aparecer en la alerta o en la web de BCSC. En todo caso, los contenidos de la alerta y las contestaciones que pudieran darse a través de los diferentes correos electrónicos son opiniones y recomendaciones acorde a los términos aquí recogidos no pudiendo derivarse efecto jurídico vinculante derivado de la información comunicada.

Cláusula de prohibición de venta

Queda terminantemente prohibida la venta u obtención de cualquier beneficio económico, sin perjuicio de la posibilidad de copia, distribución, difusión o divulgación del presente documento.

Sobre el BCSC

El Centro Vasco de Ciberseguridad (Basque Cybersecurity Centre, BCSC) es la entidad designada por el Gobierno Vasco para elevar el nivel de madurez de la ciberseguridad en Euskadi.

Es una iniciativa transversal que se enmarca en la Agencia Vasca de Desarrollo Empresarial (SPRI), sociedad dependiente del Departamento de Desarrollo Económico, Sostenibilidad y Medio Ambiente del Gobierno Vasco. Así mismo, involucra a otros tres Departamentos del Gobierno Vasco: el de Seguridad, el de Gobernanza Pública y Autogobierno, y el de Educación, y a cuatro agentes de la Red Vasca de Ciencia, Tecnología e Innovación: Tecnalia, Vicomtech, Ikerlan y BCAM.

El BCSC es la entidad de referencia para el desarrollo de la ciberseguridad y de la confianza digital de ciudadanos, empresas e instituciones públicas en Euskadi, especialmente para los sectores estratégicos de la economía de la región.

La misión del BCSC es por tanto promover y desarrollar la ciberseguridad en la sociedad vasca, dinamizar la actividad empresarial de Euskadi y posibilitar la creación de un sector profesional que sea referente. En este contexto se impulsa la ejecución de proyectos de colaboración entre actores complementarios en los ámbitos de innovación tecnológica, investigación y transferencia tecnológica a la industria de fabricación avanzada y otros sectores.

Así mismo, ofrece diferentes servicios en su rol como Equipo de Repuesta a Incidentes (en adelante CERT, por sus siglas en inglés “Computer Emergency Response Team”) y trabaja en el ámbito de la Comunidad Autónoma del País Vasco para aumentar la capacidad de detección y alerta temprana de nuevas amenazas, la respuesta y análisis de incidentes de seguridad de la información, y el diseño de medidas preventivas para atender a las necesidades de la sociedad vasca. Con el fin de alcanzar estos objetivos forma parte de diferentes iniciativas orientadas a la gestión de incidentes de ciberseguridad:

FIRST Improving Security Together
TF-CSIRT Trusted Introducer
CSIRT.es

Resumen ejecutivo

BianLian es un *malware* de tipo *ransomware* identificado por primera vez a mediados de julio de 2022 y que ha afectado a varias organizaciones en sectores como la fabricación, la educación, la atención médica y las finanzas. Cabe destacar que existe otro troyano bancario para sistemas Android al que algunos investigadores se han referido también como *BianLian* o *Hydra* pero, hasta la fecha, no se hay indicios de que esté relacionado con el grupo de ransomware.

Los actores detrás de *BianLian* siguen un modelo de doble extorsión, muy común en otras familias de ransomware en el que, tras el compromiso de la red de la víctima, exfiltran cierta información sensible de la compañía previamente al cifrado de los equipos y amenazan con publicar esa información como mecanismo para presionar a la víctima a pagar el rescate solicitado. Para ello, albergan un sitio web en la red Tor donde informan de las filtraciones y ponen a disposición pública la descarga de los datos una vez concluido el plazo en caso de no haberse producido el pago. Desde su puesta en macha, el número de víctimas publicadas en dicho sitio ha ido aumentando significativamente, lo que lo convierte en un actor persistente en el tiempo y una amenaza actual.

El código de *BianLian* está desarrollado en el lenguaje de programación Go, un lenguaje que está cogiendo cierta popularidad entre los desarrolladores de malware por su capacidad multiplataforma y gran cantidad de librerías disponibles que plantea algunos desafíos iniciales, pero no insuperables, de ingeniería inversa. No obstante, pese a dicha capacidad multiplataforma del lenguaje, las muestras identificadas de este ransomware únicamente tienen por objetivo sistemas Windows de 64 bits.

A diferencia de otros ransomware criptográficamente seguros, el sistema de cifrado de *BianLian* se basa exclusivamente en AES-256 en modo CBC. Cada binario contiene una clave y vector de inicialización (IV) únicos incrustados en el código que son utilizados para cifrar todos los ficheros del sistema víctima y que solo se cifran de forma parcial para acelerar el proceso. Por este motivo, en enero de 2023 la firma antivirus Avast publicó un descifrador para esta familia de ransomware basado en las muestras que han podido localizarse hasta la fecha. A consecuencia de ello, el grupo está cambiando sus ataques recientes a un modelo de extorsión basada en el robo de datos sin cifrado de los sistemas. Esta táctica sigue siendo convincente, ya que las filtraciones de datos provocan daños en la reputación de la víctima que socavan la confianza del cliente e introducen complicaciones legales.

Pese a la debilidad criptográfica que presenta el ransomware, para poder llegar a descifrar los ficheros sin necesidad de pagar el rescate demandado es necesario conocer los valores de la clave y el IV, que son únicos por cada binario de *BianLian*. El descifrador publicado por Avast únicamente contiene algunos de

los valores encontrados en diferentes muestras hasta el momento de su publicación, por lo que podría no servir para nuevas víctimas de esta amenaza mientras no se actualice con nuevos valores. Además, hay que tener en cuenta que BianLian tiene la capacidad de autoborrarse una vez termina el proceso de cifrado por lo que, en caso de que el descifrador de Avast no funcionase, sería esencial disponer del binario que ha cifrado los ficheros para poder extraer de aquí el valor de inicialización.

Análisis técnico

Flujo de infección

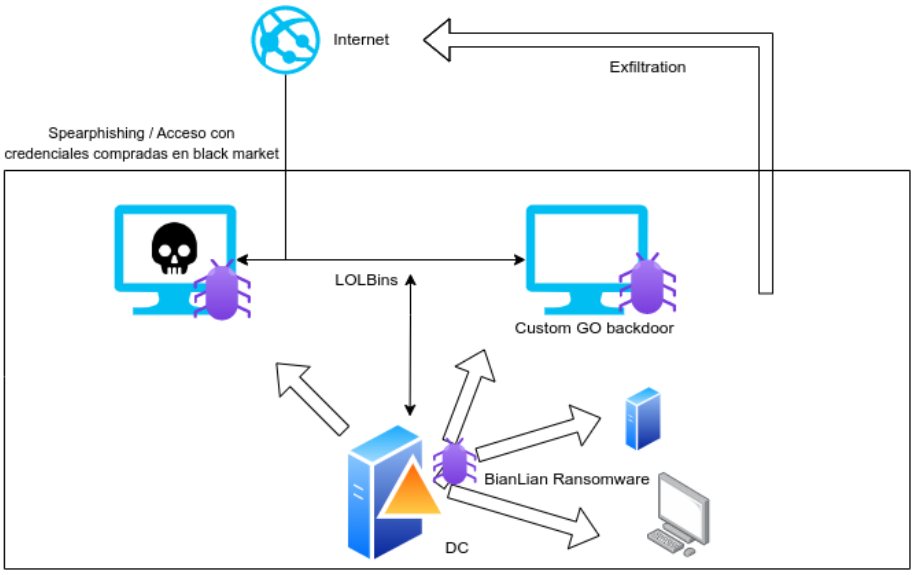


Ilustración 1: Flujo de infección de BianLian Ransomware.

El análisis e información compartida por la comunidad sobre diferentes casos que involucran la ejecución de BianLian indican que, al igual que ocurre en la mayoría de casos de ransomware, ésta es la última fase de un proceso de intrusión previo.

Los actores encargados de las operaciones de intrusión relacionadas con este malware utilizan técnicas sutiles para explotar, enumerar y moverse lateralmente en las redes de las víctimas y permanecer sin ser detectados. Sin embargo, posteriormente pasan a un perfil mucho más agresivo en el que buscan contrarrestar las protecciones de detección y respuesta de *endpoints* (EDR) a la hora de lanzar el ransomware como fase final para cifrar el mayor número de equipos posible.

Como vector de entrada inicial, los atacantes parecen apuntar a los sistemas expuestos de Protocolo de Escritorio Remoto (RDP), a los que acceden mediante credenciales válidas, posiblemente compradas a intermediarios de acceso inicial o adquiridas a través de *phishing*. Una vez dentro de la organización, los actores detrás de BianLian utilizan diversas herramientas como una puerta trasera

personalizada también escrita en Go, herramientas comerciales de acceso remoto y de línea de comandos o scripts diversos para el reconocimiento de la red.

La última etapa consiste en extraer los datos que consideren más sensibles desde los equipos de la víctima a través del Protocolo de Transferencia de Archivos (FTP), la herramienta Rclone o el servicio de alojamiento de archivos Mega. Con el fin de evitar ser detectados, utilizan PowerShell y CMD para deshabilitar los procesos en ejecución asociados con las herramientas antivirus o, alternativamente, manipulando el Registro de Windows. Finalmente, el grupo despliega el ransomware *BianLian*, que cifra los datos críticos de la organización y exige un rescate en criptomoneda para su liberación.

En resumen, los actores detrás de *BianLian* utilizan un flujo de infección no relativamente sofisticado, pero sí efectivo que combinas vectores de entrada como el acceso mediante credeciales conocidas obtenidas de forma no legítima, diferentes tipologías de malware tanto propio como utilidades ya conocidas para establecer persistencia y moverse lateralmente y, finalmente, el ransomware BianLian para comprometer y extorsionar a las organizaciones. Es importante que las organizaciones implementen medidas de seguridad proactivas, como la capacitación del usuario y la aplicación de parches de seguridad, para evitar ser víctimas de este tipo de ataque.

Portal de BianLian en la red TOR

Los actotes de ***BianLian*** cuentan con un sitio en la red TOR para enumerar las organizaciones supuestamente afectadas por su ransonware y ofrecer enlaces de descarga de los datos recopilados por ellos en caso de no pagar el rescate demandado.

La dirección actual para acceder a este sitio es la siguiente:

hxxp://bianlianlbc5an4kgnay3opdemgcryg2kpfcbgczopmm3dnbz3uaunad.onion

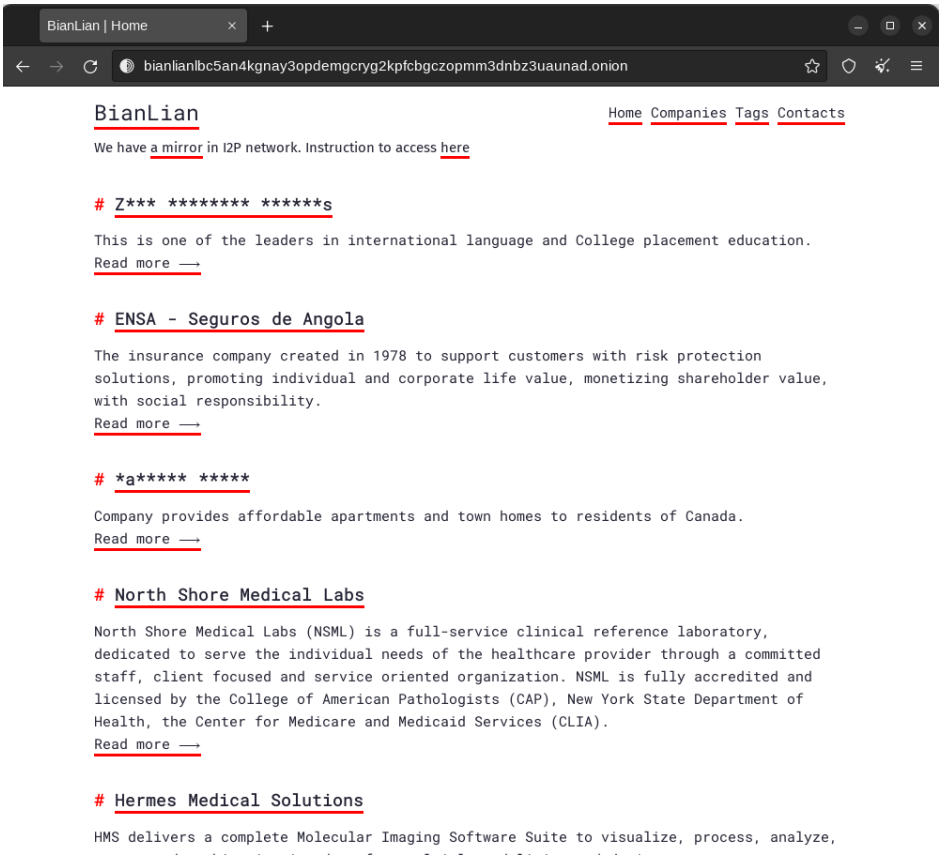


Ilustración 2: Sitio oficial de los actores de BianLian en la red TOR

En este sitio además se indican las direcciones de correo utilizadas por el grupo para comunicarse actualmente.

swikipedia@onionmail.org	swikipedia.reserve@onionmail.org

Muestra analizada

La muestra analizada corresponde a la familia ***BianLian*** y se trata de un binario Portable Ejecutable (PE) de Windows 64 bits identificado por la firma SHA256 siguiente:

3a2f6e614ff030804aa18cb03fcc3bc357f6226786efb4a734cbe2a3a1984b6f

El binario está desarrollado con Go y no parece encontrarse empaquetado mediante ningún software de protección.

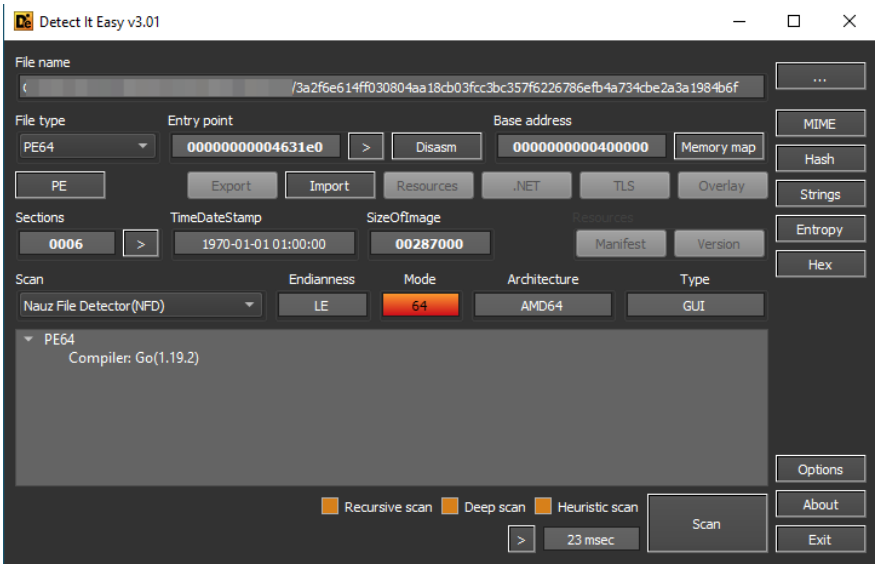


Ilustración 3: análisis de la muestra de BianLian en el software Detect It Easy (DIE)

Go Build ID y versión

Cuando se compila un programa Go, genera un *BuildID*. Un *BuildID* en Go es una representación única del archivo y su contenido. El *BuildID* se encuentra en los primeros 32Kb del archivo binario, aunque la posición exacta puede variar según el sistema operativo para el que se compiló. El *BuildID*, a su vez, está segmentado en dos partes: actionID y ContentID. El actionID es un hash de las entradas que produjeron los paquetes o el binario, y el ContentID es el hash de la salida de la compilación (el propio binario)[1]. El buildID del binario analizado es el siguiente:

N6JR_TjppN3aqqMejWWN/FWQwXkZ6aDL8DSaJQVsL/t-5svN4m62FL5YnskEfg/hU62x8xdhUJ664FDmvhi

El comando utilizado para compilar el programa también se almacena en el binario. Para compilar un programa con Go, usa el comando "*go build*". Este comando compila todos los paquetes y dependencias necesarios para la aplicación.

Para la muestra analizada, esta ruta de compilación recibió dos argumentos: *gcflags* y *trimpath*.

- -gcflags: relaciona los flags pasados al compilador

[1] https://github.com/golang/go/blob/master/src/cmd/go/internal/work/buildid.go

- -trimpath: elimina todas las rutas absolutas del sistema de archivos del ejecutable. Esto es un intento de eliminar los directorios de rutas del usuario.

```
build   -gcflags=all=-trimpath=/home/jack/Projects/project1/crypt97
```

De este comando destaca la última parte de la ruta utilizada: "crypt97". Esto puede significar diferentes iteraciones en el desarrollo, ya que este número va cambiando entre diferentes muestras o, simplemente que, por cada cliente a cifrar, podría estar creando un nuevo directorio, cambiando el número, modificando los valores necesarios y realizando ahí la compilación del nuevo binario.

El autor ha empaquetado la mayoría de funcionalidades del ransomware en un paquete llamado “project1”. Un paquete en Go es una colección de archivos fuente en el mismo directorio que se compilan juntos. El análisis de estas cadenas de caracteres y las de la rutina principal (main) ya aportan una pequeña idea de la funcionalidad del ransomware.

Ilustración 4: nombre de las principales funciones utilizadas en el binario

Argumentos de línea de comandos

La rutina principal de los programas compilados en Go se puede localizar mediante el nombre ***main_main***. Tras inspeccionar el código de dicha rutina, se puede comprobar que las primeras instrucciones de ésta están destinadas a manejar los diferentes parámetros que acepta por línea de comandos el programa.

```
v80 = e_string;
flag__FlagSet_Var(
  (__int64)flag_CommandLine,
  (__int64)go_itab__flag_stringValue_flag_Value,
  (__int64)e_string,
  (__int128 *)"e",
  1LL,
  (__int64)"Entry path",
  10LL,
  v3,
  v4);
debug_value = 0;
flag__FlagSet_Var(
  (__int64)flag_CommandLine,
  (__int64)off_56E8C8,
  (__int64)&debug_value,
  (__int128 *)"debug",
  5LL,
  0LL,
  0LL,
  v5,
  v6);
if ( !qword_6251E8 )
  runtime_panicSliceB(1, (__int64)off_56E8C8, 0, qwor
```

Ilustración 5: tratamiento de los argumentos de entrada en main_main

Como se puede observar, el programa acepta dos argumentos:

- **-e**: sirve para indicarle al programa a partir de qué ruta se desea cifrar. Si no se especifica, cifrará desde el directorio donde se encuentre el binario.
- **-debug**: sirve para indicarle al programa que escriba un fichero de logs con información de depuración sobre la ejecución.

```
if ( debug_value )
{
  v12 = 438;
  v79 = os_OpenFile(
          (unsigned int)"C:\\Users\\Public\\MicrosoftEdge_38728293_logs.txt",
          47,
          1089,
          438,
          v11,
          v14,
          v15,
          v16,
          v17,
          v53,
          v61);
  v81[0] = MEMORY[0x37];
  v81[1] = v18;
  log_Println((unsigned int)v81, 1, 1, 438, v11, v19, v20, v21, v22, v54, v62);
  log__Logger_SetOutput(qword_624040, (int)&off_56E6A0, v79, 438, v11, v23, v24, v25, v26, v5
}
```

Ilustración 6: creación del fichero de log en caso de que se active la opción

Ejecución mediante hilos para acelerar el proceso de cifrado

Tras la comprobación de los parámetros de entrada, *BianLian* pasa directamente a lanzar diferentes hilos para recorrer los directorios del equipo y cifrar sus ficheros. Para coordinarlos y esperar a que terminen todos los hilos de ejecutarse, el binario hace uso de la API de Go *WaitGroup* de la librería *sync* tras lo cual lanza las rutinas ***main_ScanForFiles***.

```
}
channel_var1 = runtime_makechan((unsigned int)&RTYPE_chan_bool, 200, v13, v12, v11, v14, v15, v16, v17, v59, v64);
sync_wait_group = (sync_WaitGroup *)runtime_newobject(&RTYPE_sync_WaitGroup);
main_ScanForFiles(*entry_path, entry_path[1], channel_var1, (signed __int64)sync_wait_group, 1, v27, v28, v29, v30);
sync__WaitGroup_Wait(sync_wait_group);
main_ScanForFiles(*entry_path, entry_path[1], channel_var1, (signed __int64)sync_wait_group, 2, v31, v32, v33, v34);
sync__WaitGroup_Wait(sync_wait_group);
data = (void *)entry_path[1];
main_ScanForFiles(*entry_path, (__int64)data, channel_var1, (signed __int64)sync_wait_group, 3, v36, v37, v38, v39);
sync__WaitGroup_Wait(sync_wait_group);
v46 = os_Executable(v40, (int)data, v41, (int)sync_wait_group, 3, v42, v43, v44, v45);
if ( v51 )
```

Ilustración 7: llamada a la rutina main_ScanForFiles mediante hilos

Borrado del programa tras acabar la ejecución

Una vez termina el proceso de cifrado, *BianLian* obtiene la ruta completa que apunta al propio binario mediante la API de Go *Executable* de la librería *os* y mediante la API *exec.Command* ejecuta el comando *del* de CMD al que le indica la ruta al binario previamente obtenida. De esta forma, se consigue borrar el binario del sistema una vez terminada su ejecución.

```
v46 = os_Executable(v40, (int)data, v41, (int)sync_wait_group, 3, v42, v43, v44, v45);// Path del binario que ha lanzado el proceso
if ( v51 )
{
  if ( !qword_6251E8 )
    goto LABEL_13;
  v46 = *(_QWORD *)qword_6251E0;
  data = *(void **)(qword_6251E0 + 8);
}
v76[0] = "/c";
v76[1] = 2LL;
v76[2] = "del";
v77[0] = 3LL;
v77[1] = v46;
v77[2] = data;
v52 = (exec_Cmd *)os_exec_Command(
                    (unsigned int)"cmd",
                    3,
                    (unsigned int)v76,
                    3,
                    3,
                    v47,
                    v48,
                    v49,
                    v50,
                    v62,
                    v67,
                    v69,
                    v70);
v58 = os_exec__Cmd_Start(v52);
```

Ilustración 8: borrado del programa tras su ejecución

Escaneo de directorios

En caso de no especificarse el parámetro correspondiente para cifrar únicamente un directorio concreto, la rutina ***main_ScanForFiles*** escanea el sistema en busca de unidades de disco mediante la llamada a la rutina ***project1_common_GetDrives***.

```
v41 = a3;
v43 = *(_QWORD *)&a4;
if ( !entry_path )
{
  result = project1_common_GetDrives((__int64)a1, 0LL, a3, a4);
  if ( a4 )
  {
    if ( debug_enabled )
    {
      v36 = result;
      v37 = v9;
      a4 = *(_QWORD *)(a4 + 8);
      v37 = *(_QWORD *)&a4;
      log_Println((unsigned int)&v37, 1, 1, a4, *((_DWORD *)&a4 + 2), v11, v12, v13, v14, v33, v34);
      return v36;
    }
  }
  return result;
}
v15 = off_61CBF0;
if ( qword_61CBF8 != 1 )
  goto LABEL_11;
v39 = (__int64)a1;
if ( !(unsigned __int8)runtime_memequal((char *)a1 + (_QWORD)entry_path - 1, off_61CBF0, 1LL) )
{
  a1 = (__int64 (__golang *)(__int64, __int64, __int64, __int64, __int64, int, int, int, int, char))v3
```

Ilustración 9: llamada a la rutina de búsqueda de unidades de disco

Aquí, el binario recorre todas las posibilidades de nombres para unidad de disco, desde la “A” a la “Z” para determinar si se encuentra montada y proceder a su inclusión en el listado a cifrar.

```
// project1/common.GetDrives
__int64 __golang project1_common_GetDrives(__int64 a1, __int64 a2, __int64 a3, __int64 a4)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  while ( (unsigned __int64)&v32 <= *(_QWORD *)(v4 + 16) )
    a1 = runtime_morestack_noctxt(a1);
  p__4_uint16 = (_4_uint16 *)runtime_newobject(&RTYPE__4_uint16);
  v32 = p__4_uint16;
  *(_QWORD *)p__4_uint16 = 0x5C003A0000LL;
  drive_letter_i = 'A';
  v8 = 0LL;
  while ( drive_letter_i <= 'Z' )
  {
    v29 = drive_letter_i;
    v30 = v8;
    *(_WORD *)p__4_uint16 = drive_letter_i;
    project1_common_GetDriveType((uintptr)p__4_uint16, 4LL, 4LL, a4, 0);
    v31 = v9;
    if ( qword_624D10 != 4 || !(unsigned __int8)runtime_ifaceeq(4LL, v9, qword_624D18) )
    {
      v14 = syscall_UTF16ToString((_DWORD)v32, 4, 4, a4, 0, v10, v11, v12, v13);
      v33 = v5;
      v34 = v5;
      v20 = runtime_convTstring(v14, 4, v15, a4, 0, v16, v17, v18, v19);
      *(_QWORD *)&v33 = &RTYPE_string;
      *((_QWORD *)&v33 + 1) = v20;
      *(_QWORD *)&v34 = MEMORY[0xC];
      *((_QWORD *)&v34 + 1) = v31;
      fmt_Errorf(
        (unsigned int)"error getting type of: %s: %s",
```

Ilustración 10: búsqueda de unidades de disco a cifrar

En función de si se especificó el parámetro -e o se han escaneado todos los discos, se llama a una nueva función ***main_ScanForFiles_func1*** o ***main_ScanForFiles_func2***.

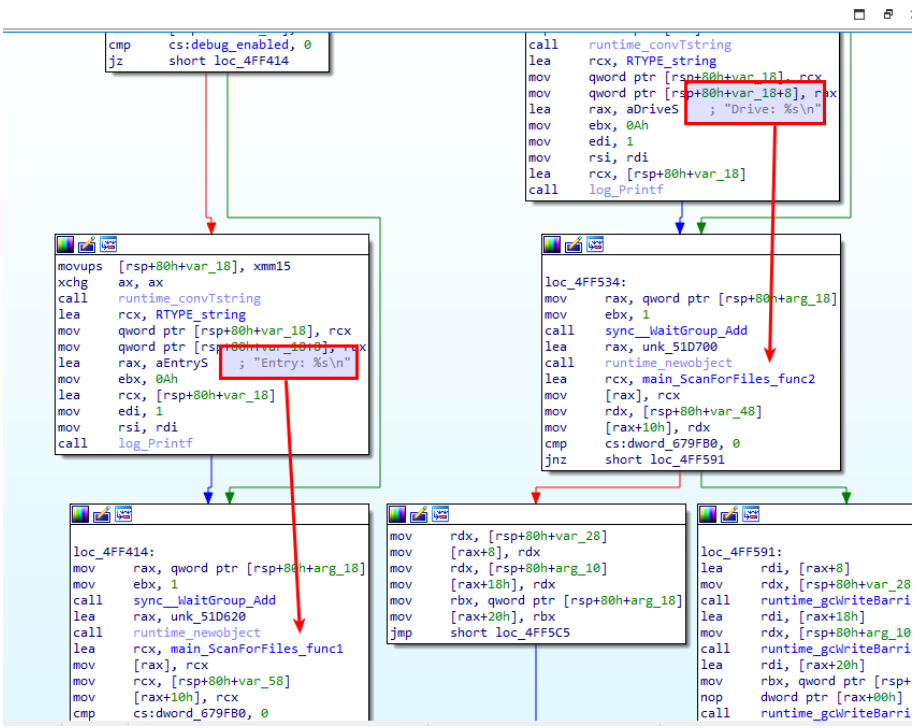


Ilustración 11: procesado de los directorios del "entry path" o disco, en función de la opción elegida

No obstante, el código de ambas funciones es idéntico y su tarea es la de llamar a una nueva función: ***main_ProcessDirectory***.

```
// main.ScanForFiles.func2
__int64 __golang main_ScanForFiles_func2(
        __int64 a1,
        __int64 a2,
        __int64 a3,
        __int64 a4,
        __int64 a5,
        int a6,
        int a7,
        int a8,
        int a9,
        char a10)
{
  _QWORD *v10; // rdx
  __int64 v11; // r14
  char **v12; // r12
  __int64 v14[5]; // [rsp-28h] [rbp-30h] BYREF
  void *retaddr; // [rsp+8h] [rbp+0h] BYREF

  if ( (unsigned __int64)&retaddr <= *(_QWORD *)(v11 + 16) )
    runtime_morestack(a4, a5);
  v12 = *(char ***)(v11 + 32);
  if ( v12 && *v12 == &a10 )
    *v12 = (char *)v14;
  return main_ProcessDirectory(
           v10[1],
           v10[2],
           v10[3],
           v10[4],
           v10[5],
           a6,
           a7,
           a8,
           a9,
           v14[0],
           v14[1],
           v14[2],
           v14[3],
           v14[4]);
}
```

Ilustración 12: llamada a main_ProcessDirectory desde main_ScanForFiles_func1 y 2

Por cada entrada encontrada aquí se comprueba si se trata de un directorio o un fichero, y se comprueba el nombre del directorio o extensión del fichero para identificar si es posible cifrarlo o se encuentra en el listado de nombres y extensiones a evitar cifrar.

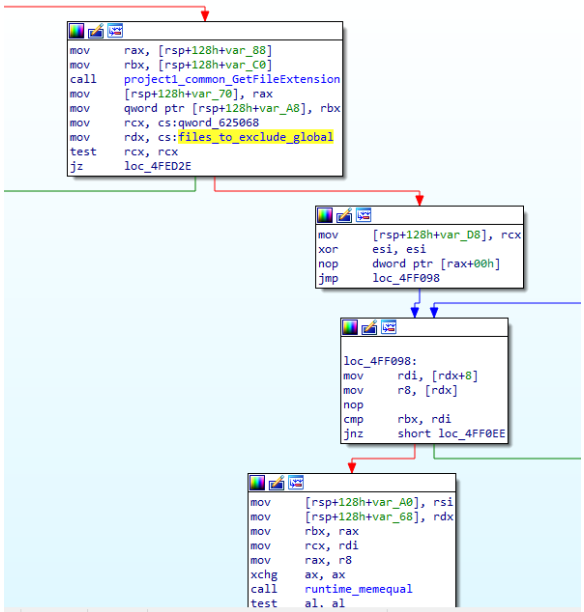


Ilustración 13: comprobación de la extensión del fichero contra el listado de extensiones en lista negra

Este listado es definido en la rutina main_init0 al inicializar el programa y para la muestra analizada es el siguiente:

- Extensiones en lista negra:

.exe, .sys, .drv, .dll, .html, .txt, .bianlian, .mui, .lnk

- Directorios en lista negra:

Windows, AppData\\Local\\Microsoft, Webroot, Sophos

```
22    a1 = runtime_morestack_noctxt(a1);
23  p__9_string = (_9_string *)runtime_newobject(&RTYPE__9_string);
24  (*p__9_string)[0].len = 4LL;
25  (*p__9_string)[0].ptr = ".exe";
26  (*p__9_string)[1].len = 4LL;
27  (*p__9_string)[1].ptr = ".sys";
28  (*p__9_string)[2].len = 4LL;
29  (*p__9_string)[2].ptr = ".drv";
30  (*p__9_string)[3].len = 4LL;
31  (*p__9_string)[3].ptr = ".dll";
32  (*p__9_string)[4].len = 5LL;
33  (*p__9_string)[4].ptr = ".html";
34  (*p__9_string)[5].len = 4LL;
35  (*p__9_string)[5].ptr = ".txt";
36  (*p__9_string)[6].len = 9LL;
37  (*p__9_string)[6].ptr = ".bianlian";
38  (*p__9_string)[7].len = 4LL;
39  (*p__9_string)[7].ptr = ".mui";
40  (*p__9_string)[8].len = 4LL;
41  (*p__9_string)[8].ptr = ".lnk";
42  qword_625068 = 9LL;
43  qword_625070 = 9LL;
44  if ( dword_679FB0 )
45  {
46    a4 = &qword_625060;
47    runtime_gcWriteBarrier(&qword_625060, a5);
48  }
49  else
50  {
51    qword_625060 = (__int64)p__9_string;
52  }
53  p__4_string = (_4_string *)runtime_newobject(&RTYPE__4_string);
54  (*p__4_string)[0].len = 7LL;
55  (*p__4_string)[0].ptr = "Windows";
56  (*p__4_string)[1].len = 23LL;
57  (*p__4_string)[1].ptr = "AppData\\Local\\Microsoft";
58  (*p__4_string)[2].len = 7LL;
59  (*p__4_string)[2].ptr = "Webroot";
60  (*p__4_string)[3].len = 6LL;
61  (*p__4_string)[3].ptr = "Sophos";
62  qword_625088 = 4LL;
63  qword_625090 = 4LL;
    if ( dword_679FB0 )
```

Ilustración 14: declaración de los listados de directorios y extensiones a evitar cifrar

Tras comprobar que no es una extensión o directorio a evitar cifrar, si se trata de un directorio se llamará a la función ***main_ProcessDirectory_func3*** que se encargará de volver a llamar a la rutina *main_ProcessDirectory* de forma recursiva. Por el contrario, si se trata de un fichero, llamará a la función ***main_ProcessDirectory_func2*** que se encargará de procesarlo.

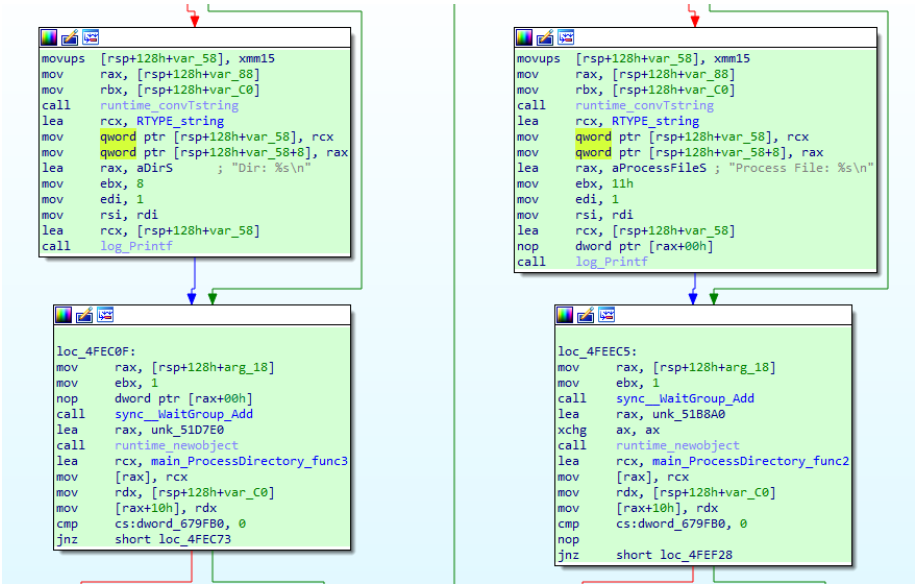


Ilustración 15: procesado de directorio y ficheros mediante funciones específicas

```
1  // main.ProcessDirectory.func3
2  __int64 __golang main_ProcessDirectory_fun
3          __int64 a1,
4          __int64 a2,
5          __int64 a3,
6          __int64 a4,
7          __int64 a5,
8          __int64 a6,
9          __int64 a7,
10         __int64 a8,
11         __int64 a9,
12         char a10)
13 {
14   __int64 v10; // rdx
15   __int64 v11; // r14
16   char **v12; // r12
17   __int64 v14; // [rsp-28h] [rbp-30h] BYRE
18   void *retaddr; // [rsp+8h] [rbp+0h] BYRE
19
20   if ( (unsigned __int64)&retaddr <= *(_QW
21     runtime_morestack(a4, a5);
22   v12 = *(char ***)(v11 + 32);
23   if ( v12 && *v12 == &a10 )
24     *v12 = (char *)&v14;
25   return main_ProcessDirectory(
26            *(_QWORD *)(v10 + 8),
27            *(_QWORD *)(v10 + 16),
28            *(_QWORD *)(v10 + 24),
29            *(_QWORD *)(v10 + 32));
30 }
```

Ilustración 16: llamada recursiva a main_ProcessDirectory desde main_ProcessDirectory_func3

Nota de rescate

Por cada directorio encontrado, la rutina main_ProcessDirectory llama a la rutina *main_LeaveInstructions* que se encarga de escribir en disco la nota de rescate. Esto únicamente ocurre cuando el ransomware no se está ejecutando con el parámetro de depuración.

```
LABEL_8:
  v30 = v13;
  v14 = os_OpenFile(v13, v10, 0, 0, v9, v11, v12, v6, v7);
  if ( v10 )
  {
    if ( debug_enabled )
    {
      v32 = v5;
      *(_QWORD *)&v32 = *(_QWORD *)(v10 + 8);
      *((_QWORD *)&v32 + 1) = v15;
      log_Println((unsigned int)&v32, 1, 1, 0, v9, v16, v17, v18, v19, v2
    }
  }
  else
  {
    v21 = os__ptr_File_Readdir(v14, 0, v15, 0, v9, v16, v17, v18, v19);
    if ( !debug_enabled )
    {
      v31 = v21;
      main_LeaveInstructions(v30, 0, v22, 0, v9, v23, v24, v25, v26);
    }
  }
  return (*v34)();
}
```

Ilustración 17: llamada a la función main_LeaveInstructions por cada directorio

La rutina *main_LeaveInstructions* simplemente se encarga de crear un fichero con nombre "*Look at this instruction.txt*" y escribir el contenido de la nota en él. Todos los valores de la nota se encuentran embebidos en el código y no hay ninguno, como por ejemplo el ID de la víctima, que sea calculado de forma dinámica.

```
  a1 = v78;
}
v77 = a1;
v10 = runtime_concatstring2(
        0,
        (unsigned int)"Your network systems were attacked and encrypted. Contact us in order to restore your ",
        229,
        (unsigned int)"To contact us you have to download \"tox\" messenger: https://qtox.githu",
        79,
        a6,
        a7,
        a8,
        a9);
v15 = runtime_concatstring2(
        0,
        v10,
        (unsigned int)"Your network systems were attacked and encrypted. Contact us in order to restore your ",
        (unsigned int)"Add user with the following ID to get your instructions:\r",
        58,
        v11,
        v12,
        v13,
        v14);
v16 = v10;
v17 = v15;
v22 = runtime_concatstring2(
        0,
        v15,
        v16,
        (unsigned int)"A4B3B0845DA242A64BF17E0DB4278EDF85855739667D3E2AE8889D5439015F07E81D12D767FC\r\n\r\n",
        80,
        v18,
        v19,
        v20,
        v21);
v23 = v17;
v24 = v22;
v29 = runtime_concatstring2(
```

Ilustración 18: rutina main_LeaveInstructions que escribe la nota en disco

Look at this instruction.txt

```
Your network systems were attacked and encrypted. Contact us
in order to restore your data. Don't make any changes in your
file structure: touch no files, don't try to recover by
yourself, that may lead to it's complete loss.

To contact us you have to download "tox" messenger:
https://qtox.github.io/

Add user with the following ID to get your instructions:
A4B3B0845DA242A64BF17E0DB4278EDF85855739667D3E2AE8B89D5439015F0
7E81D12D767FC

Alternative way: swikipedia@onionmail.org

Your ID: KARB4VvXbq

You should know that we have been downloading data from your
network for a significant time before the attack: financial,
client, business, post, technical and personal files.
In 10 days - it will be posted at our site http://bianlianlbc5a
n4kgnay3opdemgcryg2kpfcbgczopmm3dnbz3uaunad.onion / http://bian
livemqbawcco4cx4a672k2fip3guyxudzurfqvdszafam3ofqgqd.onion
with links send to your clients, partners, competitors and
news agencies, that will lead to a negative impact on your
company: potential financial, business and reputational loses.
   ---!!!---
```

Ilustración 19: nota de rescate escrita en disco

Rutina de cifrado

Por cada fichero encontrado en el sistema, la rutina *main_ProcessDirectory_func2* llama a la función ***main_ProcessFile*** para procesarlo.

```
// main.ProcessDirectory.func2
__int64 __golang main_ProcessDirectory_func2(
        __int64 a1,
        __int64 a2,
        __int64 a3,
        __int64 a4,
        __int64 a5,
        int a6,
        int a7,
        int a8,
        int a9,
        char a10)
{
  _QWORD *v10; // rdx
  __int64 v11; // r14
  char **v12; // r12
  __int64 v14[4]; // [rsp-20h] [rbp-28h] BYREF
  void *retaddr; // [rsp+8h] [rbp+0h] BYREF

  if ( (unsigned __int64)&retaddr <= *(_QWORD *)(v11 + 16) )
    runtime_morestack(a4, a5);
  v12 = *(char ***)(v11 + 32);
  if ( v12 && *v12 == &a10 )
    *v12 = (char *)v14;
  return main_ProcessFile(v10[1], v10[2], v10[3], v10[4], a5, a6, a7, a8, a9, v14[0], v14[1], v14[2], v14[3]);
}
```

Ilustración 20: llamada al procesado de ficheros

Cada fichero a procesar es abierto mediante la API de Go *OpenFile* de la librería *os*. A continuación, se obtiene el tamaño del fichero para decidir cómo proceder al cifrado. *BianLian* realiza un cifrado parcial de los ficheros y divide su lógica en función del tamaño de los ficheros. Para ficheros menores de 1KB, el tamaño de bloque a cifrar es de 16 bytes, mientras que para ficheros mayores de 1KB, se usa un tamaño de bloque de 4096 bytes.

```
    os__ptr_file_close(*p_file);
  goto LABEL_22;
}
filesize = (*((__int64 (__golang **)(void *))v94.0.tab + 7))(v94.0.data);
encrypt_size = (filesize / 4096) << 12;
if ( filesize > 0x400000 )
{
  v28 = filesize * (__int128)(__int64)0xCCCCCCCCCCCCCCCDLL;
  v94.1.data = (void *)((unsigned __int64)((*((_QWORD *)&v28 + 1) + filesize) >> 63) >> 52);
  encrypt_size = ((__int64)v94.1.data + ((*((_QWORD *)&v28 + 1) + filesize) >> 3)) >> 12 << 12;
  if ( encrypt_size >= 0x400000 )
    encrypt_size = 0x400000LL;
}
if ( encrypt_size < 16 )
  encrypt_size = 16LL;
v77 = encrypt_size;
ChunkCount = mw_GetChunkCount((__int64)v94.1.data, qword_605450, encrypt_size, qword_605450);
v79 = off_61CDD0;
v71 = qword_61CDD8;
```

Ilustración 21: cálculo del tamaño de bloque

A continuación, se obtienen a partir de variables globales tanto la clave de cifrado AES como el vector de inicalización IV y se procede a leer el contenido del fichero mediante la API de Go *File.ReadAt*.

Ilustración 22: inicialización de la clave de fichero y lectura del fichero por bloques

Como se ha comentado anteriormente, la clave de cifrado y el vector de inicalización IV se encuentran embebidas en el binario y van cambiando para cada muestra. En el caso de la muestra analizada, los valores embebidos son los siguientes (en hexadecimal):

- **Clave de cifrado**:

26dd1b400aed80b0980b34bad76d3be6599123fa562266b421a14ac8e02ecfa3

- **IV**:

d518ba928469306d579d625f56d09883

Estos valores junto al contenido del bloque leído en cada iteración del fichero son transferidos a la función de cifrado ***main_encrypt***. Esta función simplemente realiza el cifrado AES-256 en modo CBC del contenido del bloque.

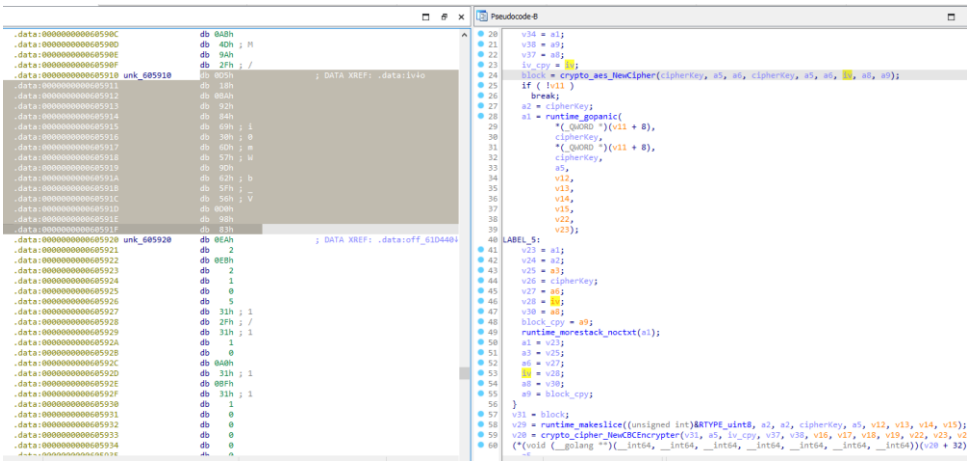


Ilustración 23: función de cifrado AES main_encrypt

Finalmente, el fichero es renombrado con la extensión ".bianlian" una vez se ha cifrado su contenido y escrito en disco.

```
00052AB92                                       ; runtime__headTailIndex_incT
00052AB9B aBackupdb     db '.backupdb'          ; DATA XREF: .rdata:000000000
00052ABA4 aBianlian     db '.bianlian'          ; DATA XREF: main_init_0+9E↑o
00052ABA4                                       ; .data:off_61CBA0↓o
00052ABAD aChangedb     db '.changedb'          ; DATA XREF: .rdata:000000000
00052ABB6 aComicdoc     db '.comicdoc'          ; DATA XREF: .rdata:000000000
00052ABBF aDaschema     db '.daschema'          ; DATA XREF: .rdata:000000000
00052ABC8 aFwbackup     db '.fwbackup'          ; DATA XREF: .rdata:000000000
00052ABD1 aJtbackup     db '.jtbackup'          ; DATA XREF: .rdata:000000000
00052ABDA aMailhost     db '.mailhost'          ; DATA XREF: .rdata:000000000
00052ABE3 aMdbackup     db '.mdbackup'          ; DATA XREF: .rdata:000000000
00052ABEC aMenudata     db '.menudata'          ; DATA XREF: .rdata:000000000
00052ABF5 aMetadata     db '.metadata'          ; DATA XREF: .rdata:000000000
00052ABFE aMindnode     db '.mindnode'          ; DATA XREF: .rdata:000000000
00052AC07 aNotebook     db '.notebook'          ; DATA XREF: .rdata:000000000
00052AC10 aSqlitedb     db '.sqlitedb'          ; DATA XREF: .rdata:000000000
00052AC19 aThumbsdb     db '.thumbsdb'          ; DATA XREF: .rdata:000000000
00052AC22 aTlbackup     db '.tlbackup'          ; DATA XREF: .rdata:000000000
00052AC2B aVboxsave     db '.vboxsave'          ; DATA XREF: .rdata:000000000
00052AC34 aVmwarevm     db '.vmwarevm'          ; DATA XREF: .rdata:000000000
00052AC3D aXmlPrev      db '.xml-prev'          ; DATA XREF: .rdata:000000000
00052AC46 a200132       db '2001::/32'          ; DATA XREF: net_init:loc_4ED
00052AC4F a200216       db '2002::/16'          ; DATA XREF: net_init:loc_4ED
00052AC58 a244140625    db '244140625'          ; DATA XREF: .data:0000000000
00052AC61 a3ffe16       db '3ffe::/16'          ; DATA XREF: net_init:loc_4ED
00052AC6A aStatus_1     db ': status='          ; DATA XREF: runtime_schedtra
00052AC6A                                       ; runtime_schedtrace+386fo
00052AC73 aAuthority    db 'Authority'          ; DATA XREF: vendor_golang_or
00052AC73                                       ; vendor_golang_org_x_net_dns
00052AC7C aBassaVah     db 'Bassa_Vah'          ; DATA XREF: unicode_init+851
00052AC85 aBhaiksuki    db 'Bhaiksuki'          ; DATA XREF: unicode_init+C11
00052AC8E aClassinet    db 'ClassINET'          ; DATA XREF: vendor_golang_or
00052AC8E                                       ; vendor_golang_org_x_net_dns
00052AC97 aCuneiform    db 'Cuneiform'          ; DATA XREF: unicode_init+FD1
00052ACA0 aDiacritic    db 'Diacritic'          ; DATA XREF: unicode_init+317
00052ACA9 aFindclose    db 'FindClose'          ; DATA XREF: syscall_init+1C8
00052ACB2 aHexDigit     db 'Hex_Digit'          ; DATA XREF: unicode_init+31F
00052ACBB aInherited    db 'Inherited'          ; DATA XREF: unicode_init+16D
```

```
        arg8);
      if ( p_file )
        os__ptr_file_close(*p_file);
      goto LABEL_22;
    }
  }
  chunk_index = index + 1;
  LODWORD(buffer) = buffer_slice;
}
if ( p_file )
  os__ptr_file_close(*p_file);
new_file_path = qword_61CBA8;
renamed_file_path = runtime_concatstring2(
                      0,
                      v77,
                      0,
                      (_DWORD)off_61CBA0,
                      qword_61CBA8,
                      offset_high,
                      offset_low,
                      num_bytes_read_high,
                      num_bytes_read_low);
rename_err = project1_common_FileRename(
               v77,
               0,
               renamed_file_path,
               v77,
               new_file_path,
               v52,
               v53,
               v54,
               v55,
               arg7,
               arg8,
               arg9,
               v60);
((void (__golang *)(__int64))*callback)(rename_err);
return 1LL;
```

Ilustración 24: renombrado del fichero cifrado

Herramienta de descifrado de Avast

Dada la debilidad criptográfica que presenta este ransomware, el fabricante de software de seguridad Avast ha realizado una recopilación de las claves de cifrado e IV de diferentes muestras y publicado un descifrador para dichas variantes.

Para poder utilizarlo, se debe introducir una muestra de un fichero cifrado y su versión original o, alternativamente, introducir directamente los valores de la clave de cifrado e IV si se conocen.

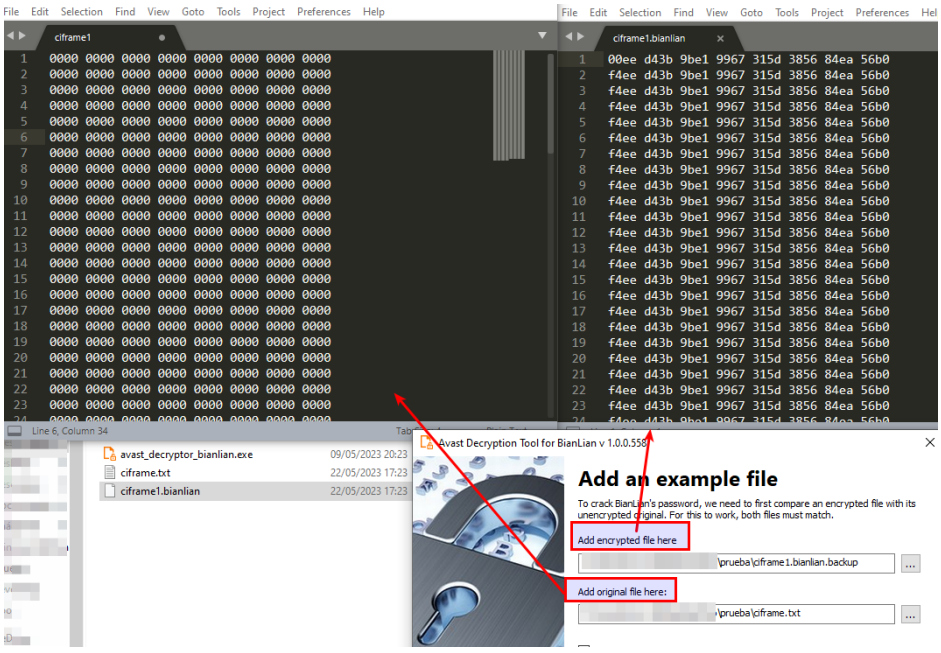


Ilustración 25: herramienta de descifrado de BianLian de Avast

Tras esto, si no se conoce la clave, iniciaría el proceso de búsqueda hasta encontrarla.

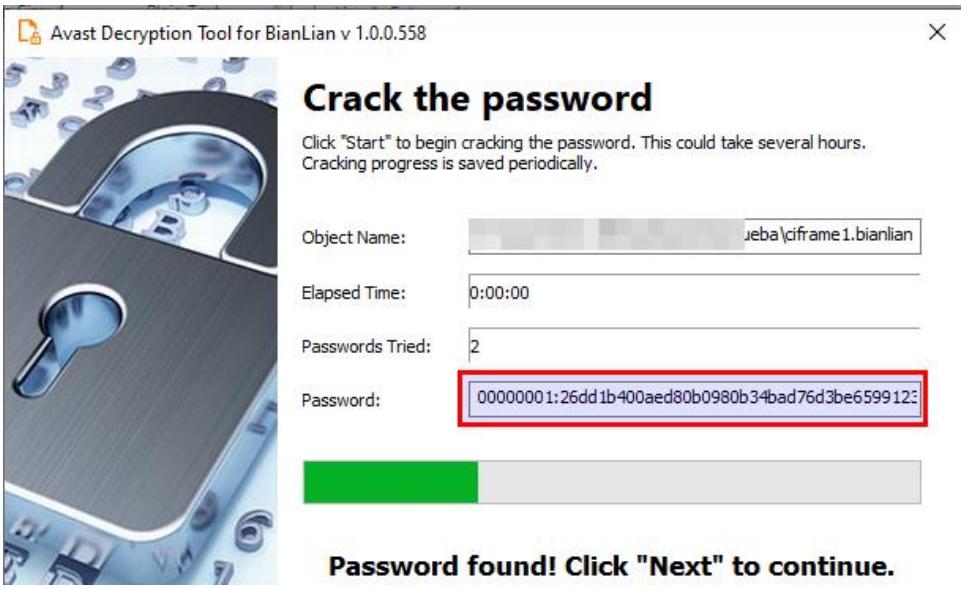


Ilustración 26: clave de cifrado e IV encontrados por el descifrador de Avast

Y, finalmente, se descifraría el fichero por bloques mediante AES. A continuación, puede observarse un ejemplo de cómo se descifraría cada bloque de 16 bytes en este caso.

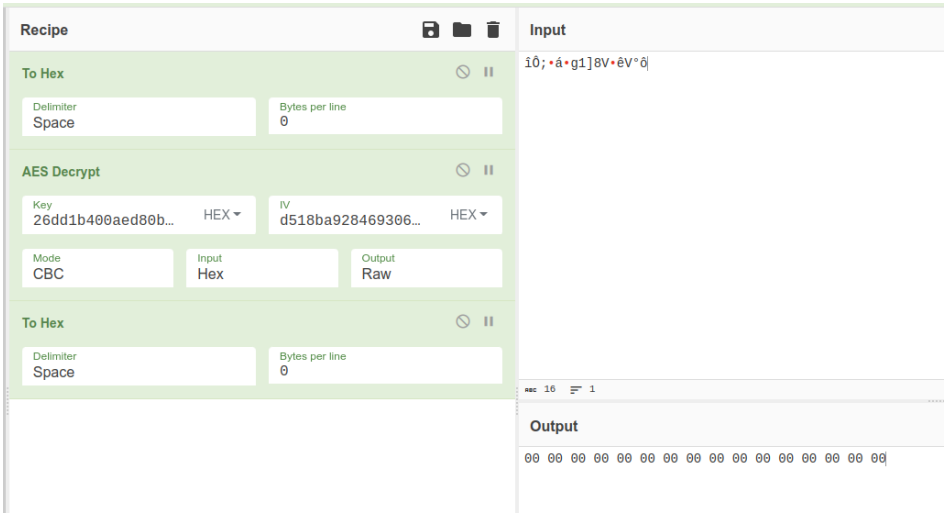


Ilustración 27: ejemplo de descifrado AES para 16 bytes con los valores de clave e IV de la muestra analizada

El binario descifrador de Avast puede descargarse desde la URL: https://files.avast.com/files/decryptor/avast_decryptor_bianlian.exe

Vulnerabilidades explotadas

La vulnerabilidad conocida que los actores detrás de BianLian han podido explotar es la CVE-2020-1472 cuyos detalles son los siguientes:

CVE-2020-1472: vulnerabilidad de elevación de privilegios que se da cuando un atacante establece una conexión de canal seguro de Netlogon vulnerable en un controlador de dominio, utilizando el protocolo remoto de Netlogon (MS-NRPC), también conocido como vulnerabilidad de elevación de privilegios de Netlogon.

La métrica de evaluación de la vulnerabilidad se compone de:

CWE 330: Use of Insufficiently Random Values

CVSS Base: 10.0 (**crítica**)

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H

- **Vector de ataque**: **Red**
- **Complejidad del ataque**: **Baja**
- **Privilegios requeridos: Ningunos**
- **Interacción con el usuario**: **Ninguna**
- **Alcance**: **Con cambios**
- **Confidencialidad: Alta**
- **Integridad: Alta**
- **Disponibilidad: Alta**

MITRE ATT&CK			
Execution	T1204.002	Malicious File	**M1038: Execution Prevention** Application control may be able to prevent the running of executables masquerading as other files.
			M1040: Behavior Prevention on Endpoint On Windows 10, various Attack Surface Reduction (ASR) rules can be enabled to prevent the execution of potentially malicious executable files (such as those that have been downloaded and executed by Office applications/scripting interpreters/email clients or that do not meet specific prevalence, age, or trusted list criteria). Note: cloud-delivered protection must be enabled for certain rules. (Citation: win10_asr)
			M1017: User Training Use user training as a way to bring awareness to common phishing and spearphishing techniques and how to raise suspicion for potentially malicious events.
	T1204	User Execution	**M1040: Behavior Prevention on Endpoint** On Windows 10, enable Attack Surface Reduction (ASR) rules to prevent executable files from running unless they meet a prevalence, age, or trusted list criteria and to prevent Office applications from creating potentially malicious executable content by blocking malicious code from being written to disk. Note: cloud-delivered protection must be enabled to use certain rules. (Citation: win10_asr)

			M1038: Execution Prevention Application control may be able to prevent the running of executables masquerading as other files.
			M1021: Restrict Web-Based Content If a link is being visited by a user, block unknown or unused files in transit by default that should not be downloaded or by policy from suspicious sites as a best practice to prevent some vectors, such as .scr, .exe, .pif, .cpl, etc. Some download scanning devices can open and analyze compressed and encrypted formats, such as zip and rar that may be used to conceal malicious files.
			M1031: Network Intrusion Prevention If a link is being visited by a user, network intrusion prevention systems and systems designed to scan and remove malicious downloads can be used to block activity.
			M1017: User Training Use user training as a way to bring awareness to common phishing and spearphishing techniques and how to raise suspicion for potentially malicious events.
	T1106	Native API	**M1040: Behavior Prevention on Endpoint** On Windows 10, enable Attack Surface Reduction (ASR) rules to prevent Office VBA macros from calling Win32 APIs. (Citation: win10_asr)
			M1038: Execution Prevention Identify and block potentially malicious software executed that may be executed through this technique by using application control (Citation: Beechey 2010) tools, like Windows Defender Application Control(Citation: Microsoft Windows Defender Application Control), AppLocker, (Citation: Windows Commands JPCERT) (Citation: NSA MS AppLocker) or

			Software Restriction Policies (Citation: Corio 2008) where appropriate. (Citation: TechNet Applocker vs SRP)
	T1059	Command and Scripting Interpreter	**M1040: Behavior Prevention on Endpoint** On Windows 10, enable Attack Surface Reduction (ASR) rules to prevent [Visual Basic](https://attack.mitre.org/techniques/T1059/005) and [JavaScript](https://attack.mitre.org/techniques/T1059/007) scripts from executing potentially malicious downloaded content (Citation: win10_asr).
			M1042: Disable or Remove Feature or Program Disable or remove any unnecessary or unused shells or interpreters.
			M1038: Execution Prevention Use application control where appropriate.
			M1045: Code Signing Where possible, only permit execution of signed scripts.
			M1049: Antivirus/Antimalware Anti-virus can be used to automatically quarantine suspicious files.
			M1026: Privileged Account Management When PowerShell is necessary, restrict PowerShell execution policy to administrators. Be aware that there are methods of bypassing the PowerShell execution policy, depending on environment configuration.(Citation: Netspi PowerShell Execution Policy Bypass)
			M1021: Restrict Web-Based Content Script blocking extensions can help prevent the execution of scripts and HTA files that may commonly be used during the exploitation process. For malicious code served up through ads, adblockers can help prevent that code from executing in the first place.
	T1059.003	Windows Command Shell	**M1038: Execution Prevention** Use application control where appropriate.

Discovery	T1135	Network Share Discovery	**M1028: Operating System Configuration** Enable Windows Group Policy “Do Not Allow Anonymous Enumeration of SAM Accounts and Shares” security setting to limit users who can enumerate network shares.(Citation: Windows Anonymous Enumeration of SAM Accounts)
	T1083	File and Directory Discovery	**This type of attack technique cannot be easily mitigated with preventive controls since it is based on the abuse of system features.**
	T1057	Process Discovery	**This type of attack technique cannot be easily mitigated with preventive controls since it is based on the abuse of system features.**
Impact	T1486	Data Encrypted for Impact	**M1040: Behavior Prevention on Endpoint** On Windows 10, enable cloud-delivered protection and Attack Surface Reduction (ASR) rules to block the execution of files that resemble ransomware. (Citation: win10_asr)
			M1053: Data Backup Consider implementing IT disaster recovery plans that contain procedures for regularly taking and testing data backups that can be used to restore organizational data.(Citation: Ready.gov IT DRP) Ensure backups are stored off system and is protected from common methods adversaries may use to gain access and destroy the backups to prevent recovery. Consider enabling versioning in cloud environments to maintain backup copies of storage objects.(Citation: Rhino S3 Ransomware Part 2)

Mitigación

Medidas a nivel de endpoint

El código de *BianLian* no se encuentra firmado, por lo que implementar una política que no permita la ejecución de binarios que no estén firmados podría prevenir la ejecución de este malware. No obstante, gran cantidad de desarrolladores y paquetes de software no distribuyen sus productos firmados, por lo que esta estrategia podría no resultar práctica en algunos casos.

En concordancia con lo anterior, pero empleando mecanismos más generales, se recomienda que las organizaciones prohíban o, al menos, monitoricen la ejecución de binarios no conocidos previamente dentro de ella o aquellos no provenientes de fuentes confiables. Aunque imperfecto, por la forma en la que se crea y distribuye el software legítimo, esta medida puede servir como una alarma inicial para impulsar una mayor investigación y, posiblemente, limitar su propagación.

Con el objetivo de disminuir el tiempo de reacción frente a este tipo de amenazas se recomienda mantener vigilado el *endpoint* con soluciones de monitorización y de antivirus/EDR así como disponer de una política de actualizaciones que mantenga el *endpoint* con las últimas vulnerabilidades.

Medidas a nivel de red

Si se dispone de los mecanismos para inspeccionar el tráfico que ocurre hacia fuera de la red, se debería identificar comunicaciones anómalas o que tengan similitudes con familias de malware ya conocidas. De esta forma se puede identificar de forma rápida y eficiente posibles máquinas infectadas dentro de la red.

Medidas y consideraciones adicionales

Se deben enviar todos los eventos del sistema, o al menos los más importantes, a un sistema externo que reúna todos los eventos de todos los equipos de la red. De esta forma se puede evitar la pérdida de trazabilidad. Además, esta mitigación podría ayudar a crear alertas tempranas que avisen de una posible intrusión en el sistema y de esta forma evitar el ataque.

Se debe mantener una política de actualizaciones. Es de suma importancia que todos los sistemas se encuentren totalmente actualizados para evitar posibles vulnerabilidades de seguridad que los atacantes puedan explotar para hacerse con el control de una máquina, obtener credenciales o realizar una escalada de privilegios.

Se debe eliminar cualquier contraseña por defecto establecida en cualquier sistema o aplicación, además de generar una política de contraseñas que obligue al uso de contraseñas seguras y que cambien de forma periódica. Aplicar sistemas de autenticación en dos pasos en todos aquellos sistemas que lo permitan.

Se debe mantener al equipo de seguridad actualizado de todas las nuevas vulnerabilidades conocidas, que tengan conocimiento de todos los sistemas utilizados en el parque tecnológico y que decidan si es necesario aplicar medidas de mitigación adicionales antes situaciones específicas.

En caso de incidente con este *malware*, se debe de reportar a las autoridades pertinentes lo más rápido posible.

Indicadores de compromiso

Los indicadores de compromiso y reglas de detección también están disponibles para su consulta y descarga en el repositorio público del Basque Cybersecurity Centre:

https://github.com/basquecscentre/technical-reports

Hashes

- 117a057829cd9abb5fba20d3ab479fc92ed64c647fdc1b7cd4e0f44609d770ea
- 1fd07b8d1728e416f897bef4f1471126f9b18ef108eb952f4b75050da22e8e43
- 3a2f6e614ff030804aa18cb03fcc3bc357f6226786efb4a734cbe2a3a1984b6f
- 46d340eaf6b78207e24b6011422f1a5b4a566e493d72365c6a1cace11c36b28b
- cbab4614a2cdd65eb619a4dd0b5e726f0a94483212945f110694098194f77095
- eaf5e26c5e73f3db82cd07ea45e4d244ccb3ec3397ab5263a1a74add7bbcb6e2

Yara:

- Estas reglas sirven para identificar las muestras de la familia *BianLian* de Windows. Se tratan de reglas yara procedentes del equipo de analistas de Blackberry:

YARA

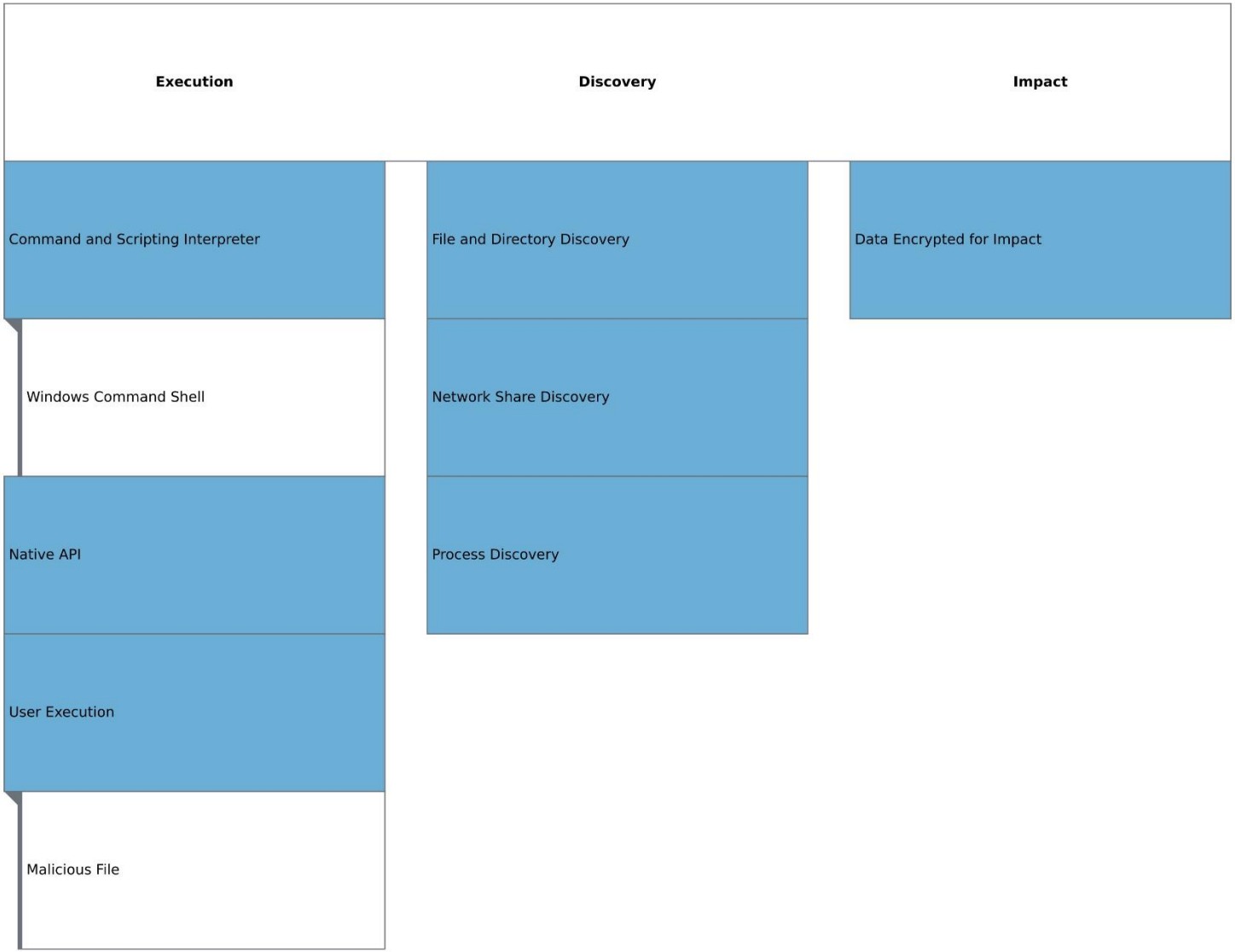
```
rule BianLian_Go_Ransomware{
      meta:
            description = "Detects BianLian ransomware"
            author = "BlackBerry Threat Research Team"
            date = "2022-09-13"
            license = "This Yara rule is provided under the Apache
License 2.0 (https://www.apache.org/licenses/LICENSE-2.0) and open to
any user or organization, as long as you use it under this license and
ensure originator credit in any derivative to the BlackBerry Research &
Intelligence Team"
      strings:
            $s1 = "trimpath=/home/jack/Projects/project1/"
            $s2 = "common.BuildPath"
            $s3 = "common.GetBlocksAmount"
            $s4 = "common.GetDrives"
            $s5 = "common.GetBlockSize"
            $s6 = "common.FileRename"
            $s7 = "common.GetFileExtension"
            $s8 = "exec.(*Cmd).Start.func1"
```

```
            $s9 = "exec.(*Cmd).Start.func2"
            $s10 = "exec.(*Cmd).Start.func3"
            $s11 = "CryptBlocks"
      condition:
            uint16(0) == 0x5a4d and all of them
}
```

Referencias adicionales

- https://malpedia.caad.fkie.fraunhofer.de/details/win.bianlian
- https://twitter.com/malwrhunterteam/status/1557789273595731969
- https://blog.cyble.com/2022/08/18/bianlian-new-ransomware-variant-on-the-rise/
- https://redacted.com/blog/bianlian-ransomware-gang-gives-it-a-go/
- https://resources.securityscorecard.com/research/bian-lian-deep-dive
- https://blogs.blackberry.com/en/2022/10/bianlian-ransomware-encrypts-files-in-the-blink-of-an-eye
- https://decoded.avast.io/threatresearch/decrypted-bianlian-ransomware/
- https://www.cisa.gov/sites/default/files/2023-05/aa23-136a_stopransomware_bianlian_ransomware_group_1.pdf
- https://www.bleepingcomputer.com/news/security/avast-releases-free-bianlian-ransomware-decryptor/

Apéndice A: Mapa de técnicas de ATT&CK



EUSKO JAURLARITZA GOBIERNO VASCO
GRUPO spri TALDEA